TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Regina Johanson 193954IABB

VEEBIRAKENDUSTE TESTIMISPROTSESSI TÄIUSTAMISE VÕIMALUSED ETTEVÕTTES ELEPORT INNOVATIONS OÜ

Bakalaureusetöö

Juhendaja: Inna Švartsman
Magistrikraad

Kaasjuhendaja: Ivari Horm
Bakalaureusekraad

Tallinn 2023

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Regina Johanson

16.05.2023

Annotatsioon

Veebirakenduste testimine on tarkvaraarenduses kriitiline protsess, kuna see tagab, et rakendus vastab selle kavandatud kasutajate kvaliteedistandarditele ja nõuetele. Käesoleva lõputöö eesmärk on pakkuda välja Eleport Innovations OÜ veebirakenduste testimisprotsessi täiustamise võimalused. Ettevõttes on olemasolev testimissüsteem, kuid see vajab tõhustamist ja automatiseerimist. Lahenduse kavandamise eelduseks oli Eleport Innovationsis uuringu läbi viimine.

Uuring sisaldas praegu ettevõttes kasutatavate testimismeetodite, tööriistade ja rollide kindlaks määramist. Selleks küsiti ettevõtte töötajate hinnanguid ja arvamusi veebirakenduste testimise osas, et teha kindlaks praegune testimisprotsess ja väljakutsed, millega nad silmitsi seisavad, ning tuua välja täiendusvaldkonnad. Samuti hõlmas uuring kirjanduse ülevaadet veebirakenduste testimise, sealhulgas selle tüüpide, meetodite ja tööriistade kohta.

Tulemuste põhjal töötati välja täiustatud veebirakenduste testimise protsess, mis hõlmab automatiseeritud testimist, testimistööriistade kasutamist ning paremat koostööd testimis- ja arendusmeeskonna vahel. Kavandatud protsessi hinnati ettevõtte esindaja ehk tehnoloogiajuhi poolt. Lõputöö tulemusi saab kasutada näidisena teistele väikeettevõtetele, kes soovivad oma veebirakenduste testimise protsessi täiustada.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 41 leheküljel, 6 peatükki, 5 joonist, 6 tabelit.

Abstract

Opportunities for improving the testing process of web applications in Eleport Innovations OÜ

Web application testing is a critical process in software development as it ensures that the application meets the quality standards and requirements of its intended users. The purpose of this thesis is to propose possibilities for improving the testing process of web applications of Eleport Innovations OÜ. The company has an existing testing system, but it needs to be improved and automated. Conducting a study at Eleport Innovations was a prerequisite for designing a solution.

The study involved identifying the testing methods, tools and roles currently used in the company. This was done by gathering opinions and assessments from employees regarding web application testing in order to determine the current testing process and the challenges they face, and to identify areas for improvement. The study also included a literature review of web application testing, including its types, methods and tools.

Based on the results, an improved web application testing process was developed, which includes automated testing, the use of testing tools, and better collaboration between the testing and development teams. The proposed process was evaluated by a representative of the company, i.e. the company's chief technology officer. The results of this thesis can be used as an example for other small businesses who wish to improve their web application testing process.

The thesis is in Estonian and contains 41 pages of text, 6 chapters, 5 figures, 6 tables.

Lühendite ja mõistete sõnastik

API	*Application Programming Interface,* rakenduse programmeerimisliides
Confluence	Atlassiani veebipõhine koostöö- ja dokumenteerimistööriist
CPM	*Charge Point Management,* Laadimispunktide haldus
EV	*Electric Vehicle,* elektrisõiduk
EVSE	*Electric Vehicle Supply Equipment,* elektrisõiduki toiteseade. Seade, mis on võimeline laadima ühte elektrisõidukit. Üks EV laadija sisaldab ühte või mitut EVSE-d.
Jira Service Management	Atlassiani pilvepõhine IT-teenuste haldustarkvara
Jira Software	Atlassiani projektihaldustööriist, mida kasutatakse tarkvaraarendusprotsesside jälgimiseks ja haldamiseks.
QA	*Quality Assurance,* kvaliteedi tagamine
QC	*Quality Control,* kvaliteedikontroll
RP	*Registration Page,* Registreerimisleht
SDLC	*Software Development Life Cycle,* tarkvaraarenduse elutsükkel
SPP	*Single Payment Portal,* Ühekordse makse portaal
STLC	*Software Testing Life Cycle,* tarkvara testimise elutsükkel
OÜ	Osaühing

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Elektrisõidukite (EV) arv Eestis on viimastel aastatel kiiresti kasvanud ning sõidukite laadimisnõudlus on viinud EV laadimise tarkvararakenduste arendamiseni. Need veebirakendused võimaldavad EV omanikel leida ja broneerida laadimisjaamu, jälgida laadimissessioone ja hallata makseid. Kuna need rakendused muutuvad EV ökosüsteemi jaoks üha kriitilisemaks, on nende funktsionaalsuse, turvalisuse ja usaldusväärsuse tagamine muutunud üha olulisemaks.

Eleport Innovations OÜ (osaühing) on tarkvaraettevõtte, mis on üle pooleteise aasta arendanud EV laadimisjaamade haldus- ja kasutajatarkvara Eleport OÜ-le. Eleport OÜ pakub EV laadimisteenust. Laadimistarkvara kasutajaskonna suurenemisega ja konkurentsis püsimiseks on äärmisel oluline rakenduste testimine nii funktsionaalsuse kui ka kasutajakogemuse aspektist.

Käesoleva lõputöö eesmärk on pakkuda välja täiustatud protsessiplaan veebirakenduste testimiseks Eleport Innovations OÜ-s, keskendudes agiilsele (*agile*) metoodikale. Eesmärgi saavutamiseks on seatud mitmed ülesanded, mille hulka kuulub ettevõtte veebirakenduste testimise hetkeseisu uurimine ja hindamine, peamiste probleemide ja väljakutsete tuvastamine ning täiustamisvõimaluste uurimine. Töö uurib erinevaid testimise tüüpe, sealhulgas dünaamilist, funktsionaalset ja mittefunktsionaalset testimist, samuti manuaalseid ja automatiseeritud testimismeetodeid, testimistööriistu ja raamistikke. Samuti hinnatakse kavandatavate täiustuste teostatavust ja nende võimalikku mõju tarkvara testimisprotsessile ja EV laadimistarkvara kvaliteedile.

Ettevõtte mobiilirakenduste testimine ei kuulu selle lõputöö ulatusse, kuna see nõuab veebirakenduste testimisest erinevaid testimistehnikaid, tööriistu ja raamistikke ning on platvormipõhine. Töö põhineb ettevõtte andmetel ja infol, mis võib olla kallutatud ja ebatäpne. Lisaks on uuringute läbiviimiseks piiratud ressursid ja aeg, mis võib mõjutada analüüsi ja hindamise sügavust ja ulatust. Lõputöö tugineb olemasolevale kirjandusele ja valdkonna parimatele tavadele, et anda terviklik arusaam veebirakenduste testimisest.

Siiski ei pruugi see täielikult hõlmata elektrisõidukite laadimistarkvara ja veebirakenduste testimise ainulaadseid väljakutseid ja võimalusi.

Töö on üles ehitatud kuueks peatükiks, millest esimene peatükk annab ülevaate töö eesmärgist, ulatusest ja piirangutest ning kirjeldab töös kasutatud teaduslikke metoodikaid. Teine peatükk kirjeldab veebirakenduste testimise hetkeseisu ettevõttes, kolmas peatükk aga uurib erinevaid veebirakenduse testimise lähenemisviise, tehnikaid, meetodeid, tööriistu, rolle ja testimiseks vajalikku dokumentatsiooni. Neljas peatükk määrab nõuded kolmandas peatükis uuritud testimisprotsessi komponentide hindamiseks. Viies peatükk kavandab veebirakenduste testimisprotsessi täiustatud plaani ning kuues peatükk teeb lõputööst kokkuvõtte.

1.1 Metoodika

Lõputöö eesmärgi saavutamiseks kasutati erinevaid teaduslikke meetodeid ja vahendeid. Töö on jagatud neljaks eraldiseisvaks osaks ning iga osa nõudis ainulaadset metoodikat uurimistööks, andmete kogumiseks ja analüüsiks. Need neli osa on taustainfo uurimine, valdkonna uurimine, andmeanalüüs ja täiustuste kavandamine.

Taustainfo uurimine hõlmas Eleport Innovations OÜ veebirakenduste testimisprotsessi hetkeseisu uurimist. Selle saavutamiseks kasutati kvalitatiivset uurimismeetodit [1]. Tutvuti olemasoleva kirjandusega ning koguti algandmeid ettevõtte töötajatelt. Andmete kogumiseks kasutati hinnanguid, arvamusi ja tarkvara testimisprotsessi tähelepanekuid.

Valdkonna uuringud seevastu keskendusid tarkvara testimise ja veebirakenduste testimise olemuse uurimisele. Selleks kasutati kvalitatiivset andmekogumismeetodit [2]. Uuriti veebirakenduste testimise erinevaid komponente, sealhulgas testitüüpe, rolle, tööriistu ja dokumentatsiooni. Uuringu andmete kogumine põhines asjakohase kirjanduse, valdkonna parimate tavade ja veebirakenduste testimisega seotud veebiallikate ulatuslikul ülevaatel.

Andmeanalüüs hõlmas valdkonna uuringutest saadud teabe hindamist ja analüüsi veebirakenduste testimisprotsessi täiustuste kavandamise jaoks. Andmete analüüsimisel kasutati kvalitatiivset andmeanalüüsi lähenemist, mis hõlmas kogutud andmete kategoriseerimist, hindamist ja tõlgendamist [2]. Uurimistulemused andsid soovitusi parenduste kavandamiseks.

Lõputöö täiustuste kavandamise osa hõlmas veebirakenduste testimisprotsessi täiustamise võimaluste kavandi väljatöötamist. Selle saavutamiseks kasutati andmeanalüüsi järeldusi. Selles osas kasutatud metoodika oli agiilne metoodika [3], mis pakkus paindlikku lähenemist täiustuste kavandamisel ja nende implementeerimisel.

2 Ettevõtte veebirakenduste testimise taustainfo

Testimisprotsessi täiustamiseks on oluline mõista hetkeolukorda ettevõtte veebirakenduste testimisprotsessis. Käesolev peatükk annab ülevaate Eleport Innovations OÜ veebirakenduste testimise taustainfost. See on jaotatud kaheks alapeatükiks, millest esimene kirjeldab ettevõtte veebirakendusi. Teine peatükk kirjeldab lähemalt olemasolevaid testimistavasid, sealhulgas rollid, dokumentatsioon, testimistööriistad ja -keskkonnad ning meeskonna peamised probleemid ja väljakutsed.

2.1 Ettevõtte veebirakendused

Eleport Innovationsi veebiarendus hõlmab kolme rakendust: Laadimispunktide haldusportaal (CPM), Registreerimisleht (RP) ja Ühekordse makse portaal (SPP). Laadimispunktide haldamise veebirakendus pakub mitmesuguseid funktsioone, sealhulgas klientide ja objektide haldamist, tehingu algatamist, tehingutulemuste visualiseerimist ja API-teenuseid (*Application Programming Interface*) mobiilirakenduste ja kolmandate osapoolte integratsioonide jaoks. See on oluline tööriist, mida Eleporti töötajad kasutavad klientide ja EV laadijate andmebaasi haldamiseks ning selle nõuetekohane toimimine on ettevõtte igapäevatöös kriitilise tähtsusega. CPM-i tarkvara rike või defekt võib oluliselt häirida töötajate tootlikkust. Seetõttu on ülioluline, et CPM-i veebirakendus läbiks põhjaliku ja korrapärase testimise, et tagada selle tõrgeteta töö.

Registreerimisleht[1] võimaldab registreeruda nii era- kui äriklientidel, pakkudes kasutajakontoga ligipääsu Eleporti mobiilirakendusele, et nutiseadmega laadimist alustada. Ühekordse makse portaal[2] pakub kasutajatele, kellel ei ole kontot registreeritud, võimalust laadida oma elektrisõidukeid ainult laadija EVSE (*Electric Vehicle Supply Equipment*) koodi ja maksekaardi numbriga. RP ja SPP on CPM-iga tihedalt integreeritud ning kõik CPM-iga seotud probleemid võivad põhjustada probleeme ka nende rakendustega. Näiteks võib CPM-i tõrge või seisak takistada uutel kasutajatel Registreerimislehel registreeruda või takistada elektrisõidukite SPP kaudu laadimistasu võtmist. Need probleemid võivad põhjustada klientide rahulolematust, kahjustada

[1] https://register.eleport.com/
[2] https://charge.eleport.com/

ettevõtte mainet ja viia klientide kaotuseni. Seetõttu on oluline tagada, et CPM, RP ja SPP läbiksid põhjaliku testimise, et tagada veebirakenduste sujuv ja usaldusväärne töö.

2.2 Veebirakenduste testimise hetkeseis

Eleport Innovations kasutab kolme erinevat keskkonda: arendus-, test- ja toodangukeskkond. Iga keskkond kasutab eraldi andmebaasi. Arenduskeskkond kasutab arendusandmebaasi, test- ja toodangukeskkond aga reaalsete andmetega baasi. Testimine toimub testkeskkonnas reaalsete andmetega, kuna osa testjuhtumeid ei saa testandmetega lõpule viia, näiteks ärikliendi registreerimine e-äriregistri andmetega.

Veebirakenduste testimisprotsess tugineb suuresti manuaaltestimisel, mida viivad läbi muude põhiülesannetega Eleport Innovationsi töötajad, Eleport OÜ piirkondlikud töötajad ja testkliendid. Sellest lähenemisest ei pruugi aga piisata, kuna see võib kaasa tuua tarkvara tõhusaks testimiseks vajalike erialaste teadmiste ja oskuste puudumise. See võib potentsiaalselt mõjutada tarkvarasüsteemi kvaliteeti, töökindlust ja turvalisust, põhjustades probleeme lõppkasutajatele.

Arenduskeskkonnas viivad testimise läbi tarkvaraarendajad ise, mis ei pruugi samuti olla kõige tõhusam lähenemine. Arendajatel võib süsteemi tundmise tõttu olla pimealasid või eelarvamusi ning nad ei pruugi olla motiveeritud leidma vigu, mis võiksid kahjustada nende tööd või takistada nende edenemist. Seevastu tõhus testimine nõuab spetsiifilist psühholoogilist profiili, mis hõlmab rutiinitaluvust, soovi leida vigu ja olla detailidele orienteeritud.

Ettevõttel puudub spetsiaalne testimismeeskond, mistõttu ei teostata testimist kvaliteetse tarkvara tagamiseks vajaliku asjatundlikkuse ja rangusega. Kuigi testimist viivad läbi arendajad ja manuaaltestijad, ei pruugi see olla piisav, et tagada veebirakenduse funktsionaalsus ja kasutajate rahulolu.

Ülesannete haldamiseks kasutab ettevõte Jira Software'i projektihaldustööriista. Tehniline kirjutaja loob tarkvara väljalasketeate (*release note*) põhjal iga ülesande jaoks testjuhtumi, kasutades Confluence'i dokumendikeskkonda. Testjuhtumite järgi testivad rakendust manuaaltestijad ning neilt tagasiside kogumiseks kasutatakse Jira Service Management haldustarkvara. Seejärel teeb tootejuht testide tulemustest juhtkonnale kokkuvõtte.

3 Veebirakenduste testimise komponendid ja nende sisu

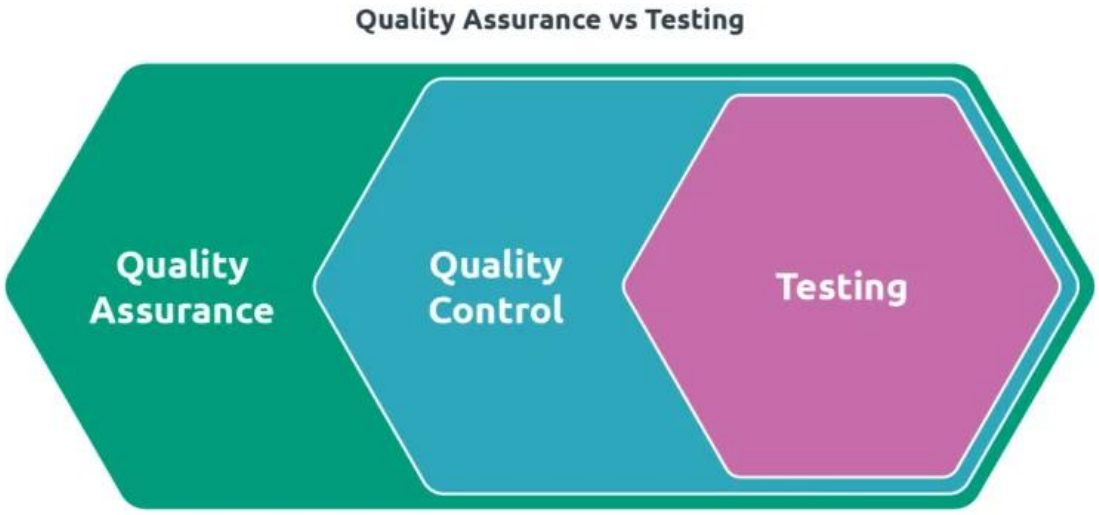
Tänasel digiajastul on veebirakendused muutunud ettevõtete ja organisatsioonide jaoks üha olulisemaks. Seetõttu on oluline tagada, et need rakendused ei ole mitte ainult funktsionaalsed ja usaldusväärsed, vaid vastavad ka nende kasutajate vajadustele. Siin tuleb mängu veebirakenduste testimine. Testimine on tarkvaraarenduse oluline aspekt ja veebirakenduste testimine ei ole erand. See peatükk annab ülevaate veebirakenduste testimisest, selle kriitilisest rollist arendusprotsessis, testimise erinevatest komponentidest ja nende sisust.

Tarkvara testimine on tarkvaraarenduse elutsükli (SDLC) lahutamatu osa (Joonis 1). See on tarkvaratoodete kontrollprotsess, mille eesmärk on tuvastada rakenduses esinevad vead, tõrked ja defektid enne selle lõppkasutajatele avaldamist. Testimine on vajalik, et tagada tarkvara ootuspärane toimimine ja huvirühmade nõuetele vastavus. [4]

Joonis 1. Tarkvaraarenduse elutsükkel.

Joonis (Joonis 1) illustreerib SDLC etappe ning sellelt on näha, et enamasti on testimine viies etapp. Sellele eelneb tarkvara arendamise faas ja järgneb kasutuselevõtu faas. Testimisel on samuti oma roll tarkvara kvaliteedi tagamises.

Kvaliteedi tagamine (QA) on protsess, mis tagab tarkvaratoodete vastavuse määratletud kvaliteedistandarditele. See hõlmab kõiki selliseid tegevusi nagu kvaliteedistandardite, protseduuride ja poliitikate loomine ja nende rakendamine. Testimine on kvaliteedikontrolli (QC) alamhulk kvaliteedi tagamise protsessis (Joonis 2). [5]



Joonis 2. *Quality assurance vs testing* [5].

QC on protsess, mille käigus kontrollitakse erinevate ülevaadete kaudu, kas tarkvaratoode vastab eelnevalt määratletud kvaliteedistandarditele. Testimine on vaid üks paljudest kvaliteedikontrolli tegevustest, mis hõlmab testjuhtumite kavandamist, nende teostamist ja defektidest teavitamist. Testimine on QC oluline osa, kuid see pole ainus tegevus, mis tagab tarkvaratoote kvaliteedi. [5]

Tarkvara testimise elutsükkel (STLC) on protsess, mis koosneb kontrollimis- ja valideerimistoimingutest, et tagada tarkvara kvaliteedieesmärkide saavutamine. STLC-mudel koosneb kuuest põhifaasist [6]:

1. Nõuete analüüs – testimisrühm analüüsib nõudeid testimise vaatenurgast, et tuvastada testitavad nõuded ja teha kindlaks automatiseerimise teostatavus.

2. Testi planeerimine – kvaliteedijuht ehk QA juht määrab testiplaani strateegia, ressursid, testimiskeskkonna, testimispiirangud ja testimise ajakava.

3. Testjuhtumi loomine – testjuhtumite ja skriptide loomine, kontrollimine ja muudetakse vastavalt eeltingimustele.

4. Testikeskkonna üles seadmine – testimiseks vajalike tarkvara- ja riistvaratingimuste määramine ja valmisoleku kontrolli teostamine.

5. Testi täitmine – tarkvara testitakse testiplaanide ja -juhtumite alusel ning vead teavitatakse ja parandatakse.

6. Testitsükli sulgemine – testi lõpetamise aruandlus ja testitulemuste kogumine, et tuvastada tulevaste testitsüklite strateegiad ja eemaldada protsessi kitsaskohad.

Käesolev töö käsitleb kõiki STLC-mudeli faase. Järgnevalt on peatükk jaotatud üheksaks alapeatükiks, alustades agiilse testimise põhimõtete ja veebirakenduste olemuse selgitamisega. Seejärel uuritakse erinevaid testimise komponente, nagu lähenemisviisid, meetodid, tehnikad, tüübid, rollid, tööriistad, raamistikud ja dokumentatsioon. Selles peatükis esitatud teavet kasutatakse neljandas peatükis testimiskomponentide põhjalikuks hindamiseks ja viiendas peatükis veebirakenduste testimisprotsessi täiustuste kavandamiseks.

3.1 Agiilne testimine ja selle põhimõtted

Agiilne testimine (*agile testing*) on tarkvara testimise praktika, mis järgib agiilse tarkvaraarenduse metoodikat, keskendudes kvaliteedi tagamisele kogu agiilse tarkvara arendusprotsessi vältel [7]. Agiilse testimise peamised põhimõtted on järgmised [7]:

1. Varajane ja pidev testimine – testimist alustada varakult ja testida pidevalt kogu arendusprotsessi vältel.
2. Kogu meeskonna lähenemine – kõik meeskonnaliikmed vastutavad toote kvaliteedi tagamise eest.
3. Sagedased tarned – töötavat tarkvara tarnitakse sageli, tavaliselt iga kahe nädala tagant.
4. Tihe koostöö – tihe koostöö meeskonnaliikmete vahel, et kõik oleksid samal arusaamal.
5. Klientide kaasamine – kliente kaasatakse tagasiside saamiseks kogu arendusprotsessi vältel.
6. Töötav tarkvara – iga iteratsiooni ajal keskendutakse kvaliteetsele tarkvarahaldusele.

7. Paindlik lähenemine – arendus on paindlik ja võimaldab igal ajal muutuvaid nõudeid.

Agiilne tarkvaraarendus on oluline, kuna see võimaldab meeskondadel läheneda arendustegevusele paindlikumalt ja kohanemisvõimelisemalt. Jaotades arenduse väiksemateks iteratiivseteks tsükliteks, saavad meeskonnad tarnida töötavat tarkvara sagedamini ning reageerida kiiremini muutuvatele nõuetele ja tagasisidele. See võib kaasa tuua kiirema turule jõudmise, parema koostöö meeskonnaliikmete vahel ja lõppkokkuvõttes parema toote.

Eleport Innovationsi tarkvaraarenduses on juba rakendatud mõned agiilsed põhimõtted, nagu sagedased tarned, töötav tarkvara ja paindlik lähenemine. Seetõttu lähtub ka käesolev lõputöö agiilsetest põhimõtetest, et täiustada veebirakenduste testimise protsessi. Rakendades veebirakenduste testimisel agiilseid põhimõtteid, saab ettevõtte testimismeeskond kiiresti ja tõhusalt tuvastada ja lahendada rakendustega seotud probleeme, mis viib kvaliteetsema ja töökindlama tooteni.

3.2 Veebirakenduse olemus ja selle testimise protsess

Veebirakendus on veebipõhine tarkvararakendus, millele pääseb juurde veebibrauseri või mobiiliseadme kaudu Interneti-ühenduse olemasolul. Selle peamine eesmärk on hõlbustada otsesuhtlust kasutajate ja ettevõtete vahel, võimaldades klientidel esitada päringuid ja täita mitmesuguseid ülesandeid. Veebirakendused on platvormist sõltumatud, mis tähendab, et neile pääseb ligi mis tahes seadmega, millel on Interneti-ühendus. Neid on erinevates vormides, alates lihtsatest kalkulaatoritest kuni keerukate e-kaubanduse platvormide ja sotsiaalmeedia veebisaitideni. [8]

Veebirakenduste testimine on oluline protsess, mis tagab rakenduse kvaliteedi, turvalisuse ja töökindluse. See hõlmab rakenduse võimalike probleemide ja vigade hindamist enne selle avaldamist. Testimisprotsess hõlmab tavaliselt kuut sammu rakenduse funktsionaalsuse, jõudluse ja turvalisuse tagamiseks: funktsionaalsuse testimine, kasutatavuse testimine, liidese testimine, ühilduvuse testimine, jõudluse testimine ja turvalisuse testimine. [9] Neid võib nimetada ka testimistüüpideks, millest on täpsemalt kirjeldatud peatükis 3.6.

Veebirakenduste testimine hõlmab erinevaid aspekte, sealhulgas funktsionaalsuse, turvalisuse, jõudluse ja kasutatavuse testimist, mida saab teha manuaalselt või automatiseeritult [10]. Järgides veebirakenduste testimise kuut sammu, saavad kvaliteedikontrolli meeskonnad tagada, et nende rakendused on turvalised, funktsionaalsed ja kasutajasõbralikud, pakkudes parimat võimalikku kasutuskogemust. Neid samme arvestatakse ka viiendas peatükis Eleport Innovationsi veebirakenduste testimisprotsessi täiustamisel.

3.3 Testimise lähenemisviis

Tarkvara testimise võib laias laastus jagada kaheks lähenemisviisiks: staatiline testimine (*static testing*) ja dünaamiline testimine (*dynamic testing*). Staatiline testimine on testimisviis, mis viiakse läbi tarkvararakenduse koodi käivitamata. Peamine eesmärk on tuvastada defektid tarkvaraarenduse elutsükli algfaasis. [11] Vigade leidmiseks testitakse tarkvara dokumentatsiooni, nõudeid, disaini ja koodi manuaalsete või automaatsete ülevaatuste ja läbivaatustega. Teise nimetusega kontrolltestimine (*verification*) aitab tuvastada puuduvaid nõudeid, disainivigu ja hooldamata koodi. [12]

Dünaamiline testimine on seevastu testimisviis, mis viiakse läbi tarkvararakenduse koodi käivitamise teel. Peamine eesmärk on leida defekte tarkvara käitusajal (*runtime*). [11] Tarkvarakoodi funktsionaalsuse testimiseks teostatakse erinevate sisendväärtustega testjuhtumeid. Nendega kontrollitakse, kas väljund on oodatud tulemusega. Tuntud ka nimetusega valideerimistestimine (*validation*), on selle viisi fookus tarkvaraarenduse nii funktsionaalsetel kui ka mittefunktsionaalsetel nõuetel. [13]

Staatiline ja dünaamiline testimine on üpris erinevad testimisviisid. Nende võrdluseks ja erinevuste paremaks mõistmiseks on järgmises tabelis (Tabel 1) välja toodud olulisemad erinevused ning lisaks lähenemisviisi peamine eelis ja puudus. Tabeli koostamiseks kasutati mitmeid informatiivseid allikad [12], [14], [15], [16].

Tabel 1. Staatilise ja dünaamilise testimise võrdlus.

Kriteerium	**Staatiline testimine**	**Dünaamiline testimine**
Definitsioon	Testimisviis, mis hõlmab koodi uurimist programmi käivitamata	Testimisviis, mis hõlmab programmi käivitamist erinevate sisenditega
Protsess	Kontrollprotsess	Valideerimisprotsess
Eesmärk	Defektide ennetamine	Defektide leidmine ja parandamine
Ajastus	Defektide tuvastamine tarkvaraarenduse elutsükli algfaasis	Vigade leidmine tarkvara tegelikus juurutamises ehk tarkvaraarenduse hilisemas etapis
Fookus	Keskendutakse dokumentatsioonile, nõuetele, disainile ja koodile	Keskendutakse tarkvara käitumisele, funktsionaalsetele ja mittefunktsionaalsetele nõuetele ja jõudlusele
Kulu	Kulud on väiksemad ja võtab üldiselt vähem aega	Kulud on suuremad ja on üldiselt ajakulukam
Keerukus	Vähem keeruline, kuna see hõlmab kontrollnimekirja ja tarkvarakoodi analüüsi	Keerulisem, kuna see hõlmab testijuhtumite ja –tulemuste kavandamist, teostamist ja analüüsimist
Testimistehnikad	Erinevad ülevaated, läbivaatused, kontrollid	Üksuse testimine, integratsiooni testimine, süsteemi testimine, vastuvõtu testimine
Eelised	Varajane vigade avastamine, madalad kulud, ei ole vaja testandmeid	Jäljendab reaalse maailma stsenaariume ja võib leida probleeme, mis staatilisel testimisel võivad puududa
Puudused	Piiratud koodi ülevaatusega, ei suuda kõiki vigu tuvastada	Võib olla aeganõudev ja kulukas, nõuab testandmeid

Tarkvaraarenduses on olulised nii staatiline kui ka dünaamiline testimine. Lõputöö keskendub aga dünaamilisele testimisele, kuna see sisaldab endas ettevõtte testimisprotsessi täiustamiseks olulisemaid aspekte, nagu funktsionaalne testimine, mittefunktsionaalne testimine ja reaalse maailma stsenaariumite jäljendamine. Järgmised alapeatükid uurivadki lähemalt dünaamilise testimisega seotud meetodeid, tehnikaid ja testimise tüüpe.

3.4 Testimismeetodid

Testimiseks on kaks peamist meetodit: manuaalne ehk käsitsi testimine ja automatiseeritud testimine. Kummalgi meetodil on oma eelised ja puudused ning otsus ühe teise asemel valida sõltub mitmest tegurist. Need tegurid hõlmavad projekti nõuded, saadaolevat eelarvet, ajapiiranguid ja saadaolevaid ressursse.

3.4.1 Manuaalne testimine

Manuaalne testimine on üks levinumaid veebirakenduse testimise meetodeid. Testija viib käsitsi läbi mitmeid testjuhtumeid automatiseerimistööriistadeta, et tuvastada vigu, tõrkeid ja probleeme. Protsess hõlmab planeerimist, testjuhtumite kavandamist, nende teostamist, tulemuste analüüsimist ja defektidest arendusmeeskonnale teatamist. [17] Eesmärk on kontrollida tarkvararakenduse funktsionaalsust ja tagada selle vastavus nõutavatele spetsifikatsioonidele ja standarditele. Manuaalne testimine on aeganõudev protsess, kuid oluline on tagada, et rakendus vastab äri- ja kasutajanõuetele. [18]

Manuaalsel testimisel on mitmeid eeliseid, nagu lihtsus, paindlikkus ja võimekus testida keerulisi ja mitmekesiseid kasutusstsenaariume. Samuti võimaldab see testijatel avastada vigu ja probleeme, mida automaattestid ei pruugi tuvastada. Puudustena on see aga aeganõudev ja kallis, nõuab kvalifitseeritud testijaid ning sõltub testijate oskustest, kogemustest ja tähelepanelikkusest, mistõttu võivad vead jääda märkamata või testimine võib olla ebatäpne. [19]

3.4.2 Automatiseeritud testimine

Automatiseeritud testimine on testimismeetod, mis kasutab käsitsi sekkumise asemel testjuhtumite teostamiseks automatiseerimistööriistu. See hõlmab skriptide kirjutamist ja tarkvaratööriistade kasutamist testimise, andmete loomise ja testitulemuste analüüsi automatiseerimiseks. [20] Automatiseeritud testimise peamine eesmärk on suurendada testi katvust, lühendada testimisaega ja parandada testimise täpsust. Automatiseeritud testimist on kõige mõistlikum kasutada nende testide tegemiseks, mida peab korduvalt tegema ning on raske manuaalselt teha. [21]

Automatiseeritud testimisel on mitmeid eeliseid, nagu testimise katvuse parandamine ja sõltuvuse vähendamine testijatest. See pakub ööpäevaringset katvust ja nõuab vähem ressursse kui manuaalne testimine. Selle rakendamine võib aga olla kulukas, ebamugav

ja koormav ning vajada täiendavalt koolitatud testijaid. Lisaks ei pruugi see sobida keerukate stsenaariumide või ainulaadsete ärinõuete testimiseks. Vaatamata oma eelistele eemaldab automatiseeritud testimine ainult testimise mehaanilise teostamise aga nõuab siiski, et testijad looks testjuhtumid. [22]

3.4.3 Manuaalse ja automatiseeritud testimise võrdlus

Manuaalne ja automatiseeritud testimine on tarkvara testimisel kaks levinud testimismeetodit, mida kasutatakse erinevates stsenaariumides, mis põhinevad projekti nõuetel, eelarvel ja ajapiirangutel. Järgmises võrdlustabelis (Tabel 2) on toodud manuaalse ja automatiseeritud testimise erinevused, sealhulgas nende eelised, puudused ja testimistehnikad. Tabeli koostamiseks on kasutatud kolme allikat, et välja tuua kõige olulisemad võrdluskriteeriumid [23], [24], [25].

Tabel 2. Manuaalse ja automatiseeritud testimise võrdlus.

Kriteerium	**Manuaalne testimine**	**Automatiseeritud testimine**
Definitsioon	Meetod veebirakenduse käsitsi testimiseks automatiseerimistööriistu kasutamata	Testimine teostatud automatiseerimistööriistade ja skriptide abil
Kiirus	Aeglasem kui automaattestimine	Kiirem kui manuaaltestimine
Paindlikkus	Võimaldab testimise stsenaariumide puhul paindlikkust	Piiratud eelmääratletud testjuhtumite ja skriptidega
Kulu	Odavam, kuna see nõuab ainult testijaid	Kallim, kuna vajab lisaks testijale ka tööriista ostmist
Täpsus	Tulemused võivad olla ebatäpsed inimlike vigade tõttu	Annab usaldusväärsemad ja ühtlasemad tulemused tänu arvutipõhisele testimisele
Hooldus	Tuleb korrata iga tarkvaramuudatuse korral	Hooldus on lihtsam, kuna teste saab hõlpsasti uuesti läbi viia
Testjuhtumid	Testjuhtumite käsitsi kavandamine ja teostamine	Testjuhtumeid saab kujundada üks kord ja teostada mitu korda
Testimistehnikad	Sobib vastuvõtu testimiseks ja kasutatavuse testimiseks	Sobib regressiooni- ja koormustestimiseks
Testija oskused	Vajalik testimisoskus ja kogemus	Vajalikud tehnilised teadmised ja programmeerimisalased teadmised

Kriteerium	Manuaalne testimine	Automatiseeritud testimine
Eelised	Lihtsus, paindlikkus, võime testida keerulisi ja mitmekesiseid stsenaariume, võime leida automaattestimise käigus vahele jäänud defekte	Kiirus, töökindlus, kuluefektiivsus, inimlike vigade vähendamine, täpsus
Puudused	Aeganõudev, kallis, oleneb testijate oskustest, vigade puudumise võimalus ja ebatäpsus	Piiratud keerukate ja erinevate stsenaariumide testimisel

Üldiselt võib automatiseeritud testimine olla väärtuslik meetod tarkvara testimisel. Kuigi automatiseeritud testimine võib pakkuda tõhusust ja skaleeritavust, ei tohiks see manuaalset testimist täielikult asendada. Nii manuaalse kui ka automatiseeritud testimismeetodite kombineerimine võib aidata saavutada tasakaalu põhjaliku testimise ja tõhusa ressursside kasutamise vahel.

3.5 Testimise tehnikad

Dünaamilise testimise alla kuulub kastitestimine, mis jaguneb kolmeks testimise tehnikaks: valge kasti testimine, musta kasti testimine ja halli kasti testimine [26]. Valge kasti testimine (*white box testing*) on testimistehnika, mis keskendub tarkvararakenduse sisemise struktuuri, disaini ja koodi testimisele. Eesmärk on rakenduse lähtekoodi põhjalikult testida. Testijal on juurdepääs rakenduse lähtekoodile, mis võimaldab testida rakenduse üksikuid komponente, hõlbustades peidetud vigade leidmist ja testjuhtumite automatiseerimist. Valge kasti testimine võib aga olla keeruline, kulukas ja aeganõudev. See nõuab professionaalseid testijaid, kellel on üksikasjalik arusaam programmeerimisest. [27]

Musta kasti testimine (*black box testing*) on testimistehnika, mis keskendub süsteemi testimisele kasutaja vaatenurgast, omamata juurdepääsu süsteemi sisemisele tööle. Testija annab sisendi ja jälgib testitava süsteemi genereeritud väljundit. Tehnika aitab tuvastada kasutatavuse probleeme ja seda, kuidas süsteem reageerib kasutaja oodatud ja ootamatule sisendile. Peamise eelisena tagab see rakenduse vastavuse kasutaja ootustele ja nõuetele ning aitab tuvastada süsteemi defekte ja vigu. Siiski ei anna see teavet rakenduse sisemise töö kohta ega pruugi olla võimeline tuvastama keerulisi vigu. [28] Musta kasti testimine

jaguneb kolme tüüpi: funktsionaalne testimine, mittefunktsionaalne testimine ja regressiooni testimine [28], millest on lähemalt räägitud 3.6 alapeatükis.

Halli kasti testimine (*grey box testing*) on testimistehnika, mis ühendab valge ja musta kasti testimise elemente. See hõlmab testijate teadmisi testitava tarkvararakenduse sisemisest tööst, kuid mitte täielikke teadmisi. Halli kasti testimine võib pakkuda mitmeid eeliseid, nagu kasutaja vaatenurgaga arvestamine, selgete testimiseesmärkide omamine, arendajatele rohkema aja andmine vigade parandamiseks ja võimalike konfliktide lahendamine. Siiski on sellel ka mõned puudused, nagu raskused defektide omistamisel ja algpõhjuste tuvastamisel, piiratud võime kooditeid läbida, suutmatus testida algoritme ja keeruline testjuhtumi ülesehitus. [29]

Igal testimise tehnikal on omad eelised ja puudused ning õige lähenemine sõltub testimisprotsessi konkreetsetest eesmärkidest. Järgmine võrdlustabel (Tabel 3) on koostatud kolme allika põhjal [30], [31], [32] ning võtab kokku kolme tüüpi kasti testimise erinevused.

Tabel 3. Valge kasti, musta kasti ja halli kasti testimise võrdlus.

Kriteerium	**Valge kasti testimine**	**Musta kasti testimine**	**Halli kasti testimine**
Definitsioon	Testimine sisemise struktuuri täielike teadmistega	Testimine sisemise struktuuri tundmiseta	Testimine piiratud teadmistega sisemise struktuuri kohta
Lähenemine	Struktuuri testimine	Käitumise testimine	Käitumis- ja struktuuritestide kombinatsioon
Eesmärk	Sisemise koodiloogika kontrollimine	Funktsionaalsete nõuete kontrollimine	Nii funktsionaalsete nõuete kui ka sisemise koodiloogika kontrollimine
Testija teadmised	Koodi täielik tundmine	Koodist teadmised puuduvad	Piiratud teadmised koodist, juurdepääs rakenduse dokumentidele ja nõuetele
Testjuhtumid	Põhineb sisemisel struktuuril ja koodil	Funktsionaalsete nõuete alusel	Lähtudes nii funktsionaalsetest nõuetest kui ka sisemisest struktuurist

Kriteerium	Valge kasti testimine	Musta kasti testimine	Halli kasti testimine
Eelised	Põhjalik katvus, tuvastab varjatud vead	Keskendub kasutaja nõudmistele, teadmisi pole vaja	Arvestab kasutaja vaatenurka, tõhusad veaparandused
Puudused	Sõltub arendaja teadmistest, aeganõudev	Koodile juurdepääs puudub, vähem tõhus vea leidmisel	Piiratud juurdepääs koodile, keeruline testjuhtumi ülesehitus

Esitatud tabel tõstab esile erinevused kolme tehnika vahel, sealhulgas lähenemine, eesmärk ja testija teadmiste tase rakenduse sisemise toimimise kohta. Tabelis on märgitud ka iga tehnika eelised ja puudused. Üldiselt sõltub testimistehnika valik projekti spetsiifilistest nõuetest ja piirangutest ning sageli kasutatakse soovitud kvaliteeditaseme saavutamiseks tehnikate kombinatsiooni.

3.6 Dünaamilise testimise tüübid

Dünaamiline testimine hõlmab mitmesuguseid testimistüüpe, millest igaühel on oma ainulaadne eesmärk ja eelised. Dünaamilise testimise kaks peamist tüüpi on funktsionaalne testimine ja mittefunktsionaalne testimine. Kasutatava tüübi valik sõltub projekti konkreetsetest vajadustest. Teadliku otsuse tegemiseks on oluline mõista igat tüüpi testimise sisu ning nende eeliseid ja puudusi.

3.6.1 Funktsionaalne testimine

Funktsionaalne testimine (*functional testing*) on tarkvara testimise tüüp, mille käigus kontrollitakse, kas tarkvararakendus täidab kõiki funktsioone, mida on ette nähtud. Testimine hõlmab rakenduse üksikute funktsioonide või omaduste testimist, et tagada nende korrektne toimimine ja oodatud tulemuste saavutamine. [33] Funktsionaalse testimise peamised neli taset on üksuse testimine, integratsiooni testimine, süsteemi testimine ja vastuvõtu testimine [34].

Üksuse testimine (*unit testing*) testib rakenduse üksikuid üksusi või komponente eraldi, et tuvastada ja lahendada probleemid arendustsükli alguses. Integratsiooni testimine (*integration testing*) seevastu testib rakenduse erinevate üksuste või komponentide vahelist koostoimet, et tagada nende ootuspärane koostöö. Süsteemi testimine (*system*

testing) testib aga kogu süsteemi tervikuna, et tagada selle ootuspärane toimimine erinevates keskkondades ja stsenaariumides. [34]

Vastuvõtu testimine (*acceptance testing*) testib rakendust kasutaja vaatenurgast, et tagada selle vastavus nende nõuetele ja ootustele [34]. Selle saab jagada kaheks vastuvõtutestiks: alfatestimine ja beetatestimine. Alfatestimise (*alpha testing*) käigus hindab lõppkasutajate rühm või sõltumatu testimismeeskond tarkvararakendust enne selle avalikkusele avaldamist [35]. Beetatestimine (*beta testing*) on tarkvararakenduse viimane testimise etapp enne selle ametlikku väljalaskmist, mille käigus piiratud rühm lõppkasutajaid testib tarkvara toodangukeskkonnas, et tuvastada allesjäänud vead [35].

3.6.2 Mittefunktsionaalne testimine

Mittefunktsionaalne testimine (*non-functional testing*) on tarkvara testimise tüüp, mis keskendub rakenduse mittefunktsionaalsete aspektide testimisele, nagu jõudlus, kasutusmugavus, ühilduvus ja turvalisus. Testimine on seotud süsteemi kvaliteediatribuutide testimisega. [36] Mittefunktsionaalseid testitüüpe on mitmeid, kuid peamised neli on jõudluse testimine, kasutatavuse testimine, turvatestimine ja ühilduvuse testimine [37].

Jõudluse testimine (*performance testing*) mõõdab tarkvarasüsteemi reageerimisvõimet, skaleeritavust ja stabiilsust erinevatel töökoormustel ja tingimustes. Kasutatavuse testimine (*usability testing*) hindab aga tarkvarasüsteemi kasutajasõbralikkust, ligipääsetavust ja rahulolu. Turvatestimine (*security testing*) tuvastab ja maandab tarkvarasüsteemi võimalikke turvariske ja haavatavusi ning ühilduvuse testimine (*compatibility testing*) tagab veebirakenduse ühilduvuse erinevate operatsioonisüsteemide, veebibrauserite ja seadmetega. [37]

Lisaks eelpool toodule võib mittefunktsionaalse testimise alla tuua ka regressioonitestimise, liidese testimise, andmebaasi testimise ja API testimise. Regressioonitestimine (*regression testing*) testib veebirakendust pärast muudatuste tegemist, et ei tekiks uusi probleeme, säilitades aja jooksul rakenduse kvaliteedi ja stabiilsuse [38]. Liidese testimine (*interface testing*) on tarkvara testimise tüüp, mis kontrollib erinevate komponentide või süsteemide vaheliste interaktsioonide õiget toimimist [39].

Andmebaasi testimine (*database testing*) on protsess, mille käigus kontrollitakse andmebaasis salvestatud andmete terviklikkust ja järjepidevust ning see viiakse läbi testimiskeskkonnas [40]. API testimine on API eeldatava funktsionaalsuse valideerimise protsess kas manuaalselt või automatiseeritud testimise kaudu, et probleemid arendusprotsessis varem tuvastada ja lahendada [41].

3.6.3 Funktsionaalse ja mittefunktsionaalse testimise võrdlus

Nii funktsionaalsel kui ka mittefunktsionaalse testimisel on oma eelised ja puudused ning testimistüübi valik sõltub testimisprotsessi konkreetsetest eesmärkidest. Nende erinevused võtab kokku järgmine võrdlustabel (Tabel 4), mis on koostatud mitme allika põhjal [42], [43].

Tabel 4. Funktsionaalse ja mittefunktsionaalse testimise võrdlus.

Kriteerium	**Funktsionaalne testimine**	**Mittefunktsionaalne testimine**
Definitsioon	Kontrollib, kas süsteem töötab ettenähtud viisil ja vastab kasutaja nõuetele	Hindab süsteemi omadusi, nagu kasutatavus, jõudlus, turvalisus ja ühilduvus
Fookus	Testib tarkvara funktsioone ja omadusi	Testib tarkvara atribuute, mida ei saa ebaõnnestumise kriteeriumide alusel kergesti mõõta
Testimise tüübid	Üksuse testimine, integratsiooni testimine, süsteemi testimine, vastuvõtu testimine	Kasutatavuse testimine, jõudluse testimine, turvatestimine, ühilduvuse testimine
Testimise kriteeriumid	Läbitud/ebaõnnestunud, mis põhineb funktsionaalsete nõuete täitmisel	Mõõdikutepõhine hindamine võrreldes mittefunktsionaalsete nõuetega
Tööriistad ja tehnikad	Testjuhtumid, automatiseerimise tööriistad	Jõudlustööriistad, koormustesti tööriistad, turvatööriistad
Eelised	Tagab, et süsteem vastab kasutaja vajadustele, tuvastab defektid arendustsükli varajases staadiumis ja tagab kvaliteedi	Tuvastab mittefunktsionaalsete nõuetega seotud defektid, parandab kasutuskogemust ja tagab kvaliteedi
Puudused	Ei pruugi tuvastada kasutatavuse, turvalisuse või jõudlusega seotud probleeme	Ei pruugi tuvastada funktsionaalseid defekte, võib olla raske automatiseerida, aeganõudev ja kulukas

Tabel 4 annab võrdluse kõige olulisematest kriteeriumidest, mis eristavad funktsionaalset ja mittefunktsionaalset testimist, nagu nende fookus, testimise tüübid, peamised eelised ja puudused. Nende erinevuste põhjalik mõistmine on oluline, kui otsustada, millist tüüpi testimist konkreetse projekti jaoks kasutada. Kuigi funktsionaalne testimine tagab, et rakendus vastab ettenähtud eesmärgile, siis mittefunktsionaalne testimine tagab selle jõudluse, turvalisuse ja muude mittefunktsionaalsete nõuete täitmise.

3.7 Rollid ja vastutusalad

Tarkvaraarendusprojekti testimismeeskond vastutab selle eest, et tarkvara toimib korralikult, on töökindel ja vastab klientide vajadustele. Testimismeeskond koosneb mitmest rollist, millest igaühel on oma spetsiifilised kohustused ning panus tarkvara süsteemi kvaliteedi, usaldusväärsuse ja turvalisuse tagamisel. Mõned levinud rollid tarkvara testimisprotsessis on järgmised [44], [45], [46], [47]:

1. QA juht (*QA manager*) – vastutab kogu testimismeeskonna juhtimise eest ning kogu testimisprotsessi jälgimise eest alates planeerimisest, testimise õigeaegsest ja eelarve piires läbiviimisest kuni aruandluse ja analüüsini. [44], [45]

2. Tiimijuht (*team lead*) – vastutab kindla testijate tiimi juhtimise ja juhendamise, testimiste plaanipärase läbiviimise ja tulemuste raporteerimise eest QA juhile. [44], [46]

3. Testiarhitekt (*test architect*) – vastutab üldise testimisarhitektuuri, -strateegia ja -raamistiku kavandamise, testimisplaanide loomise ja kogu tarkvararakenduse teststsenaariumide kujundamise eest [45].

4. Automaattestimise insener (*automated test engineer*) – vastutab automatiseeritud testimisraamistike, -tööriistade ja testjuhtumite arendamise, hooldamise, läbiviimise ning tulemuste ja defektide raporteerimise eest. [45], [46]

5. Manuaaltestija (*manual tester*) – vastutab käsitsi testimisülesannete täitmise, testjuhtumite läbiviimise ja tulemuste dokumenteerimise eest [46].

6. Beetatestija tugi (*beta tester support*) – vastutab beetatestijatele toe pakkumise eest tarkvara seadistamisel ja kasutamisel ning tagab, et neil on testimiseks ja tagasiside andmiseks kõik vajalikud ressursid olemas [47].

7. Beetatestija (*beta tester*) – vastutab toodangukeskkonnas tarkvara testimise, probleemide raporteerimise ja tagasiside andmise eest [47].

Tarkvara testimisprotsessis on erinevatel rollidel vastastikku täiendavad vastutusalad, mis aitavad kaasa tarkvarasüsteemi kvaliteedi, usaldusväärsuse ja turvalisuse tagamisele. Rollide valik ja jaotus sõltuvad projekti ja testimisprotsessi konkreetsetest vajadustest, eesmärkidest ja piirangutest.

3.8 Tööriistad ja raamistikud

Veebirakenduste testimine võib olla keeruline, eriti kui tegemist on vigade, defektide ja turvaaukude tuvastamise ja ennetamisega. Õnneks võivad tööriistad ja raamistikud testimise protsessi lihtsamaks ja tõhusamaks muuta.

Testimistööriistad on tarkvararakendused, mis loovad, käivitavad ja jälgivad testjuhtumeid, et hinnata tarkvaraprogrammi ja süsteemi kvaliteeti. Testimise automatiseerimise tööriistad võimaldavad testikomplekti kiiret ja täpset lõpetamist, vähendades manuaalset tööd, aega ja kulusid. [48]

Testimisraamistik on tööriistade, tavade ja juhiste kogum, mis toetab testjuhtumite loomist ja täitmist, sealhulgas testiandmete töötlemist ja tulemuste haldamist. Kuigi raamistiku kasutamine pole vajalik, võib see suurendada tõhusust ja pakkuda täiendavaid eeliseid. [49]

Veebirakenduste testimise tööriistad ja raamistikud automatiseerivad veebirakenduste testimist, et tagada põhjalik ja tõhus testimine nõuetele vastavuse ja defektideta rakenduste jaoks. Saadaval on palju tööriistu ja raamistikke, kuid uuring keskendub nende võrdlemisele, mis suudavad Jira tarkvaraga integreeruda ja võimaldavad testimisprotsessis kiiremat rakendamist.

Jira Software projektihaldustarkvara saab kasutada testide haldamiseks ja Confluence dokumendikeskkonda testitulemuste aruandluseks, välistades vajaduse eraldi

testihaldustööriista järele. Automaattestide jaoks on aga vajalikud tööriistad ja raamistikud. Siin on mõned populaarsemad veebirakenduste testimise tööriistad, mis on koostatud erinevatest allikatest populaarsuse alusel [50], [51], [52]:

1. Katalon Studio – kõikehõlmav automatiseerimise testimise tööriist, mis toetab veebi-, mobiili-, API- ja töölauarakendusi [50], [51], [52]. Hind ettevõttele on 354 dollarit kuus [53].

2. Tricentis – pidev testimisplatvorm, mis pakub ettevõtte rakendustele täielikke testimislahendusi [50], [51], [52]. Ettevõttele on hind kohandatud, algab alates 50 dollarist kuus kasutaja kohta [54].

3. testRigor – pilvepõhine testimisplatvorm, mis kasutab AI-d testjuhtumite loomise ja täitmise optimeerimiseks ja kiirendamiseks [50], [51], [52]. Ettevõttele on hind kohandatud [55].

4. BugBug – veebirakenduste testimise tööriist, mis tuvastab turvanõrkused, funktsionaalsed probleemid ja jõudlusprobleemid [50], [51]. Hind on kohandatav, alates 49 dollarist kuus [56].

5. Mabl – AI-toega otsast lõpuni testimise tööriist, mis parandab testi katvust ja vähendab hooldust [50], [51]. Ettevõttele on hind kohandatav [57].

6. Selenium – avatud lähtekoodiga automatiseerimise testimise raamistik, mis toetab veebibrausereid ja mitut programmeerimiskeelt [50], [52]. Avatud lähtekoodiga, seega hind puudub [58].

7. Testpad – testihaldustööriist, mis lihtsustab testide planeerimist, teostamist ja aruandlust [50], [52]. Hind on kohandatav, alates 119 dollarist kuus [59].

8. BrowserStack – pilvepõhine testimisplatvorm, mis võimaldab veebirakendusi brauseri- ja seadmeüleselt testida [51], [52]. Ettevõttele on hind kohandatav [60].

Pärast mitme allika uurimist tuvastati veebirakenduste testimises kõige sagedamini mainitud testimisraamistikud [61], [62], [63]. Järgmises loendis on parimad raamistikud:

1. Cypress – JavaScripti täielik testimise raamistik veebirakenduste jaoks [61], [62], [63]. Ettevõttele on hind kohandatav [64].

2. Playwright – Node.js teek veebibrauserite automatiseerimiseks uusimatel veebistandarditel [61], [62]. Avatud lähtekoodiga, seega hind puudub [65].

3. Robot Framework – Pythonil põhinev märksõnapõhine testimise automatiseerimise raamistik [61], [63]. Avatud lähtekoodiga, seega hind puudub [66].

4. WebdriverIO – Node.js-i järgmise põlvkonna brauseri automatiseerimise testimise raamistik [62], [63]. Avatud lähtekoodiga, seega hind puudub [67].

5. Selenium WebDriver – laialdaselt kasutatav avatud lähtekoodiga testimise automatiseerimise raamistik veebirakenduste jaoks [61]. Avatud lähtekoodiga, seega hind puudub [58].

Testimistööriistad ja -raamistikud on kasulikud veebirakenduste testimise korraldamiseks. Nende kasutamine aitab kaasa efektiivsemale tarkvara kvaliteedi tagamisele. Vajalik on aga õige tööriista ja raamistiku valimine, milleks tuleb lähtuda projekti spetsiifilistest või ettevõtte üldistest tarkvara nõuetest.

3.9 Dokumentatsioon

Dokumentatsioon mängib veebirakenduste testimisel olulist osa, kuna see tagab meeskonnaliikmete ja sidusrühmade ühise arusaamise testimisest, hõlbustades suhtlust erinevate osakondade või osapoolte vahel. Lisaks võimaldab dokumentatsioon esitada varasemaid testitulemusi tagasivaadete või kokkuvõtete jaoks. Tarkvara testimise dokumendid võib jagada kahte kategooriasse: ettevõtte sisesed ja välised dokumendid.

Sisemised dokumendid on eriti olulised arendus- ja testimismeeskonna liikmete jaoks, kuna need sisaldavad tehnilise taseme testimise eesmärke, meetodeid ja tulemusi. Kaheksa peamist sisedokumenti hõlmab esimesena testimispoliitikat, mis kirjeldab ettevõtte testimise eeskirju. Seejärel toob testistrateegia välja erinevatel projektitasanditel testimise põhiaspektid ning testiplaan kirjeldab üksikasjalikult testimisprojekti põhiaspekte, mida jagatakse kogu meeskonnaga. [68]

Testistsenaarium määratleb toote konkreetse osa täieliku testimise protsessi, samas kui testiandmete dokument kirjeldab reaalse rakenduse kasutamise simuleerimiseks vajalikke

testandmeid. Testjuhtum annab teavet selle kohta, kuidas testida konkreetset tarkvara osa, mida teostab manuaaltestija. Testilogi salvestab teatud tüüpi testi testjuhtumid ja tulemused, samas kui jälgitavuse maatriks (*traceability matrix*) on tabel, mis jälgib testjuhtumite nõudeid ja täitmise vastavust kogu tarkvaraarenduse elutsükli jooksul. [68]

Välised tarkvaratestimise dokumendid esitavad visuaalselt lõpptulemusi ning neid esitatakse sageli klientidele ja kasutajatele. Need aitavad toodet täiustada ning tagavad, et arendajad ja kliendid on ühel arusaamal. Väliseid dokumente on peamiselt kolm. Esimene on veaaruanne, mis kirjeldab rakenduse konkreetset viga. Teine dokument on koondaruanne, mis annab ülevaate testitsükli tulemustest. Kolmas on kasutaja vastuvõtuaruanne, mis sisaldab testimise tulemusi enne tarnijale esitamist vastavuse kontrollimiseks. [69]

Üldiselt on testidokumentide eesmärk muuta testimine kõigile asjaosalistele lihtsamaks ning tagada, et testimine viiakse läbi põhjalikult ja tõhusalt [70]. Testidokumentide koosseis võib erineda vastavalt ettevõtte ja projekti vajadustele. Üldjuhul luuakse enne sisemised dokumendid ja seejärel välised. [71]

4 Sobivuse hindamise analüüs

Käesoleva peatüki eesmärk on analüüsida testimisprotsessi täiustuste jaoks sobivate testitüüpide, rollide, tööriistade ja dokumentide nõudeid ning hinnata nende integreerimise võimalust ettevõtte testimisprotsessi. Eleport Innovationsi tehnoloogiajuhi hinnangu kohaselt saab testimisprotsessi jagada neljaks etapiks: arendustestimine, teenuse testimine, piirkondlik testimine ja lõppkasutaja testimine. Iga etapi eesmärk on saavutada konkreetsed eesmärgid, nagu automatiseeritud testimine, parem osakonna struktuur ja tõhus beetatestimise protsess.

4.1 Nõuete analüüs

Eleport Innovationsi veebirakenduste testimisprotsessi täiendav lahendus peab täitma mitmeid nõudeid. Need hõlmavad kaalutlusi, mis on seotud olemasolevate tarkvaraintegratsiooni testimisvahenditega, aga ka inim- ja rahaliste ressursside piiranguid, võttes arvesse meeskonna väikest ja paindlikku olemust.

- **Testimistüüpide nõuded.** Eduka testimisprotsessi tagamiseks tuleb arvestada iga etapi nõuetega. Esimene etapp, arendustestimine, nõuab arenduskeskkonnas testimist, mille viivad läbi arendajad. Teine etapp, teenuse testimine, toimub reaalsete andmetega testkeskkonnas ning järgida tuleb vajalikke oskusi ja testimise järjekorda. Kolmandas ja neljandas etapis viivad testimist läbi vastavalt Eleport OÜ piirkondlikud töötajad ja testkliendid ning arvestada tuleb teoreetiliste testimisteadmiste puudumisega.
- **Rollide ja vastutusalade nõuded.** Testimisprotsessi jaoks sobivate rollide kujundamiseks tuleb arvestada ettevõtte suurust ja piiratud inimressursse. Igal rollil on oma spetsiifilised vastutused ja kohustused. Üks ametikoht võib täita mitut rolli, kuid ametikoha täitjal peab olema iga rolli täitmiseks vajalikud oskused ja teadmised. Rollide kombineerimisel tuleb olla ettevaatlik, et vältida võimalikke konflikte oskuste ja ülesannete vahel.
- **Tööriistade ja raamistike nõuded.** Sobivate testimisvahendite ja -raamistike valimiseks tuleb arvestada mitmete nõuetega. Esiteks peavad tööriistad ja raamistikud olema hõlpsasti integreeritavad praegusesse süsteemi ning olema tuttavad meeskonnaliikmetele, kuna eeliseks võib olla eelnev töökogemus

tööriistaga. Lisaks peavad valitud tööriistad ja raamistikud ühilduma Jira Software tarkvaraga ning toetama veebirakenduste arendamise programmeerimiskeeli, nagu C#. Lisaks peavad nad tõhususe ja täpsuse parandamiseks võimaldama automatiseeritud testimist.

- **Dokumentatsiooni nõuded.** Tõhus dokumentatsioon on eduka testimisprotsessi põhinõue. See peab olema täpne, üksikasjalik ja kergesti mõistetav. Dokumentatsioon peab sisaldama kõigi testjuhtumite ja nende tulemuste protokolle, testimisprotsessi kokkuvõtteid ja aruandeid ning tulemuste analüüsimise võimalust. Ebaõnnestunud juhtumitest tuleb teavitada arendusmeeskonda.
- **Eelarve nõuded.** Testimisprotsessiga kaasnevad erinevad kulud, sealhulgas personalikulud nagu palgad, testimisvahendite maksumus ja muud testimiseks vajalike ressurssidega seotud kulud. Kuna Eleport Innovations on väike agiilne meeskond, ei ole lähiajal plaanis täismahus testimismeeskonna loomiseks lisatöötajaid palgata. Seetõttu on oluline minimeerida täiendavaid palgakulusid, tagades samal ajal kvaliteedikontrolli. Testimisvahendite ja -raamistike valikul tuleb keskenduda kõige kuluefektiivsema ja nõutavatele kvaliteedistandarditele vastava variandi leidmisele. Eelarve optimeerimiseks on oluline hinnata kõiki kulusid ja hinnata olemasolevaid ressursse.
- **Ajakava nõuded.** Ettevõtte agiilse arendusprotsessi toetamiseks tuleb testimise ajakava koostada nii, et see on kiire ja tõhus, mõjutades minimaalselt üldist arenduse ajakava. Soovitatav on kogu testimisprotsess, sealhulgas kõik neli etappi, läbida maksimaalselt kahe nädala jooksul. Selle saavutamiseks võib osutuda vajalikuks testimistegevuste tähtsuse järjekorda seadmine kriitilisuse alusel ning testimisprotsessi sujuvamaks muutmiseks kasutada automatiseeritud testimise tööriistu ja raamistikke. Lisaks tuleb luua regulaarne suhtlus ja koostöö arendajate ja testijate vahel, et tagada testimise käigus tekkivate probleemide õigeaegne tuvastamine ja lahendamine.

Veebirakenduste testimisprotsessi täiustamisel on oluline arvestada kõigi loetletud nõuetega, sealhulgas tehniliste ja töötajatega seotud piirangutega. Eelarve ja ajakava nõuded on aga üliolulise tähtsusega, kuna need kujutavad endast piiratud ressursse. Oluline on hoolikalt planeerida ja eraldada ressursse, et viia ellu tõhusam ja tulemuslikum testimisprotsess lõpptoote kvaliteedis järeleandmisi tegemata.

4.2 Hindamisprotsess

Hindamisprotsess mängib otsustavat rolli täiustuskomponentide võimalike eeliste ja riskide kindlaksmääramisel ning teistest valikutest kõige sobivamate valimisel. Protsessi käigus võetakse arvesse mitmeid tegureid, sealhulgas tõhusust, tulemuslikkust, ühilduvust olemasoleva süsteemiga ja kuluefektiivsust. Need tegurid aitavad välja selgitada kõige sobivamad komponendid, mis vastavad projekti eesmärkidele ja nõuetele. Hindamisprotsessi järeldused koos kavandatud täiustustega asuvad viiendas peatükis.

4.2.1 Testimistüüpide hindamine

Arendustestimise etapis keskendutakse vigade ja defektide võimalikult varajasele tuvastamisele. Etapi jaoks sobivad testimistüübid on üksuse testimine, integratsiooni testimine, andmebaasi testimine ja API testimine. Võimalikud eelised on vigade varajane avastamine ning nende parandamise kulude ja aja vähenemine. On oht, et mõningaid defekte võidakse avastada alles testimise hilisemates etappides, mis võib põhjustada viivitusi ja suurendada kulusid.

Teenuse testimise jaoks sobivad andmebaasi testimine, API testimine, liidese testimine, süsteemi testimine, jõudluse testimine, turvatestimine, ühilduvuse testimine ja regressioonitestimine. Võimalikud eelised on süsteemitaseme probleemide tuvastamine ja süsteemi jõudluse ja turbenõuete täitmise tagamine. On oht, et need testimistüübid võivad olla aeganõudvad ja kulukad, mis võib mõjutada projekti ajakava ja eelarvet.

Piirkondliku testimisetapi jaoks sobivad vastuvõtu testimine (sh alfatestimine) ja liidese testimine. Võimalikud eelised on kasutajate nõuete ja probleemide tuvastamine, mis võivad tekkida erinevate süsteemide integreerimisel. On risk, et testimise ulatus võib olla piiratud ja mõned probleemid võidakse avastada alles testimise hilisemates etappides.

Kasutatavuse testimine ja vastuvõtu testimine (sh beetatestimine) sobivad viimaseks ehk lõppkasutaja testimisetapiks. Võimalikud eelised on toote kasutatavuse ja funktsionaalsuse valideerimine ning probleemide tuvastamine, mis võivad mõjutada kasutajate rahulolu. On risk, et testimine ei pruugi hõlmata kõiki võimalikke stsenaariume ja probleeme, mis võivad tekkida reaalses kasutuses.

Valides igaks etapiks sobivad testimistüübid, saab tagada, et testimisprotsess on tõhus ja tulemuslik kogu arenduse elutsükli jooksul tekkida võivate probleemide tuvastamisel ja

lahendamisel. Oluline on tasakaalustada iga testimistüübi eelised ja riskid ning võtta sobivate testimistüüpide valimisel arvesse projekti ulatust, ajakava ja eelarvet.

4.2.2 Rollide ja vastutusalade hindamine

Iga testimisprotsessi edu sõltub suuresti rollide ja vastutusalade õigest määratlemisest ja määramisest. Tõhusaks testimisprotsessiks vajalike rollide ja vastutuse hindamisel on oluline arvestada iga rolliga kaasnevaid võimalikke eeliseid ja riske.

QA juhi roll võib tagada testimisprotsessi kvaliteedi, kuid ei pruugi olla väikse testimismeeskonna jaoks vajalik. Selle rolli kombineerimine tiimijuhi rolliga võib optimeerida tõhusust kvaliteeti ohverdamata. Rollide kombineerimisel on oht, et see võib tekitada huvide konflikte või viia kohustuste ülekoormamiseni.

Tiimijuhi roll on testimisprotsessis üliolu line, eriti väikses meeskonnas. Ta annab meeskonnale juhiseid ning tagab testimisprotsessi tõhusa läbiviimise. Tiimijuhi puudumisel on oht, et testimisprotsessil võib puududa suund ja fookus.

Testarhitekti roll on oluline testjuhtumite kirjutamisel ja testitulemuste analüüsimisel. Väikses meeskonnas saab seda rolli aga kombineerida beetatestija tugirolliga, et vähendada rollide ja kohustuste hulka. Nende rollide kombineerimisel on oht, et see võib kaasa tuua keskendumise ja asjatundlikkuse puudumise mõlemas valdkonnas.

Automaattestimise inseneri roll on testimisprotsessi jaoks oluline ja selle saab sõltuvalt sooritatava testi tüübist jagada konkreetseteks rollideks. See võib tagada, et iga testitüüp viiakse läbi tõhusalt ja tulemuslikult. Konkreetsete rollide puudumisel automatiseeritud testimisel on oht, et see võib põhjustada asjatundlikkuse puudumise konkreetsetes valdkondades.

Manuaaltestija roll on testimisprotsessis oluline ning seda saab testimise kolmandas etapis lisatasuta määrata Eleport OÜ piirkondlikele töötajatele. See võib optimeerida tõhusust ja vähendada kulusid. Selle rolli määramisel piirkondlikele töötajatele on oht, et see võib kaasa tuua teadmiste puudumise või suhtlusbarjääri.

Beetatestijate tugi ja beetatestija rollid on testimisprotsessis olulised ning neid saab testimise neljandas etapis lisatasuta testklientidele määrata. See võib tagada, et testimisprotsess viiakse läbi toodangukeskkonnas ja anda väärtuslikku tagasisidet. Nende

rollide testklientidele määramisel on oht, et neil ei pruugi olla väärtusliku tagasiside andmiseks vajalikke teadmisi või ressursse.

4.2.3 Tööriistade ja raamistike hindamine

Testimistööriistade ja -raamistike hindamisel on oluline arvestada nende võimalike eeliste ja riskidega. Katalon Studio, Tricentis, testRigor, BugBug, Mabl, Selenium, Testpad ja BrowserStack on kõik tuntud oma integreerimise lihtsuse ja kasutajasõbralikkuse poolest. Selle tulemuseks võib olla kiirem ja sujuvam rakendusprotsess, kuid see võib piirata ka kohandamist ja paindlikkust. Meeskonna varasem töökogemus tööriistaga võib olla eeliseks, kuid see võib põhjustada ka kalduvust selle tööriista suhtes ja potentsiaalselt piirata muude võimaluste uurimist. Kui aga meeskonnal puudub eelnev kogemus tööriistaga, võib tekkida oht, et koolituseks kulub rohkem aega ja ressursse.

Ühilduvus Jira tarkvaraga ja veebirakenduste arendamise programmeerimiskeelte (nt C#) tugi on olulised tegurid, mida tuleb arvestada. Katalon Studio, Tricentis, Selenium, Cypress, Playwright, Robot Framework, WebdriverIO ja Selenium WebDriver vastavad kõik nendele nõuetele, kuid neil võib olla erinev funktsionaalsus ja teatud funktsioonide tugi. Kui aga tööriist või raamistik ei toeta vajalikke programmeerimiskeeli või tarkvara, ei pruugi see olla projekti jaoks mõistlik valik.

Automatiseeritud testimine on tõhususe ja täpsuse parandamise oluline nõue. Katalon Studio, Tricentis, testRigor, BugBug, Mabl, Selenium, Cypress, Playwright, Robot Framework, WebdriverIO ja Selenium WebDriver toetavad kõik automatiseeritud testimist, mis võib säästa aega ja ressursse. Siiski võib automatiseeritud testimisel esineda vigade või ebatäpsuste oht, mis võib põhjustada valepositiivseid või -negatiivseid tulemusi. Samuti võib liigne automatiseerimisele tuginemine viia oluliste servajuhtumite tähelepanuta jätmiseni ja vähendada testimise üldist tõhusust. Oluline on leida õige tasakaal manuaalse ja automaatse testimise vahel.

Üldiselt hõlmavad sobivate testimistööriistade ja -raamistike kasutamise eelised paremat tõhusust, täpsust ja katvust, samas kui riskid hõlmavad kohandamise ja paindlikkuse piiranguid, võimalikku tööriista kalduvust ja liigset automatiseerimisele tuginemist. Kõige sobivamad testimistööriistad ja -raamistikud on Katalon Studio, Tricentis, Selenium, Cypress, Playwright, Robot Framework, WebdriverIO ja Selenium

WebDriver. Need valikud pakuvad kasutajasõbralikku liidest, toetavad C# programmeerimiskeelt, ühilduvad Jira tarkvaraga ja toetavad automatiseeritud testimist. Lõplik otsus peab siiski põhinema iga projekti konkreetsetel vajadustel ja nõuetel.

4.2.4 Dokumentatsiooni hindamine

Dokumentatsioon on agiilse testimisprotsessi oluline aspekt, kuna see tagab arendatava veebirakenduse kvaliteedi. Erinevat tüüpi dokumentide valimisel on oluline arvestada igaühega kaasnevate võimalike eeliste ja riskidega.

Kuigi testimispoliitika kehtestab testimisosakonna põhimõtted, võib see loovust takistada, kui see on liiga jäik. Testistrateegia määratleb üldise testimisviisi, aga võib ka piirata testimise tõhusust, kui see on liiga kitsalt fokusseeritud. Testiplaan on määrava tähtsusega ulatuse, ajakava ja vajalike ressursside kirjeldamiseks, kuid see peab olema kohandatav testimiskeskkonna muutustega.

Testimise aluseks on testjuhtumid ning testitulemused vaadatakse hiljem üle. Kuigi testistsenaariumid võivad olla kasulikud, võib testjuhtumitest tõhusaks testimiseks piisata. Testiandmed võib lisada testimispoliitikasse ning testilogi saab siduda koondaruandega. Jälgitavuse maatriksi saab lisada testiplaani või koondaruandesse. Vead tuleb registreerida, kuid veaaruande saab ühildada koondaruandega.

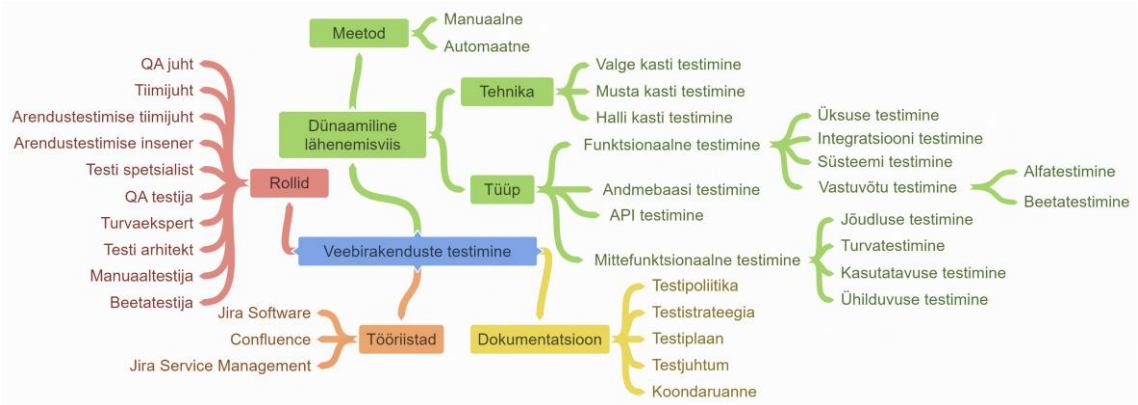
Koondaruanne sisaldab olulist teavet testitulemuste kohta, samas kui kasutaja vastuvõtuaruanne võib anda väärtuslikku tagasisidet veebirakenduse kasutuskogemuse kohta. Kuigi kõik dokumentatsioonitüübid ei ole agiilse testimisprotsessi jaoks hädavajalikud, peab koostatav dokumentatsioon olema piisav, et veebirakendus oleks kvaliteetne ja vastaks huvirühmade nõuetele.

5 Veebirakenduste testimisprotsessi täiustuste kavandamine

Eelmistes peatükkides uuriti veebirakenduste testimise erinevaid komponente, sealhulgas testitüüpe, rolle, tööriistu ja dokumentatsiooni. Samuti hinnati nende sobivust ettevõtte nõuetele. Käesolevas peatükis valitakse iga valdkonna jaoks sobivaimad komponendid ja pakutakse välja täiustamise võimalused. Peatüki eesmärk on pakkuda praktilisi võimalusi Eleport Innovations OÜ veebirakenduste testimisprotsessi tõhustamiseks.

Täiustuste kavandamisel arvestati teises peatükis välja toodud peamisi väljakutseid, nagu testijate professionaalsete teadmiste ja oskuste puudumine testimisel, vastuolulised rollid arendajate ja testijatena ning testimise ebapiisav automatiseerimine. Nende probleemide lahendamiseks esitati visiooni, võttes arvesse olemasolevaid ressursse ja ettevõtte kohanemisvõimet juurutamisetapis.

Järgnev joonis (Joonis 3) illustreerib erinevaid parendusplaani valdkondi ja valitud komponente veebirakenduste testimisprotsessi täiustamiseks. Joonise keskpunkt on veebirakenduste testimine, mida ümbritseb neli põhivaldkonda: rollid, dünaamiline lähenemisviis, dokumentatsioon ja tööriistad. Need valdkonnad on üksteisest sõltuvad ja tihedalt seotud, töötades koos, et tagada testimisprotsessi efektiivsuse. Tegeledes peamiste väljakutsetega ja rakendades täiustusi kõigis neis valdkondades, saab Eleport Innovations OÜ tõhustada oma veebirakenduste testimise protsessi ja saavutada paremaid tulemusi.



Joonis 3. Veebirakenduse testimise komponendid.

Käesolev peatükk koosneb viiest alajaotisest. Esimesed neli alapeatükki annavad üksikasjaliku ülevaate iga joonisel (Joonis 3) näidatud valdkonna komponentidest ja

nende valikukriteeriumidest. Viies alapeatükk võtab kokku peamised parenduspunktid ja annab soovitusi ettevõtte veebirakenduste testimise protsessi tõhustamiseks.

5.1 Testimistüüpide valik

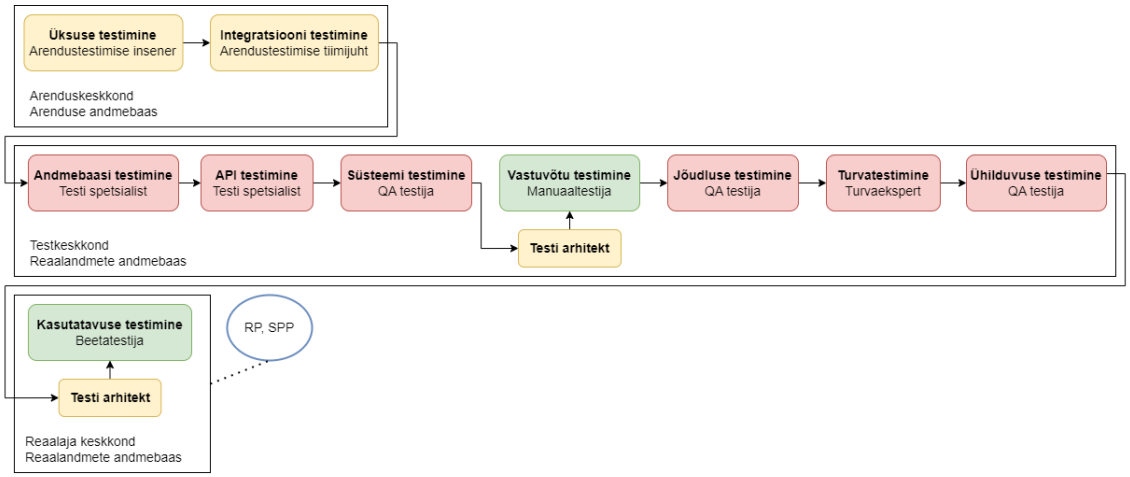
Uus testimislahendus kasutab dünaamilist lähenemist, mis ühendab manuaalse ja automatiseeritud testimismeetodid, et pakkuda testimisprotsessile olulist väärtust ja eeliseid. Manuaaltestimine on peamiselt reserveeritud vastuvõtu ja kasutatavuse testimiseks, samas kui automatiseeritud testimist kasutatakse muude testitüüpide jaoks. Testimismeeskond kasutab kõiki kolme kasti tehnikat, arendajad kasutavad valge kasti testimist ja manuaaltestijad musta kasti testimist. Teised testijad võivad sõltuvalt nende konkreetsetest ülesannetest, oskustest ja teadmistest kasutada valge kasti või halli kasti testimist.

Prioriteediks on ettevõtte jaoks kõige olulisemad testitüübid, sealhulgas kõik funktsionaalsed ja mittefunktsionaalsed tüübid. Funktsionaalsed testimise tüübid hõlmavad üksuse testimist, integratsiooni testimist, süsteemi testimist ja vastuvõtu testimist (sh alfa- ja beetatestimist), samas kui mittefunktsionaalsed testimise tüübid hõlmavad jõudluse testimist, turvatestimist, kasutatavuse testimist ja ühilduvuse testimist. Lisaks viiakse läbi andmebaaside ja API testimist.

Vastavalt testimisprotsessi neljale etapile algab protsess arendustestimisega. Selles etapis teostatakse üksuse ja integratsiooni testimist arenduskeskkonna ja -andmebaasiga. Järgmised kaks etappi viiakse läbi testkeskkonnas, kasutades reaalsete andmetega andmebaasi. Teenuse testimise etapp hõlmab andmebaasi testimist, API testimist, süsteemi testimist, jõudluse testimist, turvatestimist ja ühilduvuse testimist. Piirkondliku testimise etapp viib läbi vastuvõtutesti alfatestimise vormis ja lõpetab funktsionaalse testimise. Järgmisena on mittefunktsionaalne testimine. Viimane etapp on lõppkasutaja testimine, mis hõlmab kasutatavuse testimist või vastuvõtu testimist beetatestimise vormis reaalsete andmete baasiga toodangukeskkonnas.

Järgnev joonis (Joonis 4) illustreerib testimisprotsessi jada vastavalt testimistüüpidele. Joonisel on paksus kirjas erinevad testid ja normaalkirjas selle testija. Erandina on skeemil kaasatud ka testi arhitekt, kes on vahelüli enne vastuvõtu testimist ja kasutatavuse testimist. Nooled näitavad testimisprotsessi suunda ja valged kastid tähistavad erinevaid

keskkondi ja seotud andmebaase. Testimine toimub üldjuhul ettevõtte kõigi kolme veebirakendusega, välja arvatud kasutatavuse testimine, mida tehakse ainult Registreerimislehe (RP) ja Ühekordse makse portaaliga (SPP).



Joonis 4. Testitüüpide testimise järjekord.

Rohelisega on märgitud testid, mis on täidetud, kollasega aga need, mis on osaliselt olemas või mida on võimalik olemasolevate ressurssidega teostada. Täiendavaid ressursse ja testijat nõudvad testid on tähistatud punasega. Agiilsest testimisprotsessist tingituna võib testitüüpide järjekord muutuda või korduda.

Järgnevas tabelis (Tabel 5) on loetletud testitüübid, nende eesmärgid ja iga testitüübi eest vastutav ametikoht. Nii jooniselt (Joonis 4) kui ka tabelist (Tabel 5) on näha, et nii testi spetsialistil kui ka QA testijal on mitu vastutust. Rollidest ja nende kohustustest räägitakse lähemalt järgmises 5.2 peatükis.

Tabel 5. Testitüübid, nende eesmärgid ja vastutav ametikoht.

Testitüüp	**Eesmärk**	**Vastutav ametikoht**
Üksuse testimine	Veenduda, et tarkvarakoodi väikseim üksus töötab ootuspäraselt.	Arendustestimise insener
Integratsiooni testimine	Tagada, et erinevate moodulite ja süsteemide koostöö toimib õigesti.	Arendustestimise tiimijuht
Andmebaasi testimine	Testida rakenduse andmebaasi terviklikkust, täpsust ja funktsionaalsust.	Testi spetsialist
API testimine	Tagada rakenduses kasutatavate API-de korrektne toimimine ja oodatud tulemuste saavutamine.	Testi spetsialist

Testitüüp	Eesmärk	Vastutav ametikoht
Süsteemi testimine	Tagada, et kogu süsteem sh kõik selle komponendid toimivad koos ootuspäraselt ning süsteem vastab nõuetele.	QA testija
Vastuvõtu testimine	Tagada, et süsteem vastab kliendi nõuetele ja ootustele ning on kasutuselevõtuks valmis.	Manuaaltestija
Jõudluse testimine	Tagada rakenduse vastavus jõudlusnõutele.	QA testija
Turvalisuse testimine	Tuvastada ja hinnata rakenduse turvariske ja nõrkusi ning hinnata turvameetmete tõhusust.	Turvaekspert
Ühilduvuse testimine	Tagada rakenduse õige toimimine erinevates keskkondades.	QA testija
Kasutatavuse testimine	Tagada rakenduse vastavuse kasutaja nõuetele ja ootustele.	Beetatestija

Ettevõtte jaoks on oluline alustada nende testimistüüpide täitmisega, mida on võimalik täita olemasoleva ressursiga, nagu üksuse testimine ja integratsiooni testimine. Samuti on oluline aspekt testimise automatiseerimine. Mitmeid testimistüüpe on võimalik täita automatiseerimistööriistade kasutamisega. Tööriistade ja raamistike valikust räägib lähemalt alapeatükk 5.3.

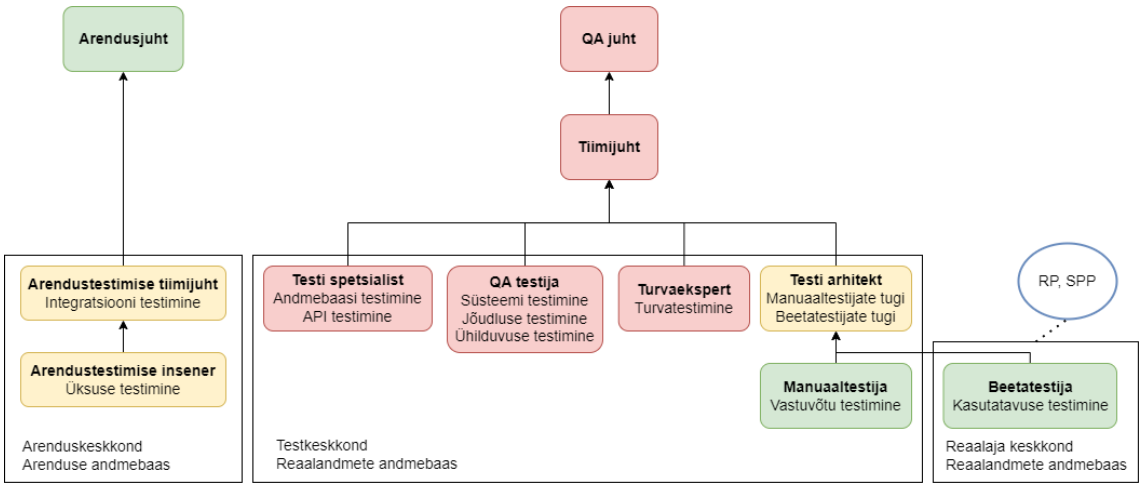
5.2 Rollide ja vastutusalade valik

Rollide ja vastutuste õigesti määratlemine ja määramine on olulise tähtsusega veebirakenduste testimise sujuva ja efektiivse töö tagamisel. Testimisprotsessi tõhusaks ja tulemuslikuks muutmiseks tuleb aga lahendada kolmest peamisest probleemist kaks. Need hõlmavad rollide ja vastutuse valikut, mis nõuavad hoolikat kaalumist.

Seni on veebirakenduste testimine tuginenud manuaaltestimisele, mida viivad läbi spetsiaalsete teadmiste ja oskusteta testijad, nagu piirkondlikud töötajad ja testkliendid. Kuigi need rollid on veebirakenduste testimise jaoks olulised, on nende võimalused teatud tüüpi testimiste läbiviimiseks piiratud. Veelgi enam, arendaja ja testija rollid on omavahel vastuolus, kuna arendajate eesmärk on luua töötav rakendus, samal ajal kui testijad keskenduvad vigade ja defektide tuvastamisele.

Nende probleemide lahendamiseks peab Eleport Innovations seadma prioriteediks pühendunud testimismeeskonna loomise, mis on võimeline läbi viima igakülgset testimist. See spetsialiseerunud meeskond suudab tagada põhjaliku ja tõhusa tarkvara testimise, vähendades tähelepanuta jäetud vigade ohtu. Meeskond koondab kõik testimistööd ning pakub piirkondlikele töötajatele ja testklientidele tuge, teavitades neid muudatustest, pakkudes testimismaterjale ja muid vajalikke ressursse. Testimisprotsessi edasiseks täiustamiseks peab ettevõte kaaluma ka kogenud testijate palkamist, kes saavad läbi viia teatud tüüpi testimisi. See samm mitte ainult ei paranda testimise kvaliteeti, vaid aitab luua ka struktureeritumat ja organiseeritumat testimisprotsessi ettevõttes.

Testimismeeskonna rollid ja ametikohad tuleb määrata nõutavate testimistüüpide, vajalike oskuste ja teadmiste, rollinõuete ja saadaolevate ressursside hoolika kaalumise alusel. Nende tegurite põhjal koostati meeskonna struktuuri, mida illustreerib testimismeeskonna struktuuriskeem (Joonis 5). Joonisel on näidatud paksemas kirjas ametikohad ja selle all tavalises kirjas selle positsiooni alla kuuluvad rollid ehk millised testimiskohustused selle positsiooniga kaasnevad. Joonisel on ka nooltega näidatud alluvussuhted ning need ametikohad, mille testimine toimub samas keskkonnas ja sama andmebaasiga, on ümbritsetud valge kastiga.



Joonis 5. Testimismeeskonna struktuuriskeem.

Kõik joonise kujutatud positsioonid vastutavad kõigi kolme veebirakenduse testimise eest, välja arvatud beetatestija, kes keskendub eranditult Registreerimislehe ja Ühekordse makse portaali testimisele (vastavalt RP ja SPP). Rohelisega tähistatud ametikohad märgivad olemasolevaid ametikohti, samas kui kollasega on märgitud ametikohad, mida saavad täita praegused töötajad, kes on valmis muutma oma ülesandeid või võtma rohkem

kohustusi. Punasega märgitud ametikohad eeldavad lisatööjõu palkamist ning arvestada tuleb sellega, et ühel ametikohal võib olla mitu töötajat.

Arendustestimise insener ja arendustestimise tiimijuht annavad aru arendusjuhile ning vastutavad veebirakenduste testimise eest arendusfaasis. Arendustestimise tiimijuht jälgib testimisprotsessi ja viib läbi integratsiooniteste, samas kui arendustestimise insener vastutab üksuse testide kirjutamise ja läbiviimise eest. QA juht on testimisosakonna kõrgeim ametikoht, kellele alluvad kõik testimisosakonna alla kuuluvad testijad.

Testimismeeskond koosneb testi spetsialistist, QA testijast, turvaeksperdist, testi arhitektist, manuaaltestijast ja beetatestijast. Paljudel ametikohtadel on mitu rolli, mida kombineeritakse ressursside vähendamiseks, kuna meeskond on väike. Kombineeritud rollide ülesanded ja nõutavad oskused on aga sarnased, mistõttu on kombineerimine lihtne ja tõhus. Ametikohtade peamised kohustused on kokku võetud järgmises tabelis (Tabel 6). Ametikoha ülesanded ja kohustused võivad muutuda vastavalt ametikoha konkreetsele sisule ja töötaja oskustele.

Tabel 6. Testimismeeskonna ametikohad ja kohustused.

Ametikoht	**Kohustused**
QA juht	- Kvaliteedi tagamise poliitikate ja protseduuride loomine ja rakendamine - Testimiseelarve väljatöötamine ja haldamine - Tööülesannete jagamine tiimijuhile - Testimistegevuse vastavuse tagamine ärieesmärkide ja eeskirjadega - Korrapäraste aruannete esitamine kõrgemale juhtkonnale testimistegevuse oleku kohta
Tiimijuht	- Tööülesannete jagamine ja vastutuse delegeerimine tiimiliikmetele - Koordineerimine teiste osakondadega, et tagada testitulemuste õigeaegne edastamine - Tiimiliikmete juhendamine ja koolitamine - Tulemuslikkuse hindamiste läbiviimine ja tiimiliikmetele tagasiside andmine - Uute tiimiliikmete töölevõtmise protsessis osalemine - Koondaruannete esitamine QA juhile

Ametikoht	**Kohustused**
Arendustestimise tiimijuht	- Kooskõlastamine arendusmeeskonnaga, et tagada testimistegevuse integreerimine arendusprotsessi - Testiplaanide ja ajakavade koostamine ja haldamine - Testimise edenemisest teavitamine testimisosakonna tiimijuhile - Testimisega seotud probleemide lahendamiseks tehakse koostööd arendajatega - Integratsiooni testimise testiplaanide väljatöötamine ja täitmine
Arendustestimise insener	- Üksusetestide kirjutamine ja täitmine - Üksuse testimise käigus leitud probleemide parandamine - Koostöö tegemine arendusmeeskonnaga, et tuvastada võimalikud probleemid ja soovitada parandusi - Koodi kvaliteedi tagamiseks koodi ülevaatustel osalemine
Testi spetsialist	- Andmebaasi ja API testimise testiplaanide väljatöötamine ja täitmine - Andmebaasi ja API testimiseks automatiseeritud testide väljatöötamine - Katsetamise käigus leitud defektide tuvastamine ja dokumenteerimine - Andmebaasi ja API-ga seotud probleemide lahendamiseks tehakse koostööd arendusmeeskonnaga
QA testija	- Süsteemi testimise, jõudluse testimise ja ühilduvuse testimise testiplaanide väljatöötamine ja täitmine - Katsetamise käigus leitud defektide tuvastamine ja dokumenteerimine - Probleemide lahendamiseks tehakse koostööd arendusmeeskonnaga
Turvaekspert	- Turvatestiplaanide väljatöötamine ja täitmine - Turvanõrkuste tuvastamine ja dokumenteerimine - Turvalisusega seotud probleemide lahendamiseks tehakse koostööd arendusmeeskonnaga - Uusimate turvatrendide ja –tehnoloogiatega kursis olemine
Testi arhitekt	- Manuaaltestijate testjuhtumite arendamine ja haldamine - Koordineerimine beetatestijatega, et tagada testimiseks piisavad ressursid - Manuaaltestijatele ja beetatestijatele juhiste ja toe pakkumine - Testimisprotsesside ja protseduuride väljatöötamine ja juurutamine

Ametikoht	Kohustused
Manuaaltestija	- Testjuhtumite teostamine vastuvõtu testimiseks alfa-testimise vormis - Katsetamise käigus leitud defektide tuvastamine ja dokumenteerimine - Süsteemi kasutatavuse ja funktsionaalsuse kohta tagasiside andmine
Beetatestija	- Rakenduste testimine kasutatavuse testimise kujul või vastuvõtu testimise beetatestimise vormis - Katsetamise käigus leitud defektide tuvastamine ja dokumenteerimine - Süsteemi kasutatavuse ja funktsionaalsuse kohta tagasiside andmine

Testimeeskonna rolle ja ametikohti saab prioriteetsuse järjekorda seada nende tähtsuse ja olemasolevate ressursside alusel. Nagu on näidatud joonisel (Joonis 4) rohelisega, on mõned ametikohad juba täidetud, samas kui teisi saab täita olemasolevate töötajatega, mida näitab kollane varjund. Arendusfaasis testimine on kriitilise tähtsusega ja seda tuleb eelistada, eriti kui on võimalik automatiseerida. QA testija ametikoht on punasega märgitud ametitest kõige olulisem, kuna see hõlmab süsteemi testimist, jõudluse testimist ja ühilduvuse testimist. Testimisprotsessi koordineerimiseks ja tööst selge ülevaate omamiseks on oluline ka tiimijuht. Väiksemate meeskondade puhul võib tiimijuhist piisata ehk ta võtab endale ka QA juhi kohustused.

Testimeeskonnas on rollid ja ametikohad hoolikalt läbi mõeldud, lähtudes nõutavatest oskustest ja teadmistest, rollinõuetest ja saadaolevatest ressurssidest. Meeskonna struktuur on loodud veebirakenduste tõhusa testimise tagamiseks kõigis keskkondades. Ametikohtade täitmise järjekorravalik sõltub ettevõtte veebirakenduste testimise prioriteetsusest.

5.3 Tööriistade ja raamistike valik

Olemasolevate tehnoloogiate kasutamine aitab oluliselt vähendada kulusid ja optimeerida testimisprotsessi. Üks tõhus viis testjuhtumite haldamiseks on Jira tarkvara kasutamine, mis võimaldab luua eraldi tarkvaraprojekti, mis on pühendatud ainult testjuhtumite haldamisele. See tagab, et kõik testjuhtumid on ajakohased, neid jälgitakse ja täidetakse vastavalt vajadusele. Lisaks saab Confluence'i dokumendihaldustarkvara kasutada muude

oluliste dokumentide haldamiseks, nagu testimispoliitikad ja testitulemuste kokkuvõtlikud aruanded. Confluence'i abil saab kogu testimise dokumentatsiooni salvestada kesksesse asukohta, mis muudab selle hõlpsasti kättesaadavaks kõigile meeskonnaliikmetele.

Peatükis 4.2 hinnati mitmeid tööriistu ja raamistikke, sealhulgas Katalon Studio, Tricentis, Selenium, Cypress, Playwright, Robot Framework, WebdriverIO ja Selenium WebDriver. Need tööriistad leiti olevat kõige sobivamad nende kasutajasõbraliku liidese, C# programmeerimiskeele toe, Jira tarkvaraga ühilduvuse ja automatiseeritud testimise toe tõttu. Lõputöö koostamise ajapiirangu tõttu ei olnud aga võimalik nende tööriistade ja raamistike kasutamist põhjalikumalt uurida, et integreerida need ettevõtte testimisprotsessi. Seetõttu jätkub ettevõttesiseselt sobivaima tööriista ja raamistiku otsimine ka väljaspool lõputööd.

5.4 Dokumentatsiooni valik

Dokumentatsioonil on testimisprotsessis ülioluline roll, kuna see tagab testimisprotseduuride ja -tulemuste järjepidevuse ja selguse. Ettevõtte jaoks on olulised viis peamist dokumenti: testipoliitika, testistrateegia, testiplaan, testjuhtumid ja koondaruanne.

Testipoliitika toob välja üldised testimise põhimõtted ja juhised ettevõtte jaoks. Lisaks võib poliitika sisaldada olulisi mõisteid, mida iga testimismeeskonnaliige peab teadma. See täpsustab, millist tüüpi testimist ettevõte teeb ja kuidas ta testiandmeid haldab. See dokument on oluline kõigi projektide testimise järjepidevuse tagamiseks. Testipoliitika koostab QA juht koostöös teiste osakondadega (arendusosakond, tooteosakond, operatsioonide osakond) ja ettevõtte kõrgema juhtkonna ehk tehnoloogiajuhiga.

Testistrateegia kirjeldab testimise eesmärkide saavutamiseks kasutatavat lähenemisviisi ja tehnikaid. See kirjeldab testimise ulatust, testimise eesmärke, testimise liike, testimistehnikaid ning testimise läbiviimiseks vajalikke tööriistu ja ressursse. Selle koostab tiimijuht koostöös QA juhiga.

Testiplaan kirjeldab üksikasjalikku lähenemisviisi konkreetse valdkonna testimiseks. See täpsustab testimise ulatuse, testi eesmärgid, testimiskeskkonna, testimise ajakava,

testimistehnikad ning testimismeeskonna rollid ja kohustused. Testiplaani koostab tiimijuht koostöös konkreetse valdkonna vastutajaga.

Testjuhtum määrab konkreetse testi testisammud ja eeldatavad tulemused. See sisaldab iga testjuhtumi sisendandmeid, eeldatavat väljundit ja eeltingimusi. Iga testija koostab iga testitüübi kohta eraldi testjuhtumid ning seejärel viib need läbi. Manuaaltestimise puhul loob testjuhtumid testi arhitekt ning testid viib läbi manuaaltestija.

Koondaruanne võtab kokku testimisprotsessi tulemused. See võib sisaldada testilogi, jälgitavusmaatriksit, veaaruandeid ja kasutajate vastuvõtuaruannet. See aruanne on oluline testimise tulemuste edastamiseks sidusrühmadele ja testimise käigus leitud defektide dokumenteerimiseks. Iga valdkonna vastutaja teeb oma testijuhtumite tulemustest kokkuvõtte, millest seejärel teeb koondkokkuvõtte tiimijuht. Tiimijuht esitab koondaruande QA juhile, kes teeb selle põhjal kokkuvõtteid teistele osakondadele.

Dokumentatsiooni täiustustega tuleb alustada testjuhtumite ja koondaruannete üle vaatamise ja nende täiustamisega. Mida põhjalikumad ja selgemad on aruanded, seda lihtsam on analüüsida testide tulemusi. Lisaks on kriitilise tähtsusega ka vigade ja defektide raporteerimine, milleks kasutab ettevõte Jira Service Management haldustarkvara. See on väärtuslik tööriist, mille kasutamist tuleb kindlasti jätkata.

Dokumentatsiooni järjepidevuse ja täpsuse tagamiseks tuleb kehtestada dokumentide kontrollimise protsess. See protsess peaks hõlmama versioonikontrolli, üle vaatamist ja kinnitamist ning juurdepääsu kontrolli. Dokumentatsioonide haldamiseks tuleb kasutada Confluence'i dokumendihalduskeskkonda, mis juba on ettevõttes kasutusel. Korrektne dokumentatsioonihaldus tagab testimise järjepidevuse ja täpsuse ning aitab tulemusi sidusrühmadele edastada.

5.5 Järeldused

Veebirakenduste töökindluse ja tõhususe tagamiseks on oluline, et tarkvara testimisprotsessid järgiksid valdkonna parimaid tavasid ja standardeid. Tarkvarasüsteemi regulaarseks ja süstemaatiliseks testimiseks on tungivalt soovitatav luua spetsialiseerunud tarkvara testimismeeskond. See meeskond peab koosnema liikmetest, kellel on vajalikud oskused, teadmised ja ressursid põhjaliku testimise läbiviimiseks, tarkvaravigade

tuvastamiseks ja nendest teavitamiseks ning tarkvara vastavuse tagamiseks soovitud kvaliteedistandarditele ja -kriteeriumidele.

Tõhusa testimisprotsessi rakendamine ei taga ainult veebirakenduste töökindlust ja kvaliteeti, vaid suurendab ka klientide rahulolu, tugevdab kaubamärgi mainet ja annab turul konkurentsieelise. Tõhusa testimisprotsessi loomiseks peab ettevõte kaaluma arendustestimise inseneri, arendustestimise tiimijuhi ja QA testija palkamist, kes vastutavad kõige olulisemate testimistüüpide eest: üksuse testimine, integratsiooni testimine, süsteemi testimine, jõudluse testimine ja ühilduvuse testimine. Siiski on oluline meeles pidada, et olemasolevate ressursside ärakasutamiseks tuleb jätkata manuaaltestija ja beetatestija rollidega.

Lisaks peab ettevõte eelistama automatiseeritud testimistööriistade ja -tehnikate kasutamist, et kiirendada testimisprotsessi, parandada testide ulatust ja minimeerida inimlike vigade riski. Üksuse, integratsiooni, süsteemi, jõudluse ja ühilduvuse testimist saab ka sobivate testimistööriistade ja -raamistike abil automatiseerida. Kuna konkreetseid automatiseerimise tööriistu ja raamistikke ei valitud lõputöö ajapiirangu tõttu, siis anti suund, milliseid tööriistu ja raamistikke edasi uurida. Tööriistade ja raamistike lõplikul valikul tuleb aga arvestada ka testitüüpide ja ametikohtade valikuga ning seega on vaja täiendavat ettevõttesisest uurimistööd väljaspool lõputööd.

Dokumentatsioon on testimisprotsessi oluline komponent ja sellele tuleb omistada sama tähtsust kui mis tahes muule protsessile. Olemasolevaid dokumente, nagu testjuhtumid ja koondaruanded, tuleb pidevalt täiustada. Samuti on tungivalt soovitatav luua testimispoliitika, et kõigil testimismeeskonna liikmetel oleks ühtne arusaam sellest, mis on testimine, mida testitakse, millised on nõuded, jne. Testjuhtumite ja dokumentatsiooni nõuetekohase haldamise tagamiseks tuleb jätkata Jira Software, Jira Service Management ja Confluence tarkvara kasutamist.

6 Kokkuvõte

Lõputöö eesmärgiks oli pakkuda välja võimalused Eleport Innovations OÜ veebirakenduste testimisprotsessi tõhustamiseks, milleks analüüsiti ettevõtte hetkeolukorda ning selgitati välja peamised väljakutsed ja piirangud. Leiti, et Eleport Innovations OÜ veebirakenduste testimise hetkeseis nõuab veebirakenduste kvaliteedi, töökindluse ja turvalisuse tagamiseks täiendusi, mida on võimalik saavutada läbi agiilsete metoodikate kasutamise, et tõsta testimise efektiivsust ja üldist tarkvara kvaliteeti, mis toob kaasa kõrgema kliendirahulolu.

Selle saavutamiseks kasutati kvalitatiivset andmekogumismeetodit, et uurida tarkvara testimise erinevaid aspekte, sealhulgas rolle, tööriistu ja dokumentatsiooni ning veebirakenduste testimise olemust. Analüüsiti erinevaid testimistüüpe, nagu dünaamiline, funktsionaalne ja mittefunktsionaalne testimine, ning uuriti manuaalseid ja automatiseeritud testimise meetodeid ja raamistikke.

Seejärel pandi paika erinevate komponentide nõuded ning nende nõuete järgi hinnati kogutud teabe sobivust. Viimase peatükina koostati täiustamise võimalused ning anti soovitusi ettevõtte veebirakenduste testimisprotsessi tõhustamiseks. Kavandatavate täiustuste hulka kuulusid pühendunud testimismeeskond, automatiseeritud testimistööriistade integreerimine ja pideva testimise rakendamine.

Kuigi täiustusi ei saanud ajapiirangu tõttu ellu viia, andis ettevõtte tehnoloogiajuht tagasisidet kavandatavatele parendustele (Lisa 2). Üldiselt jäi ettevõte pakutud täiustustega rahule, kuid edasine arendus ja rakendamine sõltub ettevõtte äriprotsessidest ja ressurssidest. Käesolev lõputöö on täitnud oma eesmärgi ja andnud soovitusi veebirakenduste testimise protsessi täiustamiseks ettevõttes Eleport Innovations OÜ.

Kasutatud kirjandus

[1] QuestionPro, „Research Design: What it is, Elements & Types,“ [Võrgumaterjal]. Loetud aadressil: https://www.questionpro.com/blog/research-design#research_design_types. [Kasutatud 17. aprill 2023].

[2] QuestionPro, „Qualitative Research Methods: Types & Examples,“ [Võrgumaterjal]. Loetud aadressil: https://www.questionpro.com/blog/qualitative-research-methods/. [Kasutatud 17. aprill 2023].

[3] S. Laoyan, „What is Agile methodology? (A beginner’s guide),“ Asana, 15. oktoober 2022. [Võrgumaterjal]. Loetud aadressil: https://asana.com/resources/agile-methodology. [Kasutatud 17. aprill 2023].

[4] Rapid7, „Software Development Life Cycle (SDLC),“ [Võrgumaterjal]. Loetud aadressil: https://www.rapid7.com/fundamentals/software-development-life-cycle-sdlc/. [Kasutatud 31. märts 2023].

[5] Triad, „What is the difference between Testing and Quality Assurance? And, does it matter?,“ [Võrgumaterjal]. Loetud aadressil: https://www.triad.co.uk/news/testing-vs-qa/]. [Kasutatud 4. aprill 2023].

[6] T. Hamilton, „STLC (Software Testing Life Cycle) Phases, Entry, Exit Criteria,“ 4. märts 2023. [Võrgumaterjal]. Loetud aadressil: https://www.guru99.com/software-testing-life-cycle.html. [Kasutatud 31. märts 2023].

[7] Microfocus, „What Is Agile Software Testing?,“ [Võrgumaterjal]. Loetud aadressil: https://www.microfocus.com/en-us/what-is/agile-testing. [Kasutatud 6. aprill 2023].

[8] Ramotion, „What is a Web Application?,“ 1. juuni 2021. [Võrgumaterjal]. Loetud aadressil: https://www.ramotion.com/blog/what-is-a-web-application/. [Kasutatud 18. aprill 2023].

[9] R. Vogels, „A 6-Step Guide to Web Application Testing,“ [Võrgumaterjal]. Loetud aadressil: https://usersnap.com/blog/web-application-testing/. [Kasutatud 18. aprill 2023].

[10] T. Hamilton, „Web Application Testing: How to Test a Website?,“ 25. märts 2023. [Võrgumaterjal]. Loetud aadressil: https://www.guru99.com/web-application-testing.html. [Kasutatud 31. märts 2023].

[11] T. Hamilton, „What is Static Testing? Software Testing Techniques,“ 8. aprill 2023. [Võrgumaterjal]. Loetud aadressil: https://www.guru99.com/testing-review.html. [Kasutatud 15. aprill 2023].

[12] T. Hamilton, „Static Vs Dynamic Testing: Difference Between Them,“ 14. veebruar 2023. [Võrgumaterjal]. Loetud aadressil: https://www.guru99.com/static-dynamic-testing.html. [Kasutatud 15. aprill 2023].

[13] pp_pankaj, „Software Testing | Dynamic Testing,“ [Võrgumaterjal]. Loetud aadressil: https://www.geeksforgeeks.org/software-testing-dynamic-testing/. [Kasutatud 14. aprill 2023].

[14] pp_pankaj, „Difference between Static and Dynamic Testing,“ [Võrgumaterjal]. Loetud aadressil: https://www.geeksforgeeks.org/difference-between-static-and-dynamic-testing/. [Kasutatud 14. aprill 2023].

[15] Javatpoint, [Võrgumaterjal]. Loetud aadressil: https://www.javatpoint.com/static-testing-vs-dynamic-testing. [Kasutatud 14. aprill 2023].

[16] Testbytes, „Pros and Cons of Dynamic Testing and Static Testing,“ 31. august 2017. [Võrgumaterjal]. Loetud aadressil: https://www.testbytes.net/blog/dynamic-testing-and-static-testing/. [Kasutatud 15. aprill 2023].

[17] Global App Testing, „Manual Testing - what is it?,“ [Võrgumaterjal]. Loetud aadressil: https://www.globalapptesting.com/manual-testing-best-practices. [Kasutatud 15. aprill 2023].

[18] Testbytes, „Web Application Testing Manually (Step by Step A Complete Guide),“ 24. november 2022. [Võrgumaterjal]. Loetud aadressil: https://www.testbytes.net/blog/web-application-testing-manually/. [Kasutatud 15. aprill 2023].

[19] R. Singh ja V. Singh, „Manual Testing,“ 19. oktoober 2021. [Võrgumaterjal]. Loetud aadressil: https://www.toolsqa.com/software-testing/manual-testing/. [Kasutatud 15. aprill 2023].

[20] T. Hamilton, „What is Automation Testing? Test Tutorial,“ 9. veebruar 2023. [Võrgumaterjal]. Loetud aadressil: https://www.guru99.com/automation-testing.html. [Kasutatud 15. aprill 2023].

[21] Software Testing Help, „What Is Automation Testing (Ultimate Guide To Start Test Automation),“ 23. märts 2023. [Võrgumaterjal]. Loetud aadressil: https://www.softwaretestinghelp.com/automation-testing-tutorial-1/. [Kasutatud 15. aprill 2023].

[22] ashushrma378, „Advantages and Disadvantages of Automated Testing,“ [Võrgumaterjal]. Loetud aadressil: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-automated-testing/. [Kasutatud 15. aprill 2023].

[23] Javatpoint, „Automation Testing vs. Manual Testing,“ [Võrgumaterjal]. Loetud aadressil: https://www.javatpoint.com/automation-testing-vs-manual-testing. [Kasutatud 15. aprill 2023].

[24] T. Hamilton, „Difference Between Manual and Automation Testing,“ 4. märts 2023. [Võrgumaterjal]. Loetud aadressil: https://www.guru99.com/difference-automated-vs-manual-testing.html. [Kasutatud 15. aprill 2023].

[25] A. Pai, „Manual Testing vs Automation Testing: How to optimize for testing & costs,“ 15. detsember 2022. [Võrgumaterjal]. Loetud aadressil: https://www.browserstack.com/guide/manual-vs-automated-testing-differences. [Kasutatud 15. aprill 2023].

[26] T. Hamilton, „What is Dynamic Testing? Types, Techniques & Example,“ 11. märts 2023. [Võrgumaterjal]. Loetud aadressil: https://www.guru99.com/dynamic-testing.html. [Kasutatud 16. aprill 2023].

[27] T. Hamilton, „White Box Testing – What is, Techniques, Example & Types,“ 25. märts 2023. [Võrgumaterjal]. Loetud aadressil: https://www.guru99.com/white-box-testing.html. [Kasutatud 16. aprill 2023].

[28] Imperva, „Black Box Testing,“ [Võrgumaterjal]. Loetud aadressil: https://www.imperva.com/learn/application-security/black-box-testing/. [Kasutatud 16. aprill 2023].

[29] Intellipaat, „What is Grey Box Testing?,“ 8. märts 2023. [Võrgumaterjal]. Loetud aadressil: https://intellipaat.com/blog/what-is-grey-box-testing/. [Kasutatud 16. aprill 2023].

[30] Javatpoint, „Black Box Testing vs. White Box Testing vs. Grey Box Testing,“ [Võrgumaterjal]. Loetud aadressil: https://www.javatpoint.com/black-box-testing-vs-white-box-testing-vs-grey-box-testing. [Kasutatud 16. aprill 2023].

[31] Tutorialspoint, „Software Testing - Methods,“ [Võrgumaterjal]. Loetud aadressil: https://www.tutorialspoint.com/software_testing/software_testing_methods.htm. [Kasutatud 16. aprill 2023].

[32] tarunsinghwap7, „Difference between Black Box Vs White Vs Grey Box Testing,“ [Võrgumaterjal]. Loetud aadressil: https://www.geeksforgeeks.org/difference-between-black-box-vs-white-vs-grey-box-testing/. [Kasutatud 16. aprill 2023].

[33] D. Shain, „What is Functional Testing? Types and Example (Full Guide),“ 13. mai 2022. [Võrgumaterjal]. Loetud aadressil: https://applitools.com/blog/functional-testing-guide/. [Kasutatud 18. aprill 2023].

[34] M. Calvello, „The 4 Levels of Testing in Software Engineering Explained,“ 8. november 2022. [Võrgumaterjal]. Loetud aadressil: https://fellow.app/blog/engineering/the-levels-of-testing-in-software-engineering-explained/#4. [Kasutatud 18. aprill 2023].

[35] Software Testing Help, „Types Of Software Testing: Different Testing Types With Details,“ 14. märts 2023. [Võrgumaterjal]. Loetud aadressil: https://www.softwaretestinghelp.com/types-of-software-testing/#4_Acceptance_Testing. [Kasutatud 18. aprill 2023].

[36] S. Jain, „What is Non-Functional Testing?,“ 3. veebruar 2023. [Võrgumaterjal]. Loetud aadressil: https://www.browserstack.com/guide/what-is-non-functional-testing. [Kasutatud 18. aprill 2023].

[37] Smartbear, „Software Testing Methodologies,“ [Võrgumaterjal]. Loetud aadressil: https://smartbear.com/learn/automated-testing/software-testing-methodologies/. [Kasutatud 18. aprill 2023].

[38] Katalon, „What is Regression Testing? Definition, Tools & How to Get Started,“ [Võrgumaterjal]. Loetud aadressil: https://katalon.com/resources-center/blog/regression-testing. [Kasutatud 18. aprill 2023].

[39] T. Patel, „A Comprehensive Guide to Interface Testing [with Examples],“ 18. aprill 2022. [Võrgumaterjal]. Loetud aadressil: https://www.testgrid.io/blog/interface-testing/. [Kasutatud 18. aprill 2023].

[40] ReQtest, „A Complete Guide To Database Testing With Examples,“ 27. juuli 2020. [Võrgumaterjal]. Loetud aadressil: https://reqtest.com/testing-blog/database-testing/. [Kasutatud 18. aprill 2023].

[41] Postman, „API testing overview,“ [Võrgumaterjal]. Loetud aadressil: https://www.postman.com/api-platform/api-testing/. [Kasutatud 18. aprill 2023].

[42] Software Testing Help, „Functional Testing Vs Non-Functional Testing,“ 23. märts 2023. [Võrgumaterjal]. Loetud aadressil: https://www.softwaretestinghelp.com/functional-testing-vs-non-functional-testing/#Difference_Between_Functional_and_Non-Functional_Testing. [Kasutatud 18. aprill 2023].

[43] pp_pankaj, „Differences between Functional and Non-functional Testing,“ [Võrgumaterjal]. Loetud aadressil: https://www.geeksforgeeks.org/differences-between-functional-and-non-functional-testing/. [Kasutatud 18. aprill 2023].

[44] Johnny, „Inside The World Of Software Testing: A Breakdown Of Roles,“ 12. märts 2023. [Võrgumaterjal]. Loetud aadressil: https://testertips.com/software-testing-roles-and-responsibilities/. [Kasutatud 11. aprill 2023].

[45] O. Sheremeta, „Roles & Responsibilities in a Software testing team,“ 1. august 2022. [Võrgumaterjal]. Loetud aadressil: https://testomat.io/blog/roles-responsibilities-in-a-software-testing-team/. [Kasutatud 11. aprill 2023].

[46] Testbytes, „Roles & Responsibilities in a Software Testing Team,“ 26. detsember 2017. [Võrgumaterjal]. Loetud aadressil: https://www.testbytes.net/blog/software-testing-team/. [Kasutatud 11. aprill 2023].

[47] R. Walker, „How Stakeholders are Involved in Beta Tests,“ [Võrgumaterjal]. Loetud aadressil: https://www.centercode.com/blog/how-stakeholders-are-involved-in-beta-tests. [Kasutatud 11. aprill 2023].

[48] R. Lynn, „What is meant by testing tools?,“ 10. veebruar 2023. [Võrgumaterjal]. Loetud aadressil: https://www.quora.com/What-is-meant-by-testing-tools. [Kasutatud 23. aprill 2023].

[49] Smartbear, „Test Automation Frameworks,“ [Võrgumaterjal]. Loetud aadressil: https://smartbear.com/learn/automated-testing/test-automation-frameworks/. [Kasutatud 23. aprill 2023].

[50] J. Charlton, „10 Best Web Application Testing Tools In 2023,“ 17. jaanuar 2023. [Võrgumaterjal]. Loetud aadressil: https://theqalead.com/tools/best-web-application-testing-tools/. [Kasutatud 23. aprill 2023].

[51] V. Lam, „10 Best Web Automation Tools For QA In 2023,“ 29. märts 2023. [Võrgumaterjal]. Loetud aadressil: https://theqalead.com/tools/best-web-automation-tools/. [Kasutatud 23. aprill 2023].

[52] T. Hamilton, „21 BEST Website Testing Tools (Web Application Testing) 2023,“ 1. märts 2023. [Võrgumaterjal]. Loetud aadressil: https://www.guru99.com/top-20-web-testing-tools.html. [Kasutatud 23. aprill 2023].

[53] Katalon, „Katalon Pricing,“ [Võrgumaterjal]. Loetud aadressil: https://katalon.com/pricing. [Kasutatud 23. aprill 2023].

[54] Tricentis, „Request Tricentis Test Automation pricing,“ [Võrgumaterjal]. Loetud aadressil: https://www.tricentis.com/products/tricentis-test-automation/pricing. [Kasutatud 23. aprill 2023].

[55] testRigor, „Select a plan to proceed,“ [Võrgumaterjal]. Loetud aadressil: https://testrigor.com/sign-up/. [Kasutatud 23. aprill 2023].

[56] BugBug, „Effortless cloud testing that fits your budget,“ [Võrgumaterjal]. Loetud aadressil: https://bugbug.io/pricing/. [Kasutatud 23. aprill 2023].

[57] Mabl, „Plans that Scale with You,“ [Võrgumaterjal]. Loetud aadressil: https://www.mabl.com/pricing. [Kasutatud 23. aprill 2023].

[58] Selenium, „Getting Started,“ [Võrgumaterjal]. Loetud aadressil: https://www.selenium.dev/. [Kasutatud 23. aprill 2023].

[59] Testpad, „Pricing,“ [Võrgumaterjal]. Loetud aadressil: https://testpad.com/plans. [Kasutatud 23. aprill 2023].

[60] BrowserStack, „Pricing,“ [Võrgumaterjal]. Loetud aadressil: https://www.browserstack.com/pricing?product=automate. [Kasutatud 23. aprill 2023].

[61] Mailosaur, „Top 4 frameworks for testing web-based applications,“ [Võrgumaterjal]. Loetud aadressil: https://mailosaur.com/blog/top-4-frameworks-for-testing-web-based-applications/. [Kasutatud 23. aprill 2023].

[62] GH ja A. Badkar, „Popular Test Automation Frameworks,“ 9. märts 2023. [Võrgumaterjal]. Loetud aadressil: https://www.browserstack.com/guide/best-test-automation-frameworks. [Kasutatud 23. aprill 2023].

[63] J. Colantonio, „11 top open-source test automation frameworks: How to choose,“ detsember 2019. [Võrgumaterjal]. Loetud aadressil: https://techbeacon.com/app-dev-testing/top-11-open-source-testing-automation-frameworks-how-choose. [Kasutatud 23. aprill 2023].

[64] Cypress, „Pricing,“ [Võrgumaterjal]. Loetud aadressil: https://www.cypress.io/pricing?v=2#monthly. [Kasutatud 23. aprill 2023].

[65] Playwright, „Getting starter,“ [Võrgumaterjal]. Loetud aadressil: https://playwright.dev/. [Kasutatud 23. aprill 2023].

[66] Robot Framework, „Introduction,“ [Võrgumaterjal]. Loetud aadressil: https://robotframework.org/. [Kasutatud 23. aprill 2023].

[67] WebdriverIO, „Home page,“ [Võrgumaterjal]. Loetud aadressil: https://webdriver.io/. [Kasutatud 23. aprill 2023].

[68] B. Dzhekishev ja A. Topalidis, „Test Documentation in Software Testing: Internal test documents,“ 6. aprill 2022. [Võrgumaterjal]. Loetud aadressil:

https://maddevs.io/blog/test-documentation-in-software-testing/#internal-test-documents. [Kasutatud 21. aprill 2023].

[69] B. Dzhekishev ja A. Topalidis, „Test Documentation in Software Testing: External test documents,“ 6. aprill 2022. [Võrgumaterjal]. Loetud aadressil: https://maddevs.io/blog/test-documentation-in-software-testing/#external-test-documents. [Kasutatud 21. aprill 2023].

[70] B. Dzhekishev ja A. Topalidis, „Test Documentation in Software Testing: What are the main goals of test documentation?,“ 6. aprill 2022. [Võrgumaterjal]. Loetud aadressil: https://maddevs.io/blog/test-documentation-in-software-testing/#what-are-the-main-goals-of-test-documentation. [Kasutatud 21. aprill 2023].

[71] B. Dzhekishev ja A. Topalidis, „Test Documentation in Software Testing: How to write a test document?,“ 6. aprill 203. [Võrgumaterjal]. Loetud aadressil: https://maddevs.io/blog/test-documentation-in-software-testing/#how-to-write-a-test-document. [Kasutatud 21. aprill 2023].

Lisa 1 – Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[1]

Mina, Regina Johanson

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Veebirakenduste testimisprotsessi täiustamise võimalused ettevõttes Eleport Innovations OÜ“, mille juhendaja on Inna Švartsman
   1.1. reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
   1.2. üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.
2. Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.
3. Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

16.05.2023

1 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Eleport Innovations OÜ tehnoloogiajuhi hinnang täiustustele

„Eleport on *start-up*, mis tähendab, et ettevõtte ressursid on väga piiratud ning sellest tingituna mulle meeldis väga lõputöös olev kommentaar leheküljel 44, mis soovitab ressursside kokkuhoiuks erinevaid testimisega seotud rolle kombineerida. See tähendab, et üliõpilane on selgelt aru saanud, et teooria ning praktika on kaks erinevat asja ning parimate tavade rakendamisel tuleb teha mööndusei d. Tudengi poolt kirjutatud lõputöö on loogiline ja lähtub parimast tööstuse praktikast ning on kindlasti õige suund kuhu Eleport Innovations peaks liikuma. Minu hinnangul on antud lõputöös väljapakutud lahendus eesmärk, mille suunas peab ettevõte liikuma pikemaajalise testimiskorralduse visiooni täitmiseks.“

Eleport Innovations OÜ tehnoloogiajuht

Rain Neemlaid

11.05.2023