TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Teele Pae 192418IABM

Läbivtestide testimisvahendi valik veebirakenduse näitel

Magistritöö

Juhendaja: Jekaterina Tšukrejeva
Magistrikraad

Tallinn 2022

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: Teele Pae

11.05.2022

Annotatsioon

Tänapäeval on olemas palju erinevaid läbivtestide töövahendeid. Samas on keeruline teha valik vaid nende kodulehel oleva info põhjal. Vale otsuse korral on läbivtestide töövahendi vahetamine aega nõudev tegevus, ning selle tõttu ärile kulukas. Magistritöö eesmärk oli leida hea meetod läbivtestide töövahendi valimiseks. Vaadati üle hetkel olemasolevad meetodid automaattestide töövahendite võrdlemiseks. Valiti esialgsete kriteeriumite põhjal potentsiaalsed läbivtestide töövahendid, millele rakendati populaarsuse mõõdikut. Seejärel tehti kolm testi kasutades välja valitud töövahendeid ning rakendati testidele testmõõdikud. Tulemuste põhjal valiti välja läbivtestide töövahend veebirakendusele Dashboard.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 40 leheküljel, 6 peatükki, 11 joonist, 11 tabelit.

Abstract

Choosing an End-to-End Testing Tool for a Web Application

Nowadays there are many end-to-end testing tools to choose from. Choosing the right one based on their homepage can be quite complex. If a wrong decision is made, choosing a new tool is time consuming and bad for the business. The purpose of this thesis is to find a good method for choosing an end-to-end testing tool. An overview of the existing methods was given. Primary criteria were selected for choosing potential end-to-end testing tools. Then a popularity metric was applied. After that, three tests were made with selected testing tools and test metrics were applied. Based on results an end-to-end testing tool was selected for the Dashboard web application.

The thesis is in Estonian and contains 40 pages of text, 6 chapters, 11 figures, 11 tables.

Lühendite ja mõistete sõnastik

Graafiline kasutajaliides	*Graphic user interface*, interaktiivne liides, põhineb graafiliste kuvaelementide osutamisel
Kompleksarendaja	*Full-stack developer*, arendaja, kes on võmaline välja töötama süsteemi kõiki kihte või osi
Tagasüsteem	*Backend*, tarkvara süsteem, mida kasutaja ei näe. Töötlev, talletav, käitlev põhiosa.
Prooviserver	*Staging server*, testimis- ja demotarbeline server rakenduse, andmestu, veebilehe vm ajutiseks paigutamiseks enne ta viimist tööserverile
Funktsionaalsus	*Functionality*, tarkvaratoote võime pakkuda spetsifitseeritud tingimustes kasutamisel funktsioone, mis rahuldavad teatatud ja eeldatavaid vajadusi
Veebibrauser	*Web Browser*, programm Interneti kaudu saadud failide tõlgendamiseks ja esituseks, tavaliselt eeldab protokolli HTTP ja HTML-dokumente, võib hõlmata ka muid andmeliike.
Raamistik	*Framework*, stereotüüpne pakett, milles on mudelielemente, mis spetsifitseerivad taaskasutatavat süsteemi(osa) arhitektuuri
Hall kirjandus	*Grey literature*, kindlale kasutajateringile määratud kirjandus, ei levitata raamatukaubanduse kaudu [1]
Teek	*Library*, infoobjektide kogu, üldiseks korduvaks kasutamiseks
Terminal	Antud töös integreeritud arenduskeskkonna osa, mille kaudu sisestatakse käsklusi ning mille kaudu saab infot pärast käskluse käivitamist selle tulemuse kohta.
Veebirakendus	Hajus rakendusprogramm, mis ei sõltu platvormist ja on klient-server-arhitektuuriga ehk kliendiks on veebibrauser ja serveriks on veebiserver. Andmeid hoitakse ja töödeldakse peamiselt serveris ning andmevahetus toimub võrgu kaudu.
Kasutajaliides	*User interface*, mehhanismid inimese interaktsiooniks arvutisüsteemiga
Objektorienteeritud	*Object-oriented*, näiteks objektorienteeritud keeled kasutavad objekt tüüpi andmestruktuure.
Eessüsteem	*Front end*, kliendipoolne, kasutajat või kasutavat süsteemi tagaosaga liidestav süsteem

Tehnilised mõisted on võetud andmekaitse ja infoturbe leksikonist [2].

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

1.1 Probleemi taust ja kirjeldus

Teaduslikul maastikul on levinud teoreetiliste probleemide käsitlemine, kuid tarkvara projektides soovitakse leida viise kuidas testide efektiivsust tõsta läbi praktilise lähenemise. Testide korraldamine ja testide automaatseks tegemine on populaarsed teemad tarkvara tootvas tööstuses [3]. Kahjuks ei rakendata testide automatiseerimist alati adekvaatselt [4].

Automaattestid on oluline osa testimisprotsessis, kuna nii saavad tarkvaraarendajad võimalikult kiiresti tagasisidet, kui midagi on arenduse käigus katki läinud. Tänapäeval on olemas palju erinevaid läbivtestide töövahendeid. Samas on keeruline teha valik vaid kodulehel oleva info põhjal, sest vale otsuse korral on läbivtestide töövahendi vahetamine aega nõudev tegevus, ning selle tõttu ärile kulukas. Aeg-ajalt tehakse selle probleemi lahendamiseks võrdlevaid tabeleid ja tutvustusi [5], [6], [7], [8]. Samas pole kinnitust, et tabeli põhjal tehtud valik on parim.

Lisaks pakutakse järjest uuemaid läbivtestide töövahendeid ning vanadele lisanduvad uued funktsionaalsused ehk olemasolev info aegub. Seega puudub kindel protsess, kuidas läbivtestide tööriistu võiks võrrelda ning kuidas erinevust mõõta. Autor puutus selle probleemiga kokku siis, kui hakkas tööl uut töövahendit otsima veebirakenduse Dashboard läbivtestide tegemiseks. Varem kasutatud läbivtestide tööriista oli vaja välja vahetada, kuid võrdlevad tabelid andsid liiga üldist infot ning oli raske valikut teha. Antud töös võrreldakse potentsiaalseid läbivtestide töövahendeid veebirakendusele Dashboard.

1.2 Ülesande püstitus

Magistritöö eesmärk on luua kasulik meetod läbivtestide töövahendi valimiseks. Vaadatakse üle hetkel olemasolevad võrdlevad meetodid ning valitakse sobilikud Dashboard veebirakenduse jaoks.

Uurimisküsimused:

- Millised on levinumad kriteeriumid automaattestide jaoks loodud töövahendite võrdluses?
- Milliseid testmõõdikuid on rakendatud automaattestide töövahendite võrdluses?
- Kuidas valida sobiv läbivtestide töövahend kriteeriumide ja mõõdikute põhjal?

1.3 Töö struktuur

Magistritöös on kokku 6 peatükki. Esimese peatükis kirjeldatakse tasuta ja probleemi, sõnastatakse eesmärk ja esitatakse uurimisküsimused. Teises peatükis tehakse ülevaade Dashboard veebirakendusest ja seda arendavast tiimist ning tiimi töö põhimõtetest. Kolmandas peatükis tehakse ülevaade automaattestimisest, automaattestide töövahendite kriteeriumitest varasemates võrdlustes ning valitakse sobivad Dashboard veebirakenduse uue läbivtestide töövahendi kriteeriumideks. Samuti tehakse ülevaade testmõõdikutest, mida on rakendatud veebirakenduste automaattestide töövahendite võrdlusel ning valitakse sobivad testmõõdikud Dashboard veebirakendusele. Neljandas peatükis tehakse lühiülevaade potentsiaalsetest läbivtestide töövahenditest ja rakendatakse populaarsuse mõõdikut läbivtestimiseks sobilikele töövahenditele. Viiendas peatükis kirjeldatakse kolme testjuhtumit ning nende põhjal tehakse testid, kasutades välja valitud töövahendeid ning rakendatakse testmõõdikud. Seejärel tehakse kriteeriumite ja mõõdikute tulemuste põhjal võrdlev tabel, mille põhjal langetatakse valik. Seejärel tehakse tulemustest analüüs. Kuuendas peatükis tehakse kokkuvõte magistritöö tulemustest.

2 Dashboard veebirakendus ja seda arendav tiim

Concise Systems OÜ pakub teenust EasyPark Group firmale ehk lühidalt EasyParkile, arendades erinevaid veebirakendusi ja muid süsteeme. Dashboard veebirakendus on üks nendest toodetest.

Parking Dashboard ehk lühidalt Dashboard on loodud parkimisoperaatoritele oma tegevustest ülevaate saamiseks. Seal on võimalik kaardi vaates näha täpsemalt parkimise alasid ning nende kohta käivat infot. Näiteks infot parkimishindadest, -tsoonidest, -piirangutest ja nende kestvusest. Samuti saab seal vaadata aruandeid parkimiste kestvuse ja maksete kohta ning ka üleüldist statistikat parkimise aktiivsuse jms kohta. Kasutajad saavad sisse logida ühe operaatori vaatesse või valida mitme operaatori vahel, kui neile on vastavad õigused antud. Lisaks saab otsida infot konkreetse parkimise kohta või näiteks teatud asukohas olnud parkimiste kohta ja selle info alla laadida. Teenuse klientideks on tavaliselt linnad või eraomanikud, kes soovivad oma infot visuaalsel ja arusaadavamal kujul veebis näha.

Dashboardi veebirakenduse tiim koosneb kolmest Concise Systems firmas töötavast tarkvaraarendajast, ühest Concise Systems firmas töötavast tiimijuhist ja ühest teisest firmast pärit tarkvaraarendajast. Kuna ka teised firmad pakuvad arendusteenust EasyParkile, siis tiimides võib olla ka Concise Systems firmast väljaspool töötavaid inimesi. Tooteomanik on EasyParki töötaja, kes suhtleb klientidega ja teab mis muudatusi nad veebirakenduses juurde vajavad, märgib need üles ja määrab olulisuse. Tiimijuht tegeleb koosolekute korraldamisega, tiimi motiveerimisega ning aitab tiimi liikmetel areneda. Tarkvaraarendajate ülesanneteks on veebirakenduse arendamine ja haldamine – uue funktsionaalsuse lisamine, vigade parandamine, testimine, vajadusel nõuete olulisuse ja täpsustavate aspektide analüüsimine, veebirakenduse seire ja muudatuste avalikustamine. Seda arendusprotsessi ja arendusfilosoofiat kutsutakse ka DevOpsiks [9].

Dashboardi arendav tiim lähtub Superagile raamistiku põhimõtetest, kuid neil on võimalik seda endale mugavamaks kohandada. Tasakaalustatud testimise puhul mainitakse Seleniumi, kuid tiimil on võimalik valida ka teine töövahend, kui see on testimise jaoks parem.

2.1 Läbivtestid ja nende jaoks loodud töövahendid

Testimist saab teha manuaalselt või siis automatiseeritult kasutades selleks sobivat töövahendit. Automaattestimise abil saab samu teste korduvalt jooksutada ning sellepärast on see kiirem ja kasulikum kui manuaalne testimine. Läbivtestid ehk *end-to-end tests* on testid, mille abil on võimalik süsteem tervikuna läbi testida. Kasutades töövahendit, mis võimaldab veebibrauserit automaatselt jooksutada, saab testide abil nupu vajutusi teha ja andmeid sisestada [10]. Läbivtestidega jäljendatakse kasutajate tegevusi võimalikult realistlikult ehk sisselogimisest kuni väljalogimiseni. Testides vajutatakse nuppe ja sisestatakse andmeid läbi kasutajaliidese ning saadakse tagasi vastus prooviserverit jäljendades realistlikke andmeid. Läbi kasutajaliidese on võimalik teha ka muid teste, kuid autor keskendub antud töös läbivtestide jaoks töövahendi leidmisele ning sellepärast kasutab ka termineid nagu näiteks läbivtestide töövahendid mitte kasutajaliidese testimise töövahendid.

Töövahendite all on siin magistritöös mõeldud tarkvara, mis võimaldab veebibrauseri automatiseerimist ning mida kasutatakse läbivtestide tegemiseks. Osasid läbivtestide töövahendeid kutsutakse raamistikuks, kuna neil on mitu teeki ja töövahendit omavahel integreeritud, et selle kasutamine oleks võimalikult mugav. Samas teised töövahendid võimaldavad ainult veebibrauseri automatiseerimist ning vajavad muid testide kirjutamise jaoks vajalikke teeke juurde. Neid töövahendeid kutsutakse vahel teekideks. Mõlemal juhul on need autori jaoks läbivtestimise töövahendid, mis avavad testimise käigus veebibrauseri ning jäljendavad testides kirjutatud käskluste abil nupuvajutust ja andmete sisestamist ning kontrollivad tagasi saadud andmete õigsust.

2.2 Olemasolevad läbivtestid

Olemasolevad läbivtestid on kirjutatud kasutades Selenide. See on Seleniumi ümbris, mis kasutab Selenium WebDriverit. Käsklustel on lihtsam kirjaviis kui Selenium WebDriveril ning sisaldab Ajax tuge käskluste täitmise ootamisel ja automaatseid ekraanitõmmiseid [11]. Algsel tööriista valimisel puudusid tiimiliikmed, kes on nüüd tiimi koosseisus. Samuti ei toimunud algselt tööriista valimisel analüüsi ning pole teada, mille põhjal valik tehti. Ühe koosoleku käigus, kui analüüsiti Superagile raamistiku kasutust, leiti et praegune läbivtestide tegemise töövahend pole tarkvaraarendajate jaoks kõige mugavam. Olemasolev töövahend on Java keeles kirjutamiseks ja paar inimest tiimis töötab

igapäevaselt JavaScript programmeerimiskeeles. Kui funktsionaalsus tegi läbivtestid katki, siis said ainult need tiimiliikmed neid parandada, kes Java programmeerimiskeelt ostasid. See ei tundunud jätkusuutlik tarkvaraarendusviis, sest kõik tiimiliikmed peaksid hoolitsema, et läbivtestid oleks korda tehtud kui nende poolt loodud uus funktsionaalsus need katki teeb. Samuti ei olnud Selenide töövahendi kasutamine eriti mugav. Lisaks oli keeruline erinevaid veebibrausereid testida. Erinevate töövahendite erinevusi oli vaja analüüsida ja leida tiimi jaoks sobivaim töövahend läbitestide tegemiseks.

Tiimis on kõigil arendajatel kohustus kontrollida, et läbivtestid jooksevad sujuvalt peale funktsionaalsuse publitseerimist. Ka peale suurema funktsionaalsuse arendamist, testivad arendajad läbivtestidega oma koodi. Kõik tiimiliikmed peavad oskama neid kirjutada ja hallata. Kuna läbivtestid on kallid, lepiti kokku testjuhtumid ja testide kirjutamine peale raamistiku valimist. Edaspidi uute testide lisamine lepitakse kokku sõltuvalt vajadusest. Peale testide loomist on kõigil tarkvaraarendajatel neid kohustus parandada, kui nende muudatused on läbivtestid katki teinud.

2.3 Superagile raamistiku punktid

Superagile on raamistik [12], mida Dashboardi veebirakendust arendav tiim kasutab. See koosneb mitmest põhimõttest, mis tuuakse välja, et näidata, kuidas jõuti järelduseni, et Dashboardi veebirakendust arendaval tiimil oleks vaja uut testimisvahendit läbivtestide jaoks. Superagile raamistiku punktid on põhimõtted, mida Dashboardi veebirakendust arendav tiim jälgib oma igapäevases töös.

- Kompleksarendajate tiim – tiimi sees on kõik oskused toote tegemiseks olemas. See ei tähenda et kõik on eksperdid kõiges, pigem kõik teavad hetkeolukorda ja mis muutusi on vaja sisse viia. See vajab info ja teadmiste jagamist ning kui see puudub, tuleb see leida väljastpoolt tiimi.

- Tiimi liikmed on teadlikud ärilistest eesmärkidest – arusaamine kasutaja vajadustest ja äri visioonist.
- Prioriteetide muutmine – tiim on valmis prioriteete muutma, kui selline käsk tuleb äripoolelt.

- Pidev publitseerimine ehk inglise keeles *Continuous deployment* – võimalikult kiire muudatuste viimine kasutajateni, et saada tagasisidet. Manuaalse testimisprotsessi puudumise tõttu on arendajad sunnitud kirjutama äriloogikalt põhinevaid automaatteste, et tagada koodi kvaliteet.

- Tüvel põhinev arendus ehk *trunk based development* – väikeste tihedate muutustega on võimalik peatüvesse otse koodi lükata ehk *push.* Samuti käib selle meetodiga kaasas koodi läbivaatus.

- Tasakaalustatud testimine – kõike ei saa 100% testida, sest see läheb kulukaks. Siiski kasutab see raamistik kombinatsiooni testimispüramiidist ja järelevalvest. Tagasüsteemi puhul alustatakse integratsioonitestide kirjutamisest. Testitakse kasutuslugu ilma erandjuhtudeta. Ülejäänud loogika testimiseks kasutatakse üksuste teste. Eesüsteemide puhul testitakse üksuste testidega React komponente. Seleniumi kasutades asendatakse manuaalne testimine automaatsete läbivtestidega. Kaetakse ära ainult kõige olulisemad kasutuslood. Vead mis läbi lipsavad testimisest saab kinni püüda kasutades head järelevalvesüsteemi nt Sentry.

- Mikroteenused – firmasiseselt lihtsam hallata, kuna saab mikroteenused tiimide vahel ära jaotada nii et kohustus jaguneks.

- Automatiseerimine – oluline on võimalikult palju protsesse automeerida, et säästa pikas plaanis arendajate aega.

- MVP mõtlemine – MVP ehk *minimal viable product*. Võimalikult kiirelt väärtuse toomine kliendini. Ning ülejäänu lisamine hiljem.

- Lühike suhtlusvoog – võimalikult otsene suhtlus äripoolega.

- Autonoomia – kui tiim sõltub võimalikult vähe teistest väljaspool tiimi, saab tööd efektselt ja kiirelt teha.

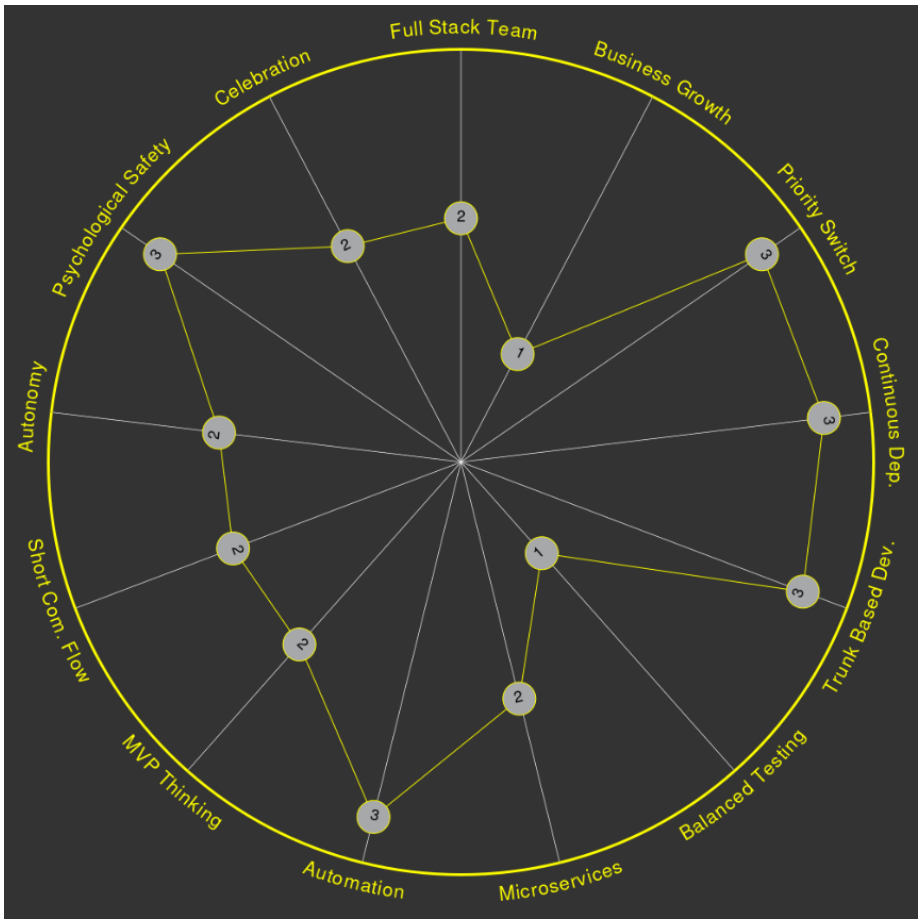
- Psühholoogiline turvalisus – tiimil on võimalik ideid ja küsimusi arutada nii, et kõik suhtuvad nendesse lugupidavalt.

- Tähistamine – tänu näitamine aitab teistel tunda, et on väärtuslikud ja kasulikud.

2.4 Superagile raamistiku kasutamine

Aeg-ajalt analüüsivad Concise Systems firmas töötavad tiimid raamistiku kasutust oma töös. Seda tegi ka Dashboard veebirakendust arendav tiim ning kasutas selleks Superagile mobiilirakendust, kus kogu tiim individuaalselt andis igale teemale punktid nullist kolmeni, sõltuvalt sellest kui hästi tiim seda teemat oma igapäevases töös rakendab. Peale hindamist analüüsiti võimalusi, kuidas madalamade hinnetega punke parandada. Joonisel 1 on kujutatud tulemuste kaarti, kus on ringi servale kirjutatud 2.3 peatükis olevad punktid ning ringi sees on igale punktile antud number. Numbrid on ühendatud omavahel ning nendest on moodustatud kaart. Joonisel l on näha, et Dashboardi tiimi jaoks on tasakaalukas testimine hinnatud madala hindega ehk 1 ja hakati arutama, kuidas saaks seda parandada.

Tasakaalustatud testimise punktis oli välja toodud Selenium töövahend läbivtestide jaoks. Siiski ei olnud Dashboard veebirakendust arendav tiim kindel, kas see töövahend nende jaoks on parim valik. Hetkel kasutab tiim Selenide töövahendit, millega tehakse teste Java programeerimiskeeles. Arutelu käigus tuli välja, et see töövahend pole kõige sobivam, sest osad tarkvaraarendajad oskavad ainult JavaScripti programeerimiskeelt ning ei soovi Javat kasutada testide tegemisel. See tegi kogu arendusprotsessi teistele tarkvaraarendajatele ebamugavaks, sest need kes Java programeerimiskeelt oskasid, pidid parandama ja uuendama läbivteste ka siis, kui nende muudatused testide katkiminekut ei põhjustanud. Selle tõttu vajas Dashboard veebirakendust arendav tiim analüüsi, et selgitada välja läbivtestide töövahendite erinevused ja võimalused, et valida uus töövahend. Samuti tuli arutelu käigus välja, et oli probleemi ka eessüsteemi üksuste testimisega ja eessüsteemi integratsiooni testidega. Nende tõttu oli samuti tasakaalustatud testimise punkt Dashboard tiimi poolt hinnatud madala hindega, kuid seda töö ei käsitle.



Joonis 1. Tulemuste kaart.

3 Metoodika

Käesolevas peatükis on ülevaade automaattestimisest. Samuti ka Dashboard tiimi nõuded läbivtestide kirjutamiseks sobivatele töövahendile, lisaks ka varasemad tehtud võrdlused ning mõõdikud, mille põhjal saaks hinnata läbivtestide töövahendite erinevusi. Võrdlemiseks on enamasti kasutatud tabeleid, kuid on kasutatud ka mõõdikuid, et valik langetada [13].

3.1 Automaattestimine

Automaattestimine on oluline osa pidevast tootearendusest ja toote uuenduste avalikuks tegemisest. Tarkvaraarendus ei ole odav ning seetõttu üritatakse võimalikult palju tarkvaraarenduses olevaid tegevusi automatiseerida. Samuti võimaldab see kiirema toote uuendamise klientide jaoks.

Nagu 2.1 peatükis mainiti, on läbivtestimine ehk *end-to-end testing* kogu süsteemi korraga testimine. Selliste testide tegemine ja koostamine on ajakulukas, ning sellepärast peaks neid olema võrreldes teiste automaattestidega vähem. Testimispüramiidi järgi [14], peaks läbi kasutajaliidese olema vähem testimist, integreeritud testimist sellest rohkem ning üksuste testimist kõige rohkem. Testimispüramiidi joonistatakse tavalise püramiidina ja jaotatakse kolmeks kihiks. Alumisel kihil on üksuste testimine, keskmisel kihil on integreeritud testimine ja ülemisel kihil on kasutajaliidese testimine.

3.1.1 Üksuste testimine

Üksuste testimise ehk täpsemalt komponendi osade testimise mõistet kasutatakse sõltuvalt testitavast süsteemist. See võib tähendada mingi funktsionaalsuse testimist või näiteks objektorienteeritud keeles võib üksus tähendada ka meetodit või kogu klassi [10]. Kõige olulisem on nõutud funktsionaalsuse testimine ning selle jaoks jäljendatakse andmeid, mida funktsionaalsus kasutab.

Üksuste testimist tehakse testimispüramiidi kohaselt kõige rohkem, kuna neid on lihtne lisada, hooldada ja parandada [14]. Nendeks tegevusteks minev ajakulu on väiksem kui teiste automaattestide puhul. Üksuste testimist saab teha nii eessüsteemile kui ka tagasüsteemile.

3.1.2 Integratsiooni testimine

Integratsioonitestide puhul testitakse süsteemi jaoks vajalikke osi, mille info tuleb mujalt [10]. Näiteks testitakse veebirakenduse puhul integratsiooni andmebaasiga.

Integratsiooniteste tehakse vähem kui üksuste teste, kuna mujalt tuleva info ootamine testis võtab kauem aega ning sellepärast on integratsioonitestide teostamine aeglasem [10]. Integratsiooniteste saab samuti teha nii eessüsteemile kui ka tagasüsteemile,

3.1.3 Kasutajaliidese testimine

Kõige vähem tehakse testimispüramiidi meetodi järgi automatiseeritud teste läbi kasutajaliidese. Läbi kasutajaliidese on võimalik teha kasutajaliidese teste ehk *user interface tests* ja ka läbivteste. Kasutajaliidese terminit kasutatakse vahel segamini graafilise kasutajaliidese terminiga, kuid kasutajaliideseks saab pidada ka terminali ehk käsurea liidest, vormi ja menüü kujul liideseid ning ka naturaalkeelega suhtlust võimaldavat liidest [15]. Kasutaja interaktsioon liidesega nagu näiteks nupu vajutus peaks kasutajaliideses protsessi käivitama ning see peaks olema kasutajale nähtav.

Kuigi vahel ei tehta läbivtestide ja kasutajaliidese testide puhul vahet, siis tegelikult neil siiski on erinevus [10]. Kasutajaliidese testide puhul ei pea olema info mis interaktisooni käigus saadakse realistlik ning seda võib imiteerida. Läbivtestide puhul soovitakse saada kasutajaliidese interaktsioonist tulev info võimalikult realistliku protsessi käigus. Näiteks tehakse prooviserver, kus on tehakse testimiseks andmed ja kasutajad, mis on sarnased päris tootele.

Kuigi läbivtestimist saab teha ilma graafilise kasutajaliideseta [10], keskendub antud töö graafilise kasutajaliidesega läbivtestimisele. Veebibrausereid saab automatiseerida testides ka ilma graafilise kasutajaliideseta. Sellist veebibrauserit kutsutakse sel juhul *headless* veebibrauseriks [10].

3.2 Tarkvara mõõdikud

Tarkvara mõõdikutega saadakse mõõtmise käigus väärtused, mida kasutatakse projektide hindamiseks, lisaks ka progressi ja soorituse mõõtmiseks. Mõõdikud hõlmavad kõiki kalkulatsioonidest tulenevaid mõõtmisi, mis tehakse ükskõik millisele tarkvara osale [16].

Mõõdikute tulemuste abil on võimalik saada täpsemat infot toote, selle protsessi ja projekti kohta. Samuti on võimalik mõõdikute abil selle infoga seotud tähtaegu ja ajalisi eesmärke hinnata ning anda ennustusi. Mõõdikute põhjal saab ka infot riskidest ja probleemidest, mida parendada mõõdetavas tootes, protsessis või projektis [17]. See info on oluline tarkvara toote, protsessi ja projekti juhtimisel.

3.2.1 Testmõõdikute eesmärk ja jaotumine

Testmõõdikute puhul on oluline kogu testimise vältel neid jälgida ja nende põhjal leida üles kohad, mida saaks projektis parandada [16]. Nende kasutamise eesmärk on parandada toote kvaliteeti ja teenuste produktiivsust, samuti saada konkreetsem ülevaade olukorrast tänu numbrilistele väärtustele ja jälgida tootega seotud protsesside edukust [18].

Testmõõdikuid on palju ja neid on ka erinevalt kategooriatesse jaotatud olenevalt autorist. Üldiselt jaotatakse [17] need kaheks kategooriaks – toote mõõdikud ja protsessi mõõdikud. Protsessi mõõdikud annavad teavet testide ettevalmistamise, nende läbiviimise ja ka edu kohta. Toote mõõdikud annavad infot testide oleku ja toote testimise seisundi kohta ning see info kujuneb testimise käigus. Toote mõõdikud annavad aimu toote kvaliteedi kohta, samas kui protsessi mõõdikud näitavad toodetava tarkvara protsesside kvaliteeti.

Protsessi mõõdikutega mõõdetakse näiteks mitu testjuhtumit on kirjutatud, mitu testjuhtumit on testide poolt teostatud. Samuti antakse infot edasi mitu protsenti testjuhtumitest teostati edukalt, mitu protsenti läks katki ja mitu protsenti üleüldiselt teostati. Veel kuulub protsessi mõõdikute alla näiteks testjuhtumi teostamise keskmise aja mõõtmine [17].

Toote mõõdikutega mõõdetakse näiteks testimiseks arvestatavat ajakulu, tegelikku testimise aega ja testide katkiminemise aja möödumist eelmises korrast. Lisaks mõõdetakse ka mitu korda testid lähevad katki kindlas ajavahemikus [17].

Mõõdikuid on jaotatud ka teisiti [18]. Need jaotati manuaalseteks testmõõdikuteks, soorituse testimise mõõdikuteks ja automatiseerituse testmõõdikuteks.
Manuaalsete testmõõdikutega mõõdeti näiteks testjuhtumi kirjutamise produktiivsust, mitu vigast testi arvestatakse päriselt katkiseks ja mitu vigast testi annab vale infot ja uuesti testi toestades on see korras. Soorituse testimise mõõdikutega mõõdeti peamiselt eessüsteemi ja tagasüsteemi võimekust testimise käigus. Automatiseerituse testmõõdikutega mõõdeti näiteks mitu manuaaltesti on automatiseeritud ja lisaks automatiseeritud testjuhtumi teostamise produktiivsust.

3.2.2 Testmõõdikute määramise protsess

Allpool olev tabel 1 on näitena välja toodud, kuidas on võimalik kirjeldada testmõõdikute tegemise protsessi [18]. Esimene sammuna protsessis tehti analüüs, kus vaadati üle kõik mõõdikud, mida on võimalik rakendada ja valiti sobivad. Järgmiseks sammuks oli kommunikatsioon, kus selgitati mõõdikute eesmärki ja olulisust teistele projektis olijatele. Kolmandaks sammuks oli hindamine ehk saadud andmete korrektsuse kontrollimine ja andmete põhjal mõõdikute väärtuste arvutamine. Viimase sammuna aruanded tulemuste põhjal ja edastati vajalikele osapooltele ja saadi tagasiside.

Tabel 1. Testmõõdikute elutsükli näide [18]

Analüüs	Mõõdikute valik ja defineerimine Valitud mõõdikute parameetrite defineerimine
Kommunikatsioon	Mõõdiku vajaduse selgitamine teistele Teistele mõõdiku andmete olulisuse selgitamine
Hindamine	Andmete saamine ja õigsuse kontrollimine Andmete põhjal mõõdikute väärtuste arvutamine
Aruandlus	Aruande koostamine saadud tulemuste põhjal Aruande edastamine ja tagasiside saamine

3.2.3 Testide töövahendite võrdlemiseks kasutatud mõõdikud

Leidub paar testide automatiseerimise töövahendite võrdlust, kus on kasutatud mõõdikuid. Näiteks on varem võetud mõõdikuteks testide teostamise aeg, nende efektiivsus ja edukus lähtuvalt testi nõuetest [13], [19]. Tabelis 2 on kolm mõõdikut eespool kirjeldatu kohta ning ka nende seletused.

Tabel 2. Töövahendite võrdlemiseks kasutatud mõõdikud

Nimi	Seletus
Testide teostamise aeg	Mõõdetakse iga testi teostamise aega
Testide efektiivsus	Kirjutatud koodiridade arv
Edukus lähtuvalt testi nõuetest	Mitu nõuet on testide poolt kaetud

- Testide teostamise aja ehk *test execution time* mõõdiku all on mõeldud aega, mis kulub testi käimapanekust kuni testi lõppemiseni. Testide aega mõõdetakse tavaliselt sekundites või millisekundites.

- Testide efektiivsuse ehk *test efficiency* mõõdikuga loetakse kokku koodiread iga testi puhul [13]. Nii mõõdeti kui palju vaeva nõudis ühe testi kirjutamine. Kui ühes testis on võrreldes teise testiga vähem koodiridu, siis esimese testi puhul nähti vähem vaeva.

- Edukus lähtuvalt testi nõuetest ehk *requirement-based test coverage* mõõdikuga vaadatakse üle testides olevad nõuded ning kas töövahendi kasutamisel oli võimalik teha testid nii, nagu oli plaanitud.

3.2.4 Testmõõdikud Dashboard veebirakendusele

3.1.2 peatüki näite põhjal on antud töös testmõõdikute tegemise protsess järgmine:

1. Analüüs – valitakse sobivad mõõdikud läbivtestide töövahendite võrdlusesse
2. Hindamine – rakendatakse mõõdikud ja saadakse tulemused
3. Aruandlus – järelduste tegemine mõõdikute tulemuste põhjal

Testmõõdikuid rakendatakse kõigile läbivtestidele, mis kirjutatakse valitud veebibrauseri automatiseerimist võimaldavate töövahenditega. Läbivtestid kirjutatakse kolme

testjuhtumi põhjal. Testmõõdikuid rakendatakse et leida, kas ja kui palju võib töövahendist sõltuvalt läbivtestide tulemus oleneda. Tulemuste põhjal võrreldakse nende ja kriteeriumide olulisust.

Autori jaoks on kõik 3.1.2 peatükis välja toodud testmõõdikud olulised läbivtestide töövahendi valimisel. Samuti soovis autor täiendada testide teostamise aja mõõdikut. Seda mõõdikut saab rakendada erinevates veebibrauserites ning tulemusi võrrelda [13]. Autor võttis testmõõdikuteks testide teostamise aeg sõltuvalt veebibrauserist mõõdiku, edukus lähtuvalt testi nõuetest mõõdiku ja testide efektiivsuse mõõdiku.

3.3 Võrdlemine kriteeriumite alusel

Automaattestimise töövahendite võrdluses on toodud välja erinevaid kriteeriume.
Tabelis 3 on näide kriteeriumite jaotusest, mida rakendati automaattestide töövahendite võrdluses [5]. Võrreldi töövahendeid, mis võimaldavad automaatselt operatsioone veebibrauseris teha. Seal oli parameetrite jaotatud nii: kasutusse võtmise lihtsus, kirjutamise ja aruandluse võimalused, töövahendi kasutatavus. Parameetritele järgnes kriteeriumite tulp ehk kasutusse võtmise parameeter jagunes litsentsi hinna ja kasutuse lihtsuse kriteeriumiteks. Kirjutamise ja aruandluse võimaluste parameeter jagunes skripti kirjutamise aja, skripti keele, objekti tuvastuse, õppimise aja, skripti teostamise kiiruse, raamistiku ja pideva integratsiooni kriteeriumideks. Töövahendi kasutatavuse parameeter jagunes veebibrauserita rakenduste, operatsioonisüsteemi, veebibrauseri ja seadme toe kriteeriumiks.

Teises automaattestimise töövahendite võrdluses on lisatud eelmainitud kriteeriumitele lisaks ka skripti jooksmise jäädvustamine ja andmepõhisus ehk väliste andmetele ligipääsetavus, näiteks Excel tabelite puhul. Välja toodi veel töövahendi funktsioon ehk millist tüüpi testimist toetatakse ja muu rida, kus on olulised asjad märgitud, millel konkreetset kriteeriumit ei olnud [20].

Eelnevalt välja toodud kriteeriume on veel mainitud [8]. Kriteeriumiteks võeti seal võrdluses platvormi kasutus, milliste rakenduste jaoks saab töövahendiga teste kirjutada, testide kirjutamisel kasutatavad keeled, õppimise keerukus, paigaldus ja kasutus, testide

tegemiseks minev ajakulu, objektide hoidla, pildi põhjal testimine, toote toetus, pidev integratsioon.

Tabel 3. Parameetrite ja kriteeriumite jaotus [5]

Parameeter	Kriteerium
Kasutusse võtmise lihtsus	Litsentsi hind
	Toe võimalused
Kirjutamise ja aruandluse võimalused	Skripti tegemise aeg
	Skripti keel
	Objekti tuvastus
	Õppimise aeg
	Skripti (teostamise) aeg
	Raamistik
	Pidev integratsioon
Töövahendi kasutatavus	Rakenduste (veebirakendusi välja arvates) tugi
	Operatsioonisüsteemi tugi
	Veebibrauseri tugi
	Seadme tugi

On leitud, et halli kirjanduse põhjal on peamised kriteeriumid automaattestide töövahendite võrdluses järgmised [7]:

- Skripti keeled
- Integreerimisvõimalus
- Tehniline tugi
- Skripti jooksmise jäädvustamine
- Elemendi tuvastus
- Kasutatavus

- Hind

- Mobiiliseadme tugi

Kriteeriumite hindamine olenes allikast ning mingitel kriteeriumitel olid numbrilised hinnangud samas kui teistes allikates hinnati vähem/rohkem stiilis [7].

Eelnevate võrdluste põhjal teeb autor mainitud kriteeriumitest ülevaade ning lisab ka nende selgituse. Selgitused on kirja pandud autori arusaama järgi, kuna kõikides allikates pole kriteeriumitele definitsioone antud:

- Litsentsi hind – Töövahendi kasutamise loa saamine rahalise maksega.

- Toe võimalused/tehniline tugi – Tehniliste probleemide korral abi saamine läbi virtuaalkogukonnast või töövahendi tegijatelt. Samuti on oluline hea dokumentatsioon.

- Skripti tegemise aeg – Ajakulu, mis läheb testide tegemisele.

- Objekti tuvastus/ objektide hoidla – Objekti tuvastuse all on mõeldud elementide leidmist testitavas rakenduses töövahendi poolt. Näiteks võrreldi töövahendites olevat funktsionaalsust, mis leiab veebilehel olevad elemendid, et neid testimisel kasutada [5]. Neid elemente kasutatakse skripti teostamise jäädvustamist võimaldavate testide tegemisel. Sellised testid ei vaja programmeerimisoskust ning nendes testides näidatakse töövahendile ette elemendid ja kuidas neid kasutada ning salvestatakse elementide kasutamine läbi kasutajaliidese. Objektide hoidla all ilmselt mõeldakse tehnoloogiat, kuidas testid leiavad veebilehel olevaid elemente [8].

- Õppimise aeg/õppimise keerukus/kasutatavus – Töövahendi õppimiseks kuluv aeg kiire/aeglane väärtustega või siis õppimise keerukus keskmine/kõrge

väärtustega. Kasutatavuse [7] definitsiooni välja ei toodud, kuid autori arvates see põhimõte sarnaneb õppimise keerukusega.

- Skripti (teostamise) aeg – Aeg, mis kulub testi käimapanekust kuni testi lõpetamiseni.

- Raamistik – Võrreldi kõiki teeke ja töövahendeid, mis on raamistikku integreeritud.

- Pidev integratsioon – Võrdlusesse toodi välja võimalusi, millega saab töövahenditega tehtud teste pidevasse integratsiooni kaasata.

- Ilma veebibrauseriteta rakenduste tugi/mitut erinevat tüüpi rakendusi saab testida – Kas on võimalik nende töövahenditega kirjutada automatiseeritud teste rakendustele, mis veebibrauserit ei kasuta.

- Operatsioonisüsteemi tugi – Kõik operatsioonisüsteemid, mille peal on võimalik võrdluses olevaid töövahendeid kasutada.

- Veebibrauseri tugi – Kõik veebibrauserid, millega võrdluses olevaid töövahendid võimaldavad teste jooksutada.

- Seadme tugi/mobiiliseadme tugi – Kriteerium, mis võrdleb töövahendite võimalusi automaattestide tegemiseks mobiiliseadmetele.

- Skripti teostamise jäädvustamine – Ehk *Record and Playback*, mille kasutust oli mainitud ka kriteeriumis objekti tuvastus/objektide hoidla.

- Millist tüüpi testimist toetatakse – Milliseid teste on võimalik teha võrdluses olevate töövahenditega.

- Muu – Eeliste ja puuduste ning muude kommentaaride jaoks tehtud veerg.

- Pildi põhjal testimine – Selle meetodi puhul leiab töövahend kasutajaliidese elemendid kasutades pildi põhjal olevate pikslite tuvastust [21].

- Programmeerimise oskuse vajamine – Kriteerium, mis on loodud selleks et välja selgitada kas automaattestide tegemist võimaldav töövahend vajab kasutamiseks eelnevat testide kirjutamise oskust.

- Tulemuste esitamine/aruandluse genereerimine – Kuidas testide tulemused on töövahendi kasutajale esitatud.

- Seadistamine – Automaattestide tegemist võimaldava töövahendi seadistamise keerukus.

- Integreerimisvõimalus – Võrdluses olevate töövahendite integreerimise keerukus.

- Välistele andmetele ligipääsetavus – Töövahendis olevad võimalused välistest tekstidokumentides ja muudest allikatest testidesse andmete lisamine.

Eraldi mainitud parameetreid, mida on näha tabelis 3, mujal võrdlustes ei leidunud. Kõigis eelnevalt mainitud võrdluses oli kriteeriumiteks skripti keel ja õppimise aeg või keerukus. Kolmes võrdluses oli mainitud tehnilise toe võimalusi, hinda, objekti tuvastust ja operatsioonisüsteemi tuge. Paljude kriteeriumite puhul olid väärtused rohkem/vähem stiilis ehk need võisid sõltuda autori hinnangust. Nendeks kriteeriumideks on näiteks skripti tegemise aeg, õppimise keerukus ja seadistamine.

3.3.1 Nõuded läbivtestide töövahendile Dashboard veebirakenduse näitel

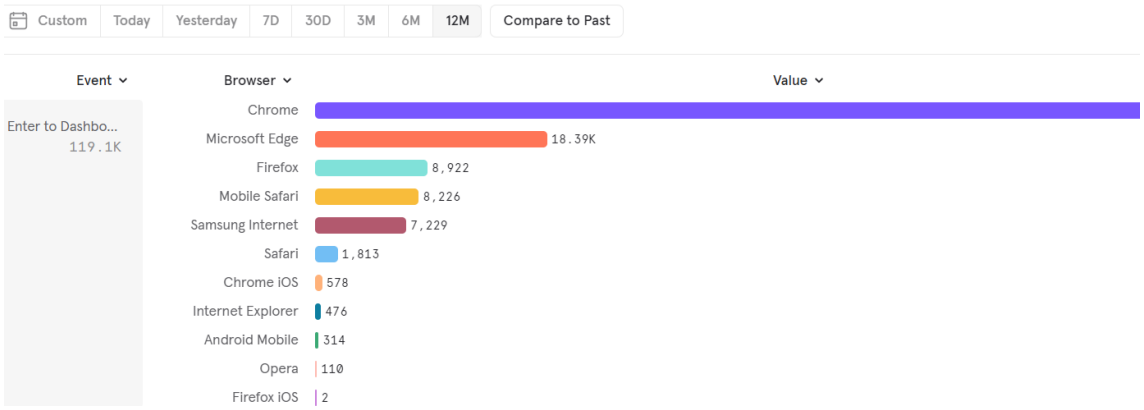
Autor analüüsib eelnevas peatükis välja toodud kriteeriume ja valib kõige sobivamad läbivtestide töövahendite võrdluse jaoks. Samuti lisab kriteeriume juurde vastavalt vajadusele. Seejärel jagab kriteeriumid kahte kategooriasse. Nendeks on esialgsed kriteeriumid ja täpsemad kriteeriumid.

- Skripti keeled – See on Dashboard tiimi jaoks oluline kriteerium, sest eelistatakse JavaScripti, et kogu tiim saaks teste hallata.

- Operatsioonisüsteemi tugi – Windows 7 ja sellest uuemad versioonid ning lisaks Linux Ubuntu 12.04 ja sellest uuemad versioonid peaksid olema toetatud.

- Hind – Kõik kasutusele võetavad raamistikud võiks olla eelistatult tasuta saadaval.

- Erinevate veebibrauserite tugi – Kasutajate poolt kasutatavad veebibrauserid on välja toodud joonisel 2. Info saamiseks on kasutatud Mixpanelit [22]. Kõige olulisem on ära testida kolm kõige populaarsemat veebibrauserit. Kõige rohkem on kasutajate poolt Dashboard veebirakendust külastatud kasutades Chrome, Microsoft Edge ja Firefox veebibrauserit.



Joonis 2. Veebibrauserid, mida kasutatakse Dashboardi veebirakenduse külastamisel.

- Õppimise keerukus – See on mitut moodi tõlgendatav termin aga autor leiab, et kui on piisavalt hea dokumentatsioon, siis on ka vähem vaja abi otsida foorumitest. Samas kui läbivtestide töövahend on populaarne, siis leiab palju infot ka foorumitest või muudest allikatest.

- Paigaldus – Autor jaoks on paigaldus ajakulu, mida tuleb arvesse võtta. Kuigi testimise käigus tehakse paigaldus juba ära, on hea teada selle keerukust. Näiteks soovitakse tulevikus paigaldada sama töövahend mingile teisele veebirakendusele.

- Muu – See on võrdlevas tabelis veerg, kuhu autor lisab spetsiifilised erinevused, mida otseselt võrrelda ei saa vaid annavad lisainfot.

- Graafilise kasutajaliidese kasutamine – 3.1.3 peatükis mainiti, et veebibrauseri automatiseerimist saab teha ilma graafilise kasutajaliideseta. Autor eelistab siiski brauseri automatiseerimist testides läbi graafilise kasutajaliidese, kuna nii on parem testjuhtumi samme jälgida.

- Sobivus Dashboard veebirakenduse teekidega – Töövahend peaks võimaldama läbivtestide tegemist, kui veebirakendus on kirjutatud kasutades JavaScript teeki Reactjs. Autor toob selle kriteeriumi välja, kuna eksisteerib läbivtestide töövahendeid, mis on loodud konkreetse JavaScript raamistikuga tehtud veebirakenduse jaoks. Üheks selliseks töövahendiks on Protractor [23].

- Testi teostamise aeg – eelnevalt sõnastatud kui skripti teostamise aeg. Selle mõõtmiseks kasutab autor testmõõdikut peatükis 5.2.

Esialgsete kriteeriumite alla lähevad järgmised nõuded:

- Hind
- Skripti keeled
- Graafilise kasutajaliidese kasutamine
- Sobivus Dashboard veebirakenduse teekidega
- Operatsioonisüsteemi tugi
- Microsoft Edge, Chrome ja Firefox veebibrauserite tugi

Täpsemad kriteeriumid võrdlemiseks tähtsusjärjekorras:

- Õppimise keerukus
- Testi teostamise aeg
- Seadistamine
- Muu

Autor jagas kriteeriumid kaheks grupiks, kuna mingid kriteeriumid on kindla väärtusega, ning ei ole mõtet võrrelda läbivtestide töövahendeid, mis on näiteks tasulised, kuna neid ei saaks kasutada ja proovida. Esialgsed kriteeriumid on kindla väärtusega ehk hind peaks olema tasuta, skripti keel ehk testide tegemise keel peaks olema JavaScript ning läbitestimine peaks käima läbi graafilise kasutajaliidese.

Skriptide ehk testide tegemise aega on raske hinnata, sest see sõltub paljudest erinevatest teguritest. Autori jaoks objekti/elemendi tuvastus ja skripti ehk testi teostamise jäädvustamine pole oluline, kuna võrdleb töövahendeid, kus testid on koodi kujul ning elemendi tuvastus toimub samuti töövahendi poolt. Programmeerimise oskuse vajamise kriteeriumit autor samuti ei vajanud. Raamistiku kriteerium annab täpsema ülevaate töövahendi kõikides teekidest ja muudest raamistikest. See on lisainfo, mida näidatakse võrdlevas tabelis kriteeriumi all muu. Pideva integratsiooni kriteerium annab samuti võimalustest ülevaate, kuid autori arvates on tänapäeval see muutunud juba standardiks, et töövahendid sellega arvestavad.

Integreerimisvõimaluse kriteerium oli autori jaoks liiga üldine ning selletõttu on raske seda hinnata. Tulemuste esitamise kriteeriumi arutati kogu Dashboard tiimiga, kuid sellele oli raske olulisust määrata. Kuna on palju erinevaid tulemuste töölaudu ja muid teeke on võimalik juurde integreerida, jäi see kriteerium välja.

Ülejäänud kriteeriumid, mida varasemates võrdlustes on mainitud ehk ilma veebibrauserita rakenduse tugi, mobiilseadme tugi, mis tüüpi testimist toetatakse, pildi põhjal testimine, välistele andmetele ligipääsetavus – need ei ole vajalikud antud töös edasi uurimiseks.

4 Potentsiaalsed töövahendid Dashboard veebirakenduse läbivtestimiseks

Kuna automaattestide töövahendite turul on väga palju valikut, siis rakendas autor peatüki 3.2.1 põhjal esialgseid kriteeriumeid, et leida läbivtestide jaoks potentsiaalsed töövahendid, mida lähemalt võrrelda. Nende kriteeriumite põhjal leitud töövahenditele rakendab autor populaarsuse mõõdikut, kuna populaarsemate töövahendite dokumentatsiooni uuendatakse tihemini ning on ka tõenäolisem, et seda arendatakse tulevikus edasi [24], [25].

4.1 Esialgse valiku tegemine

Läbivtestidele sobilike töövahendite info leidis autor peamiselt hallist kirjandusest, samuti oli paarist sobilikust töövahendist infot ka teaduslikus kirjanduses [5], [13]. Esialgsed kriteeriumid olid tähtsuse poolest võrdsed ehk see, et läbivtestide töövahend oleks tasuta oli sama oluline, kui see, et testide keeleks on JavaScript. Põhjus oli selles, et kui ühe või teise nõudega ei arvestatud, polnud võimalik neid töövahendeid edasi uurida ja nendega teste teha. Esialgseteks kriteeriumideks oli:

- Tasuta saadavus
- Teste saab JavaScript keeles kirjutada
- Läbi graafilise kasutajaliidese testide teostamine
- Sobivus Dashboard veebirakenduse teekidega
- Operatsioonisüsteemid Windows 7 ja Linux Ubuntu 12.04 ja nende uuemad versioonid
- Microsoft Edge, Chrome ja Firefox veebibrauserite tugi

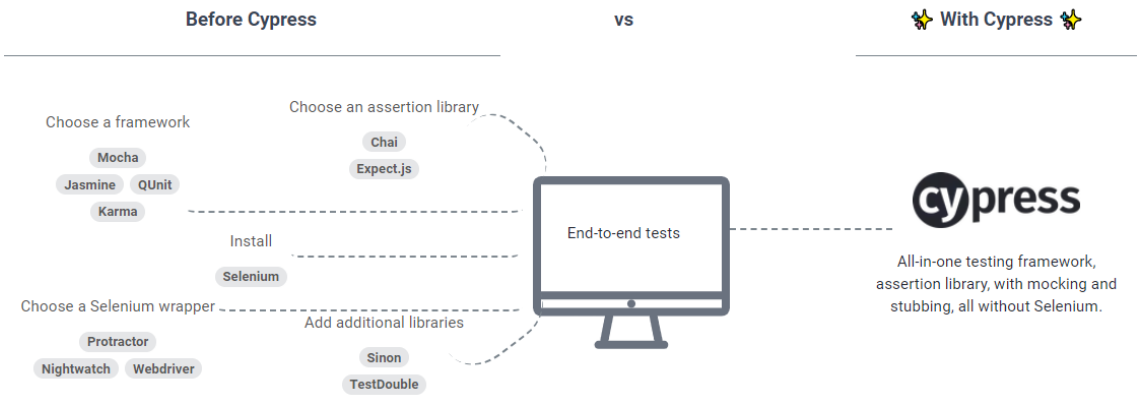
4.2 Esialgsetele kriteeriumitele vastavad töövahendid

Käesolevas peatükis tehakse lühiülevaate esialgsetele kriteeriumitele vastanud töövahenditest ning võrdlev tabel.

4.2.1 Cypress

Cypress kirjeldab ennast kui kõik ühes testimisraamistik, ning teeb veebibrauseri automatiseerimist ilma Seleniumi WebDriverit kasutamata [26]. Joonis kolmel on näiteks Selenium töövahend, mis vajab ka muid teeke ja ka raamistikku, ning paremal pool Cypress töövahend, mis on raamistik ja sisaldab kõiki vajalikke teeke.

Joonis kolmel üleval kirjas on *Before Cypress vs With Cypress* ehk enne Cypress raamistikku versus Cypress raamistikuga. *Before Cypress* all on kirjas *Choose a framework* ehk raamistiku valimine, kus on välja toodud erinevad testimise raamistikud ehk Mocha, Jasmine, QUnit, Karma. Samuti on *Before Cypress* all kirjas *Choose an assertion library* ehk kinnituse teegi valimine, kus on toodud näitena Chai ja Expect.js. Lisaks on seal välja toodud Seleniumi installeerimise samm ning *Selenium wrapper* ehk Selenium ümbrise valimine, mis on näiteks Protractor, Nightwatch, Webdriver (ilmselt on mõeldud WebdriverIO töövahendit). Lisaks saab veel teeke juurde lisada ehk *Add additional libraries* all on töötud näitena välja Sinon ja TestDouble. Pildi keskel on monitor kuhu on kirjutatud *End-to-end tests* ehk läbivtestid ning paremal poole monitori on kirjas Cypress ja selle all on kirjas, et see on kõik ühes testimise raamistik, kus on olemas kinnituse teek ning Seleniumit ei kasutata.



Joonis 3. Seleniumi töövahendi ja Cypress töövahendi võrdlus [26]

4.2.2 Selenium WebDriver

Selenium WebDriver on veebibrauseri automatiseerimisvahend, mis kuulub Seleniumi testimissüsteemi, kus on veel Selenium IDE ja Selenium Grid töövahendid [27]. Testide tegemisel saab juurde lisada Selenium WebDriverile veel teisi töövahendeid. Sellepärast nimetatakse seda mõnikord teegiks, mitte raamistikuks. Näiteks sinna saab olenevalt juurde lisada kinnituse teegi nagu joonisel 3 on välja toodud. Kinnituse teek teeb lihtsamaks käskude kirjutamise, et kontrollida kas tulemus vastab oodatud tulemusele. Samuti on vaja ka sellist raamistikku, mis testide teostamise käsklust ja testide kirjapilti lihtsustab ning ka aruandeid genereerib.

4.2.3 WebdriverIO

See raamistik kasutab ka veebibrauserite automatiseerimise lahendust [6]. Joonis kolmel on näidatud, et WebdriverIO on kui Seleniumi ümbris, mis kasutab Seleniumi WebDriveri veebibrauserite automatiseerimise funktsionaalsust.

Ümbris ehk *wrapper* on midagi, mis kasutab teist olemit, võttes ära selle keerukuse ja tehes seda kasutajale lihtsamaks. Autor liigitab ka sellised ümbrised läbivtestide töövahendite alla, kuna need on lihtsalt raamisikud, mis kasutavad Selenium Webdriveri töövahendi funktsionaalsust. WebdriverIO töövahendi esimesel installeerimisel oma projekti, käivitub terminalis seadistamise abistaja, mis küsib abistavaid küsimusi ja millega saab lisaks installeerida abistavaid teeke. Selle protsessi käigus saab valida, millist kinnituse teeki ja testimise raamistikku lisada.

4.2.4 Nightwatch

Nightwatch kirjeldab ennast kui integreeritud raamistik, mis on loodud läbivtestimiseks ja kasutab veebibrauserite automatiseerimiseks Selenium Webdriveri poolt pakutavat funktsionaalsust [28].

4.2.5 Playwright

Playwright võimaldab veebirakenduste läbivtestimist. See töövahend ei kasuta Selenium Webdriveri funktsionaalsust vaid nende poolt loodud funktsionaalsust veebibrauseri kontrollimiseks ja kasutamiseks [29].

4.2.6 TestCafe

TestCafe ei kasuta Selenium WebDriveri lahendust, vaid kasutab läbivtestide teostamiseks lokaalses masinas olevaid veebibrausereid. TestCafel on kinnituse teegid ja muud vajalikud teegid raamistikku sisse ehitatud [30].

4.2.7 Esialgsete kriteeriumidega võrdlev tabel

Autor lisas kõik eespool alampeatükkides välja toodud läbivtestide töövahendid võrdlevasse tabelisse. Tabel 5 võrdluse põhjal on kõik läbivtestide töövahendid tasuta ning võimaldavad läbi graafilise kasutajaliidese testide teostamist. Samuti on kõigi töövahenditega võimalik teste kirjutada JavaScript programmeerimiskeeles. Kõik töövahendid sobivad Reactjs teegiga tehtud veebirakenduse läbivtestimiseks ning neil pole ka operatsioonisüsteemide toe suhtes kriitilisi puudujääke. Samuti saab nendega teha läbivtestimist vähemalt kolmes erinevas veebibrauseris. Sellise võrdleva tabeli põhjal on näha, et kõik esialgsed kriteeriumid on täidetud ning autori jaoks on raske valida on antud võrdlevas tabeli info põhjal.

Järgmine oluline kriteerium on autori jaoks see, et läbivtestide töövahendil oleks hea dokumentatsioon, mida pidevalt uuendatakse ning ka suur kogukond, kes seda töövahendit kasutavad ning infot töövahendi kohta blogides ja ka muudes allikates levitavad. Samuti on oluline, et läbivtestide töövahendit arendatakse pidevalt edasi ning selle teeke ja muud funktsionaalsust uuendatakse. Tarkvara maailmas kõik pidevalt uueneb ning kui mingit teeki või raamistikku enam ei toetata, peab uue alternatiivi valima. Sellepärast töö autor rakendabki järgmises peatükis 4.3 populaarsuse mõõdikut.

Tabel 4. Läbivtestide töövahendite võrdlus esialgsete kriteeriumitega

Kriteeriumid	Cypress	Nightwatch	Playwright	Selenium Webdriver	Testcafe	WebdriverIO
Hind	Tasuta	Tasuta	Tasuta	Tasuta	Tasuta	Tasuta
Testide programmeerimiskeel	JavaScript	JavaScript	TypeScript, JavaScript, Python, .NET, Java	JavaScript, Python, Ruby, Java, Kotlin, C#	JavaScript	JavaScript
Võimaldab läbi graafilise kasutajaliidese testide teostamist	Jah	Jah	Jah	Jah	Jah	Jah
Sobivus React teegiga	Jah	Jah	Jah	Jah	Jah	Jah
Operatisooni süsteemide tugi	Vähemalt macOS 10.9, vähemalt Linux Ubuntu 12.04, Linux Fedora 21, vähemalt Windows 7	Pole piiranguid	Toetab kõike	Toetab kõike	Pole piiranguid	Toetab kõike
Brauserite tugi	Chrome, Firefox, Microsoft Edge	Chrome, Safari, Firefox, Microsoft Edge	Chrome, Safari, Firefox, Microsoft Edge	Chrome, Safari, Firefox, Microsoft Edge, Internet Explorer	Chrome, Safari, Firefox, Microsoft Edge, Internet Explorer	Chrome, Safari, Firefox, Microsoft Edge

4.3 Populaarsuse mõõdik

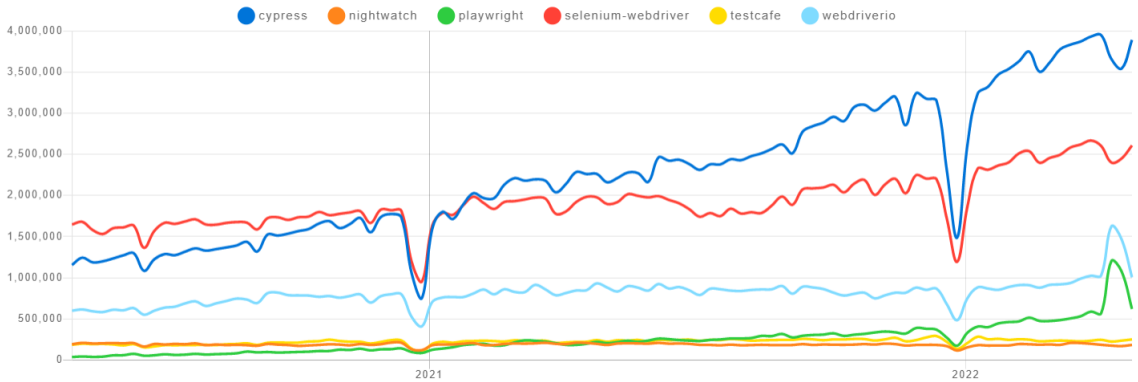
Populaarsus võib näidata, et tarkvara on kvaliteetne, kuid samuti on leitud, et see järeldus võib mõnikord olla näiline [31]. Veel on leitud [24], et populaarsemate projektide dokumentatsioon uueneb tihemini ja on tõenäolisem, et neid arendatakse tihemini [25]. Populaarsuse mõõdikuid on erinevaid ja sõltuvalt mõõdikust võib ka tulemus erineda.

Tarkvara pakettide populaarsuse mõõdikute vahelisi seoseid on uuritud [31], ning mõõdikud annavad erinevaid tulemusi, mille tõttu populaarsuse tähendus võib olenevalt uuringust erineda. Näiteks paljudes empiirilistes uurimustes võetakse esialgsesse valikusse projektid GitHub koodihoidla tähtede põhjal [25]. Samuti on levinud mõtteviis - mida rohkem on tarkvara paketti alla laetud, seda parem peaks see olema [31]. Sellise mõtteviisi põhjal tehakse tihti valik.

Allalaadimiste arvu on võrreldud selleks, et määrata läbivtestimist võimaldavate töövahendite populaarsust. Autor valis *Node Package Manager* paketihalduri ehk lühidalt NPM põhjal allalaadimiste arvu populaarsuse teguriks sellepärast, et kasutab Dashboard projektis sama paketihaldurit. Paketihaldurit kasutatakse tarkvarapakettide installeerimiseks, halduseks ja kõrvaldamiseks. NPM on standard paketihaldur Node.js rakenduste arendamisel. 2017. aasta jaanuaris oli NPM registris üle 350 000 paketi, mille tõttu oli see üks suuremaid JavaScript keeles olevaid koodi hoidlaid [32].

Joonisel 3 olev graafik saadi kasutades „npm trends“ graafikute tekitajat, mis sai info npmjs.com lehel olevast statistikast [33]. Autor võrdles Cypress, Nightwatch, Playwright, Selenium Webdriver, Testcafe ja WebdriverIO töövahendeid 2021. aastast kuni 2022. aasta maini allalaadimiste arvuga. Graafikult on näha, et kolm kõige populaarsemat alla laetud töövahendit on Cypress, Selenium Webdriver. Dashboard veebirakenduse läbivtestide töövahendite võrdlusesse võeti Cypressi, Selenium Webdriveri ja WebDriverIO.

Samuti on joonisel 3 näha, et kõige madalama populaarsusega töövahendil on allalaadimiste arv nulli lähedane, samal ajal kui kõige populaarsemal töövahendil on allalaadimisi olnud umbes 4 000 000. Selle põhjal jagab autor populaarsuse kahte gruppi. Ehk populaarsed töövahendid, millel on enamasti üle 1 500 000 allalaadimise ning vähempopulaarsed, millel on enamus ajast alla 1 500 000 allalaadimise.



Joonis 4. Veebibrauseri automatiseerimist võimaldavate töövahendite populaarsus aastal 2021-2022

On leitud, et tarkvaraarendajad märgivad tähega GitHubis olevad projektid, et näidata tunnustust, tähistada see hiljemaks vaatamiseks või siis sellepärast, et nad kasutavad seda [25].

Autor tegi võrdluse GitHub tähtede põhjal, mida on näha tabelis 6. Kõige rohkem tähti on Cypress töövahendil. Järgmisel kohal on Playwright ning kolmandal kohal on Selenium. Neljandal kohal on Nightwatch ja viiendal kohal on Testcafe. Viimasel kohal on WebdriverIO. Autor võttis võrdlusesse Seleniumi, kuna Selenium WebDriver on ainult üks töövahend Selenium projektist Github koodihoidlas.

Allalaadimiste arv põhjal läbi NPMi ja tähtede arv põhjal GitHub koodihoidlas on näha, et Cypress on mõlemas võrdluses esikohal. Teisel kohal tähtede arvult on Playwright töövahend, kuigi joonisel 3 on näha, et sellel on alla 500 000 allalaadimise olnud 2021 aastal. Selenium projektil on ainult 23 300 tähte, kuigi Seleniumi Webdriveri allalaadimisi oli 2021 aastal enamasti umbes 2 000 000 lähedal. Selline erinevus võib tulla sellest, et Selenium Webdriverit kasutavad paljud teised raamistikud veebibrauserite automatiseerimiseks.

WebdriverIO töövahendil on kõige vähem tähti tabel 6 põhjal, kuid NPM allalaadimisstatistika põhjal on see töövahend kolmandal kohal. Testcafe ja Nightwatch töövahendid on joonisel 4 umbes 250 000 allalaadimisega olnud 2021. aastal. Samuti on GitHub tähtede arv mõlemal raamistikul üksteisele lähemal teiste töövahenditega võrreldes. Nightwatchil on 11 100 tähte ja Testcafel on 9 300 tähte.

Autori jaoks on suurim erinevus Selenium ja Playwright projektidele antud tähtede ja nende tarkvarapakettide allalaadimise arvu vahel. Populaarsuse mõõdikuks võtab autor

siiski allalaadimiste arvu joonisel 4, ning võrdleb edasi töövahendeid, millel on 2021. aastal olnud üle 500 000 allalaadimise NPM andmete põhjal.

Kuigi GitHubi tähtede põhjal on Playwright projekt populaarsem kui Selenium projekt, kasutatakse Selenium WebDriveri veebibrauseri automatiseerimist paljudes muudes raamistikes, näiteks kasutab Nightwatch seda. GitHub tähtede ja NPM allalaadimiste põhjal populaarsuse mõõtmine andis erinevaid tulemusi nagu eeldati.

Tabel 5. Töövahendite populaarsus GitHub koodihoidla tähtede põhjal

Töövahend	GitHub tähed
Cypress	38 200
Nightwatch	11 100
Playwright	37 300
Selenium	23 300
Testcafe	9 300
WebdriverIO	7 500

5 Tulemused

Kirjeldatakse testjuhtumid, mille põhjal tehakse kolm testi Cypress, Selenium WebDriver ja WebdriverIO töövahendiga. Seejärel rakendatakse testmõõdikud, mis valiti 3.2.4 peatükis. Tehakse võrdlev tabel, kus on läbivtestide töövahendite jaoks täpsustavad kriteeriumid ja välja valitud testmõõdikute tulemused.

5.1 Testjuhtumid

Testjuhtum ehk *test case* on kindel arv tegevusi või samme, mida rakendades tehakse kindlaks, kas kindel funktsionaalsus töötab ootuspäraselt [34].

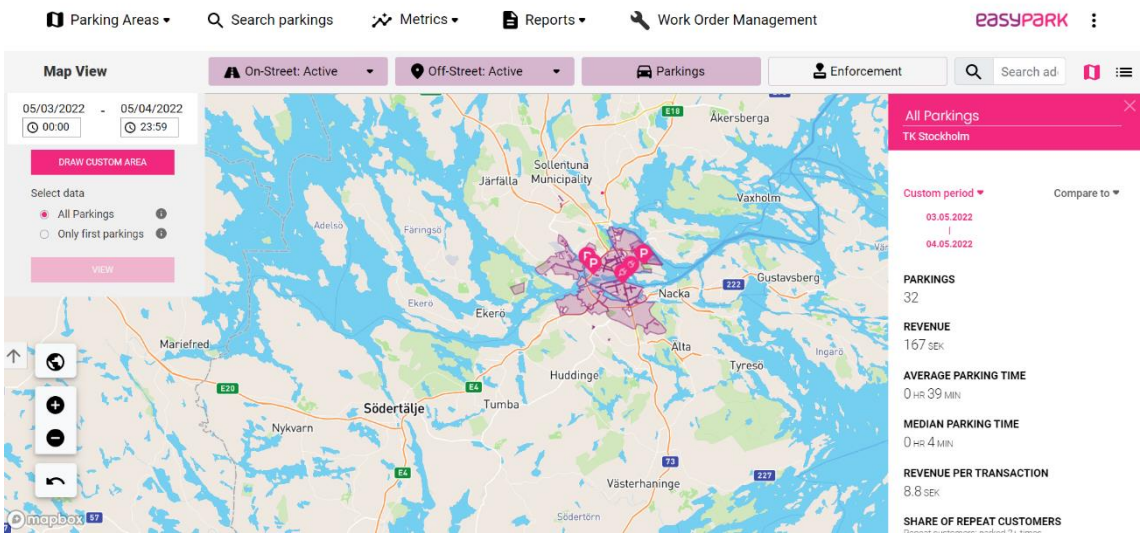
Võeti kolm kõige olulisemat testjuhtumit ning tehti nende põhjal läbivtestid kasutades Cypress, Selenium WebDriver ja WebdriverIO töövahendit. Testjuhtumid disainiti võimalikult realistlikuks tavakasutaja käitumisele. Tabel seitsmes on kirjas testjuhtumite id, testjuhtumite nimed ja nende sammud. Joonistel 5, 6 ja 7 olevad andmed ei ole päris ning on testimiskeskkonnas loodud, et tootmiskeskkonnas olevaid ehk päris kasutajate andmeid jäljendada.

- Testjuhtum TJ1 kirjeldab joonisel 5 oleva vaate kasutamist. Peale sisselogimist on näha „Parking Areas“ ehk parkimisalade veebilehte, kus vajutatakse alammenüüs olevale „Parkings“ nupule, mille järel avaneb vasakul küljepaneel, kus vajutatakse „VIEW“ nupule ning seejärel avaneb „All Parkings“ küljepaneel paremal. Kui see avanes, logitakse välja.

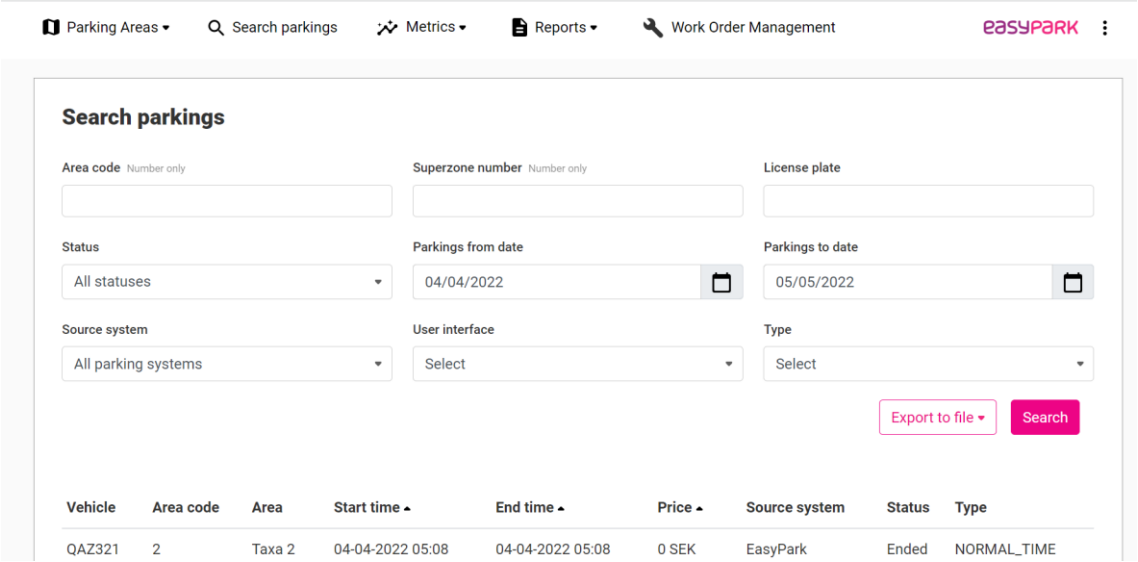
- Testjuhtum TJ2 kirjeldab parkimise otsimist tavakasutaja vaates. Tavakasutaja erineb mitme operaatoriga kasutajast, kuna peale sisse logimist tal ei tule operaatorite valiku modaal ette nagu TJ3 testjuhtumis. Peale sisselogimise sammu TJ2 testjuhtumis, järgmine samm on parkimiste otsingu veebilehele navigeerimine ülevalt menüüst vajutades „Search Parkings“. Seda veebilehte on näha joonisel 6. Seejärel sisestatakse *Area code* ehk piirkonna kood, mis on number 3 ning vajutatakse otsingu ehk „Search“ nuppu. Seejärel võetakse tulemuste tabelist esimene rida ja vaadatakse, kas piirkonna kood on number 3.

Kui on, siis vajutatakse sellele reale ja avaneb modaalaken täpsustava infoga. Seejärel logitakse välja.

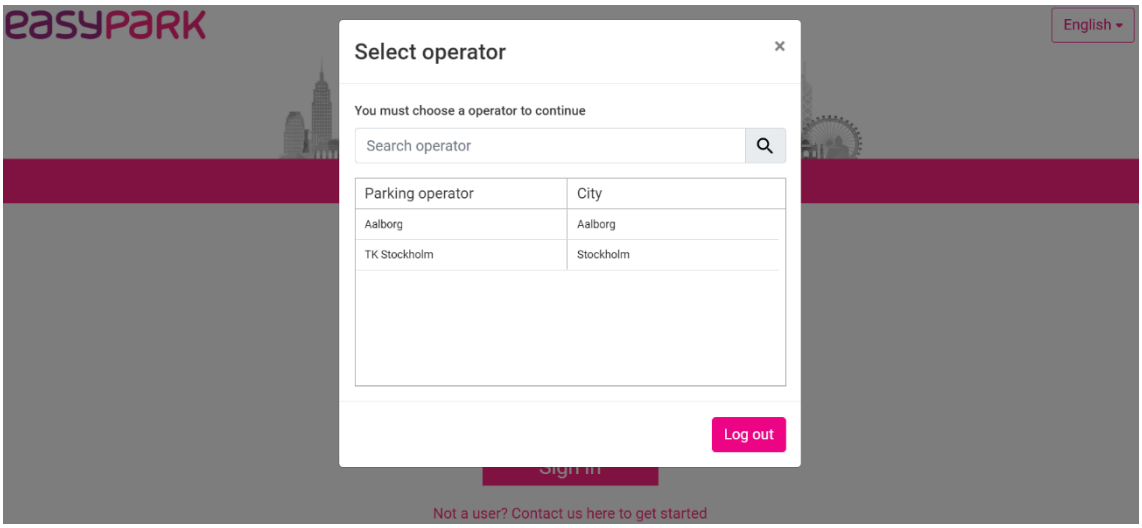
- Testjuhtum TJ3 kirjeldab sellise kasutajaga sisselogimist, kes saab siseneda mitme operaatori keskkonda. Peale kasutajanime ja parooli sisestamist ning „Sign In“ ehk sisselogimise nupule vajutamist avaneb modaalaken, kus saab valida millise parkimisoperaatori infot soovib kasutaja vaadata. Modaalakent on näha joonisel 7. Valides TK Stockholm avaneb „Parking Areas“ ehk parkimisalade veebileht. Seejärel logitakse välja.



Joonis 5. Parking Areas veebilehe ehk avalehe vaade



Joonis 6. Search parkings veebilehe vaade



Joonis 7. Operaatori valik sisselogimisel

Tabel 6. Testjuhtumite sammud

Id	Testjuhtum	Sammud
TJ1	Parkimiste vaatamine „Parking Areas“ vaates	1. Sisse logimine 2. „View“ nupu vajutamine küljepaneelil 3. „All parkings“ küljepaneel avanes 4. Välja logimine
TJ2	Parkimise otsimine – tavakasutaja vaates	1. Sisse logimine 2. Parkimiste otsingu lehele navigeerimine 3. Piirkonna koodi sisestamine 4. Otsingu nupule vajutamine 5. Esimesele otsingu tulemusele vajutamine 6. Modaalaken avaneb 7. Välja logimine
TJ3	Mitme operaatoriga kasutaja sisselogimine	1. Sisse logimine 2. Operaatorite listist operaatori valik 3. „Parking Areas“ vaade avaneb 4. Välja logimine

5.2 Testmõõdikute rakendamine

3.2.3 peatükis välja valitud testmõõdikuid rakendati läbivtestidele, mis kirjutati kasutades Seleniumit, Cypressi ja WebdriverIO töövahendeid. Autor pani oma lokaalses masinas rakenduse käima kasutades testkeskkonda, kus on testandmed. Seejärel käivitas testid, mis avasid veebibrauseri ning testis olevate käskluste põhjal jäljendasid vajutusi ja andmete sisestamist.

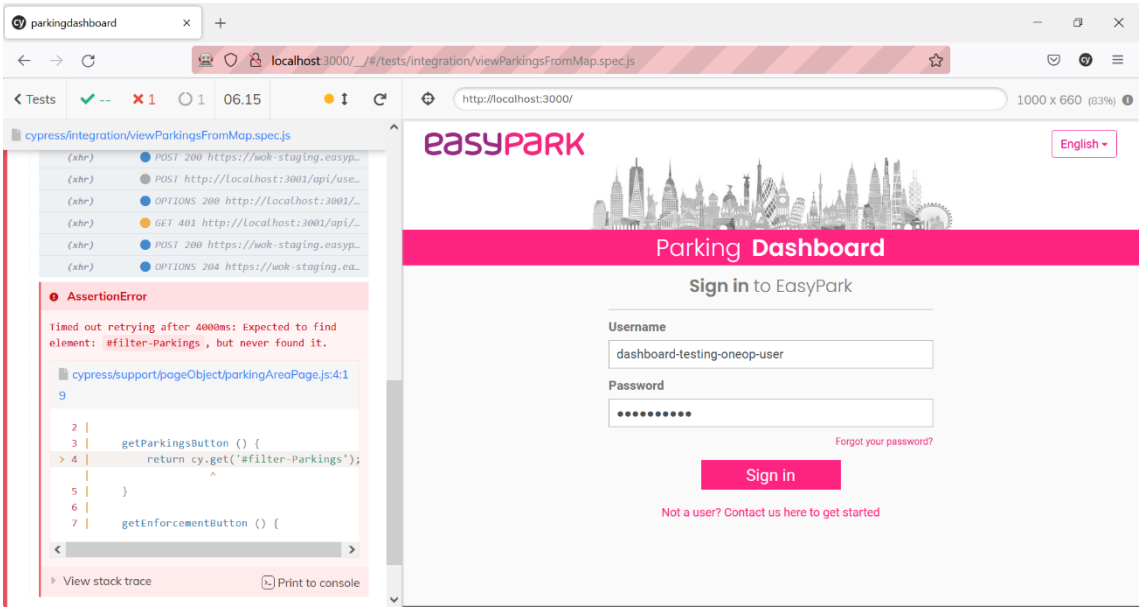
5.2.1 Edukus lähtuvalt testi nõuetest

See on ülevaatlik mõõdik, et näha kas testjuhtumite sammud saati edukalt läbiviidud. Autor märkis positiivse tulemuse korral OK ja negatiivse tulemuse korral VIGANE. Tabel 8 põhjal on näha, et TJ2 testjuhtum oli Selenium WebDriverit ja WebdriverIO töövahendit kasutades vigane ehk autor ei saanud seda tööle. Test mis oli tehtud TJ2

testjuhtumile, läks katki kuna ajaliselt ei jõudnud lehekülg end ära laadida, et navigeerida parkimise otsingu veebilehele. TJ1 ja TJ3 polnud veebilehtede vahel navigeerimist kasutades menüüd ja seda probleemi ei ilmnenud. Testi parandamine autoril esimese korraga ei õnnestunud ning Selenium WebDriveri või WebdriverIO valimisel peaks sellele probleemile lahenduse leidma.

Tabel 7. Edukus lähtuvalt testi nõuetest

Testjuhtum	Selenium WebDriver	Cypress	WebdriverIO
TJ1	OK	OK	OK
TJ2	VIGANE	OK	VIGANE
TJ3	OK	OK	OK



Joonis 8. Firefox veebibrauseris testide jooksutamine

5.2.2 Testide teostamise aeg erinevates veebibrauserites

Selle mõõdiku jaoks valis autor 3 veebibrauserit, mida Dashboard veebirakendusega enim kasutatakse ning võrdles neid ajaliselt. Tulemused, mida on näha tabelis 9, mõõdeti sekundites. Firefoxi veebibrauseris Cypress töövahendiga loodud testide käivitamine oli autori jaoks probleemne, sest samad testid mis Chrome ja Microsoft Edge veebibrauseris töötasid, ei töötanud Firefox veebibrauseris. Sisselogimise käigus peale „Sign in“ nupu vajutust, jäi test kinni. Joonis 8 peal on näha Cypress testi katki minemas. Ilmselt on võimalik testid saada tööle Firefoxiga, kuid esimese katsetamise käigus seda tööle ei saadud. Kuna testide vajaduspõhise katvuse mõõdiku tulemustest oli näha, et TJ2

testjuhtumi põhjal tehtud test oli Selenium WebDriveri ja WebdriverIOga kirjutades vigane, ei saanud autor neid selletõttu ajaliselt mõõta. Ning tabelis märkis nende tulemused puuduvaks, ehk „-“ märgiga.

Tabel 8. Testide teostamise aeg sekundites

Testjuhtum	Selenium WebDriver	Cypress	WebdriverIO
Chrome			
TJ1	3,6	6,2	3,7
TJ2	-	7,2	-
TJ3	4,6	5	3,4
Microsoft Edge			
TJ1	4,1	5,5	3,8
TJ2	-	7,1	-
TJ3	3,6	5,0	3,6
Firefox			
TJ1	6,8	-	4,9
TJ2	-	-	-
TJ3	6,2	-	4,9

5.2.3 Testide kirjutamise efektiivsuse mõõdik

Testid kirjutati eelnevas peatükis olevate testilugude põhjal Page Object mudelit kasutades. Page Object on testide kujundamise muster, mis määrab kuidas teste kaustadesse jaotada ja neid kirjutada nii, et võimalikult vähe koodi korduks [35]. Sellega üritas autor teha testide kirjapilti võimalikult sarnaseks, et oleks lihtsam võrrelda. Testide jaoks oli kaks alamkausta PageObject ja Tests kaust. PageObject kaustas oli header.js fail, parkingAreaPage.js fail, searchParkingsPage.js fail ja signInPage.js fail. Testide kaustas oli 3 faili, mis vastasid testjuhtumitele ning mida mõõdeti. Nende testide põhjal lisas autor info tabelisse. Testide efektiivsuse mõõdikuga loeti kokku mitu rida koodi oli vaja kirjutada ühe testi jaoks. Tabel 10 põhjal on näha, et Cypress ja WebdriverIO koodiread erinevad Selenium WebDriveri tulemusest. See on sellepärast, et Selenium WebDriverit kasutades peab ise vajalikud draiverid faili lisama, samal ajal kui raamistikke on see juba sisse ehitatud.

Tabel 9. Testide efektiivsus

Testjuhtum	Selenium WebDriver	Cypress	WebdriverIO
TJ1	22	17	17
TJ2	-	20	-
TJ3	21	16	16

5.3 Läbivtestide töövahendite võrdlus

Sobilike mõõdikute ja välja valitud kriteeriumide põhjal tehakse võrdlev tabel ning valitakse sobiv meetod kriteeriumide võrdlemiseks. Samuti tehakse ülevaade võrreldavatest läbivtestide töövahenditest. Cypress, Selenium WebDriver ja WebdriverIO vastasid 3.1.1 peatükis olevatele esialgsetele kriteeriumitele ning on samuti populaarsed, 4.1 peatükis tehtud mõõtmise põhjal.

5.3.1 Võrdlev tabel testimise põhjal

Täpsemate kriteeriumide põhjal 3.1.1 peatükis, pani autor kokku läbivtestide töövahendite võrdluse tabeli. Võrdlusesse lisati ka testide vajaduspõhise katvuse testmõõdik.

Tabel 11 põhjal on näha, et kriteeriumideks võeti edukus lähtuvalt testi nõuetest mõõdik, mille väärtusteks on osaline katvus või täielik katvus. Kuna seadistamise ja õppimise keerukust on raske hinnata ning see võib oleneda näiteks kogemustest ja veel muudest faktoritest, siis hindas ka autor neid lihtsam/keerulisem stiilis enda kogemuse põhjal. Väärtus olid keerulisem, kui autor pidi peale ametliku dokumentatsiooni muutest kanalitest infot otsima. Väärtus oli lihtsam, kui ametlikust dokumentatsioonist piisas.

Lihtsam/keerulisem stiilis hindas autor ka seadistamise keerukust. Muu kriteeriumi alla lisas autor info sellest, kas võrdluses olevatele läbitestide töövahenditele on vaja teeke ja muid raamistikke juurde, või pole vaja ise juurde midagi otsida.

Tabel 10. Läbivtestide töövahendite võrdlus

Kriteerium	Selenium WebDriver	Cypress	WebdriverIO
Edukus lähtuvalt testi nõuetest - mõõdik	Osaline katvus	Täielik katvus	Osaline katvus
Õppimise keerukus	Keerulisem	Lihtsam	Lihtsam
Seadistamine	Keerulisem	Lihtsam	Lihtsam
Muu	Veebibrauseri automatiseerimise vahend – vaja muid teeke ja raamistikke juurde, et teste kirjutada.	Raamistik - ehk kõik vajalikud muud teegid ja raamistikud on olemas.	Raamistik – kõik vajalikud teegid ja raamistikud on olemas ning nende vahel saab valida

Testide efektiivsuse mõõdikut ja testide teostamise aja mõõdikut ei lisatud, kuna kõiki andmeid nendele mõõdikutele ei õnnestunud saada ning seetõttu täielikku järeldust polnud võimalik teha.

5.3.2 Otsuse tegemine

Tabel 11 annab üldise ülevaate läbivtestide töövahenditest, kuid autori jaoks on kriteeriumid erineva tähtsusega. Valiku langetamise puhul on võimalik kasutada otsustusteooriat. Otsustusteooria on alternatiivsete võimaluste leidmine ja valimine sõltuvalt otsuse tegijast. Otsustusteoorias on palju erinevaid meetodeid, mida valitakse sõltuvalt kriteeriumide objektiivsusest näiteks hind või siis subjektiivsusest nagu näiteks kasutamise keerukus. Samuti on oluline, kas üks või mitu kriteeriumi, fikseeritud number või määramatu number alternatiive [36].

Kuna numbriliselt on raske kriteeriumide väärtusi tabelis 11 määrata, kasutas autor leksikograafilist meetodit ehk inglise keeles *lexicographic method* [36]. See meetod kuulub mitme atribuudiga otsuste tegemise meetodite alla. Leksikograafilises meetodis järjestatakse ja vaadatakse läbi kriteeriumid lineaarses olulisuse järjekorras ning jäetakse välja alternatiivid, mis ei ole nii eelistatud kui teised alternatiivid [37].

Kõige olulisem kriteerium ehk esimesel kohal oli autori jaoks see, et edukus lähtuvalt testi nõuetest on täidetud. Kuna selle kriteeriumi tõttu langesid välja Selenium WebDriver ja WebdriverIO, siis järgmisi kriteeriume otsustamisel enam ei pidanud valima ning autor valis Cypress töövahendiks.

5.4 Analüüs

Kriteeriumite võrdlemine tabelis annab ülevaate, mille põhjal saab kasutada kaaluvaid ja otsuse tegemist abistavaid meetodeid. Esialgse valiku kriteeriumite põhjal oli 4.1 peatükis näha, et kui on palju läbivtestide töövahendeid, mis vastavad kriteeriumitele, on raske valikut teha. Sel juhul on abi mõõdikute kasutamisest. Töös rakendas autor populaarsuse mõõdikut, edukus lähtuvalt testi nõuetest mõõdikut, testide teostamise aeg erinevates veebibrauserites mõõdikut ning testide kirjutamise efektiivsuse mõõdikut.

Peale testimist sai autor lisada väärtused täpsustavatele kriteeriumitele ning teha otsuse testmõõdikute põhjal. Kahjuks kõik testmõõdikud ei andnud täielikke tulemusi ning autor tegi otsuse ühe testmõõdiku põhjal. Seejärel rakendati otsuse tegemisel leksikograafilist meetodit, kuna tabelis olid kriteeriumid erineva olulisusega ning ei olnud numbriliselt mõõdetavad. Autori jaoks oli vajaduspõhisest testide katvuse mõõdik kõige olulisem.

Autor leiab tulemuste põhjal, et kõige parema ülevaate saab läbivtestide töövahenditega ise katsetades ning mõõdikute rakendamisel. Esialgsed kriteeriumid olid informatiivsed, kuid ilma teste tegemata oleks valik olnud raske.

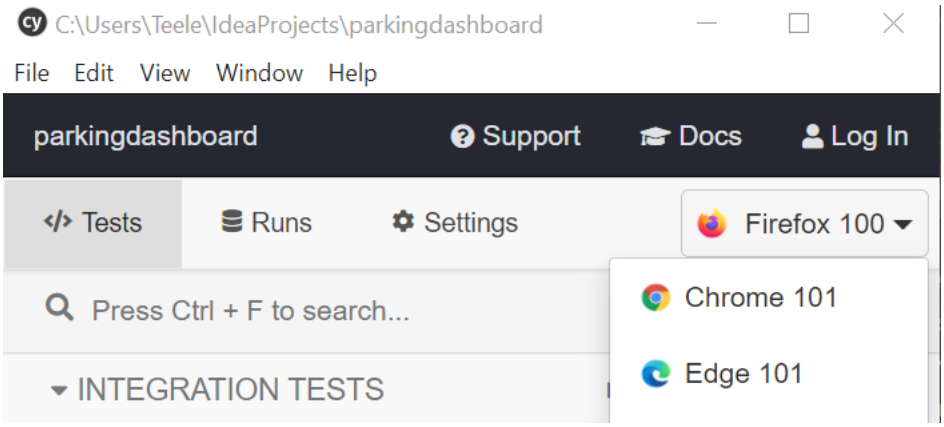
Testmõõdikute rakendamine näitas kiirelt ära, et raskesti laetavate veebilehtede puhul oli potentsiaalsete töövahendite proovimine oluline, kuna võib tulla ette probleeme, millega töövahendi kodulehel oleva info põhjal ei oskaks arvata. Samas on iga veebirakendus erinev ja kui aeglaseid veebilehti ei ole, siis ilmselt see mõõdik ei ole ka nii vajalik.

Autori jaoks oli testmõõdikute puhul huvitav näha tulemusi ajaliselt. Kuigi Cypress töövahendiga testimisel tundus kirjutamine kiiresti minevat ja selletõttu testide teostamise aeg ka kiire, siis mõõdiku põhjal on näha, et see töövahend võttis olemasolevate tulemuste põhjal rohkem aega, kui teised töövahendid. Testide teostamise aja mõõdikuga ei saadud kõigile töövahenditele kõikides testides tulemusi. Nende tulemuste põhjal, mis saadi oli

Chrome veebibrauseriga TJ1 testimisel parim aeg Seleniumiga testides ja TJ3 puhul oli parim aeg WebdriverIO raamistikuga. Edge veebibrauseriga TJ1 testimisel oli parim aeg WebdriverIO raamistikuga ning TJ3 puhul Selenium ja WebdriverIO jagasid parimat aega. Nende tulemuste järgi võib väita, et WebdriverIO andis ajaliselt parimaid tulemusi. Samas on see poolik tulemus ning tabeli võrdlusel ei saanud seda arvesse võtta.

Testide efektiivsuse mõõdiku puhul autor suurt väärtust ei näinud. Autori arust on näiteks käskude kirjutamise mugavus olulisem kui koodiridade arv. Mõõdiku eesmärk oli mõõta, kui palju vaeva nõuab testi kirjutamine erinevate läbivtestide töövahenditega. Samas on sellest mõõdikust raske järeldusi teha, kui tulemused oluliselt erinevad ei ole.

Kõike töövahendeid proovides tekkisid autoril ka uued kriteeriumid, mis enne testide tegemist ei olnud. Näiteks kuigi Cypress töövahendit kasutades Firefox veebibrauseriga samad testid ei töötanud, mis teiste veebibrauseritega töötasid, siis valimine erinevate veebibrauserite vahel oli selle töövahendiga kõige lihtsam. Joonis 9 peal on näha veebibrauseri valiku tegemist läbi Cypressi testide teosamise kasutajliidese. Veebibrauserid pidid lihtsalt lokaalses masinas olemas olema, samas kui kahe teise töövahendiga oli vaja veebibrauserite draiverid alla laadida ning need testidele kättesaadavaks teha. Joonis kümne peal on näha koodi, mida pidi kirjutama, et teha Firefox veebibrauseri kasutus Selenium WebDriveriga tehtud testidele kättesaadavaks. Joonis üheteistkümne peal on näha geckodriver.exe käivitamist. Selle faili käivitamine oli vajalik WebdriverIO testide käivitamiseks Firefox veebibrauseris.



Joonis 9. Veebibrauseri valimine testide jaoks Cypress töövahendiga

```
const { Builder } = require('selenium-webdriver');
const firefox = require('selenium-webdriver/firefox');
const service = new firefox.ServiceBuilder( opt_exe: './geckodriver.exe');
const driver = new Builder().forBrowser( name: 'firefox').setFirefoxService(service).build();
const signInPage = require('../pageObject/SignInPage');
const parkingAreaPage = require('../pageObject/ParkingAreaPage');
const header = require('../pageObject/Header');
describe('View data from Parking Areas Page', function(){
```

Joonis 10.Veebibrauseri valimine testide jaoks Selenium WebDriver töövahendiga

Joonis 11. Veebidraiveri käivitamine WebdriverIO töövahendiga tehtud testide jaoks

6 Kokkuvõte

Magistritöös keskenduti töövahendite võrdlusele, mis võivaldavad läbi graafilise kasutajaliidese läbivtestimist teha. Eesmärgiks oli luua kasulik meetod läbivtestide töövahendi valimiseks. Vaadati üle hetkel olemasolevad kriteeriumid ja mõõdikud, mida on varem sarnastes olukordades rakendatud. Seejärel valiti sobivad kriteeriumid ja varem rakendatud testmõõdikud automaattestide töövahendite võrdlusel ning lisati juurde ka populaarsuse mõõdik ning testi aja mõõdik sõltuvalt veebibrauserist.

Peale esialgsete kriteeriumide ja populaarsuse mõõdiku rakendamist jäi valikusse kolm potentsiaalset läbitestide töövahendit. Nendeks oli Selenium WebDriver, Cypress ja WebdriverIO. Peale seda kirjutati kolmele kõige olulisemale testjuhtumile testid, kasutades kolme välja valitud töövahendit. Seejärel rakendati testmõõdikud - testide teostamise aeg, testide efektiivsus, edukus lähtuvalt testi nõuetest. Seejärel tehti täpsemate kriteeriumide põhjal võrdlev tabel. Täpsemateks kriteeriumideks kujunes: tehniline tugi, mida mõõdeti populaarsuse mõõdiku põhjal, edukus lähtuvalt testi nõuetest mõõdik, õppimise keerukus, seadistamine ja viimaseks lisainfo kriteerium ehk muu.

Dashboardi veebirakenduse jaoks kõige olulisemaks kriteeriumiks kujunes testide vajaduspõhise katvuse mõõdik. Testide teostamise aja mõõtmine sõltuvalt veebibrauserist ja testide efektiivsuse mõõdikute puhul jäid tulemused ainult osaliseks. Testmõõdikute tulemused olid sellised, kuna ühe valitud testjuhtumi põhjal kirjutatud testid olid kahe potentsiaalse läbivtestide töövahendiga tehes vigased. Muud võrdlevas tabelis olevad kriteeriumid olid olulised, kuid ei olnud otsuse tegemisel määravad, kuna autori jaoks oli testide korrasolek kõige olulisem ning ühe testjuhtumi välja jätmine tulemuste võrdluses ei olnud õiglane.

Magistritöö andis olulist infot Dashboardi veebirakendust arendavale meeskonnale ning aitas välja valida läbivtestide jaoks uue töövahendi. Väljavalitud töövahendiks osutus Cypress ning järgmise sammuna tehakse sellega ka ülejäänud läbivtestid ning üritatakse leida lahendus testide ajalises mõõdikus välja tulnud puudusele.

7 Kasutatud kirjandus

[1] Eesti Rahvusraamatukogu, „Raamatukogusõnastik,“ 2018. [Võrgumaterjal]. Available: https://termin.nlib.ee/view/3446. [Kasutatud 24 Märts 2022].

[2] Cybernetica AS, „Akit,“ [Võrgumaterjal]. Available: https://akit.cyber.ee/. [Kasutatud 24 Märts 2022].

[3] V. Garousi, M. Felderer, M. Kuhrmann, K. Herkiloğlu ja S. Eldh, „Exploring the industry's challenges in software testing: Anempirical study,“ *Software: Evolution and Process,* kd. 32, nr 8, 2020.

[4] L. Leite, G. Pinto, F. Kon ja P. Meirelles, „The organization of software teams in the quest for continuous delivery: A grounded theory approach,“ *Information and Software Technology,* kd. 139, nr 106672, 2021.

[5] H. Bajaj, *Choosing the right automation tool and framework is critical to project success,* Infosys Limited, 2015.

[6] C. Tozzi, „Top 5 JavaScript Test Automation Frameworks in 2021,“ Saucelabs, [Võrgumaterjal]. Available: https://saucelabs.com/blog/top-5-javascript-test-automation-frameworks-in-2021. [Kasutatud 16 Aprill 2022].

[7] P. Raulamo-Jurvanen, M. Mäntylä ja V. Garousi, „Choosing the Right Test Automation Tool: a Grey Literature Review of Practitioner Sources,“ *EASE'17: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering,* p. 21–30, 2017.

[8] R. K. Lenka, K. M. Nayak ja S. Padhi, „Automated Testing Tool: QTP,“ *2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN),* pp. 526-532, 2018.

[9] C. Ebert, G. Gallardo, J. Hernantes ja N. Serrano, „DevOps,“ *IEEE Software,* kd. 33, nr 3, pp. 94-100, 2016.

[10] H. Vocke, „The Practical Test Pyramid,“ 26 Veebruar 2018. [Võrgumaterjal]. Available: https://martinfowler.com/articles/practical-test-pyramid.html. [Kasutatud 21 Märts 2022].

[11] Selenide, „Selenide and Selenium comparison,“ Codeborne, [Võrgumaterjal]. Available: https://selenide.org/documentation/selenide-vs-selenium.html. [Kasutatud 10 Märts 2022].

[12] Concise Systems OÜ, „Superagile,“ [Võrgumaterjal]. Available: https://concise.ee/superagile/. [Kasutatud 03 Märts 2022].

[13] F. Mobaraya ja S. Ali, *Technical Analysis of Selenium and Cypress as Functional Automation Framework for Modern Web Application Testing,* Auckland: Department of Information Technology, AGI Institute, Auckland, New Zealand, 2019.

[14] M. Cohn, Succeeding with agile: software development using Scrum, Addison-Wesley Professional, 2009.

[15] theteacher.info Ltd, „Types of user interface,“ [Võrgumaterjal]. Available: http://theteacher.info/index.php/systems-software/notes/4623-types-of-user-interface. [Kasutatud 9 Mai 2022].

[16] L. Lazic ja N. Mastorakis, „Cost effective software test metrics“.

[17] S. U. Farooq, S. M. K. Quadri ja N. Ahmad, „Software measurements and metrics: Role in effective software testing,“ *International Journal of Engineering Science and Technology,* kd. 3(1), pp. 671-680, 2011.

[18] P. B. Nirpal ja K. V. Kale, „A brief overview of software testing metrics,“ *International Journal on Computer Science and Engineering,* kd. 3, nr 1, pp. 204-2011.

[19] R. Angmo ja M. Sharma, „Performance evaluation of web based automation testing tools,“ *2014 5th International Conference - Confluence The Next Generation Information Technology Summit (Confluence),* pp. 731-735, 2014.

[20] H. Gamido ja M. Gamido, „Comparative Review of the Features of Automated Software Testing Tools,“ *International Journal of Electrical and Computer Engineering,* kd. 9, pp. 4473-4478, 2019.

[21] Ranorex Inc., „Image-based testing,“ Ranorex Inc., [Võrgumaterjal]. Available: https://www.ranorex.com/help/latest/ranorex-studio-advanced/image-based-automation/introduction/. [Kasutatud 25 Aprill 2022].

[22] Mixpanel, „Mixpanel,“ [Võrgumaterjal]. Available: https://mixpanel.com/. [Kasutatud 26 Märts 2022].

[23] Protractortest, „Protractor,“ [Võrgumaterjal]. Available: https://www.protractortest.org/#/. [Kasutatud 9 Mai 2022].

[24] M. Saini, R. Verma, A. Singh ja K. K. Chahal, „Investigating diversity and impact of the popularity metrics for ranking software packages,“ *Journal of Software: Evolution and Process,* kd. 32, nr 9, 2020.

[25] H. Borges ja M. T. Valente, „What’s in a GitHub Star? Understanding Repository Starring Practices in a Social Coding Platform,“ *Journal of Systems and Software,* kd. 146, pp. 112-129, 2018.

[26] Cypress.io, „How it works,“ Cypress.io, [Võrgumaterjal]. Available: https://www.cypress.io/how-it-works. [Kasutatud 15 Aprill 2022].

[27] Selenium, „Selenium,“ Selenium, [Võrgumaterjal]. Available: https://www.selenium.dev/. [Kasutatud 20 Märts 2022].

[28] BrowserStack, „Nightwatch.js,“ [Võrgumaterjal]. Available: https://nightwatchjs.org/. [Kasutatud 8 Mai 2022].

[29] Microsoft, „Playwright,“ [Võrgumaterjal]. Available: https://playwright.dev/. [Kasutatud 08 Mai 2022].

[30] TestCafe, „Why TestCafe?,“ [Võrgumaterjal]. Available: https://testcafe.io/documentation/402631/why-testcafe. [Kasutatud 8 Mai 2022].

[31] A. Zerouali, T. Mens, G. Robles ja G.-B. J. M., „On the Diversity of Software Package Popularity Metrics: An Empirical Study of npm,“ *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER),* Hangzhou, 2019.

[32] Node.js, „An introduction to the npm package manager,“ Node.js, [Võrgumaterjal]. Available: https://nodejs.dev/learn/an-introduction-to-the-npm-package-manager. [Kasutatud 1 Mai 2022].

[33] J. Potter, „npm trends,“ [Võrgumaterjal]. Available: https://www.npmtrends.com/cypress-vs-nightwatch-vs-playwright-vs-puppeteer-vs-selenium-webdriver-vs-testcafe-vs-webdriverio-vs-nightmare. [Kasutatud 26 Aprill 2022].

[34] BrowserStack, „How to write Test Cases with Examples,“ [Võrgumaterjal]. Available: https://www.browserstack.com/guide/how-to-write-test-cases. [Kasutatud 9 Mai 2022].

[35] Selenium, „Page object models,“ Selenium, [Võrgumaterjal]. Available: https://www.selenium.dev/documentation/test_practices/encouraged/page_object_models/. [Kasutatud 8 Mai 2022].

[36] J. Fülöp, „Introduction to decision making methods,“ *BDEI-3 workshop*, Washington, 2005.

[37] J. Dombi, C. Imreh ja N. Vincze, „Learning lexicographic orders,“ *European Journal of Operational Research,* kd. 183, nr 2, pp. 748-756, 2007.

[38] A. Contan, C. Dehelean ja L. Miclea, „Test automation pyramid from theory to practice,“ *2018 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR),* pp. 1-5, 2018.

[39] Scrum.org, „What is Scrum?,“ Scrum.org, [Võrgumaterjal]. Available: https://www.scrum.org/resources/what-is-scrum. [Kasutatud 10 April 2022].

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

Mina, Teele Pae

1. Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Läbivtestide testimisvahendi valik veebirakenduse näitel“, mille juhendaja on Jekaterina Tšukrejeva
   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.

11.05.2022

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.