OULUN AMMATTIKORKEAKOULU

Aleksi Nokela

AUTOMAATTISEN TESTAUSTYÖKALUN VALINTA PROJEKTIIN

AUTOMAATTISEN TESTAUSTYÖKALUN VALINTA PROJEKTIIN

Aleksi Nokela
Opinnäytetyö
Kevät 2023
Tietojenkäsittely
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma

Tekijä: Aleksi Nokela
Opinnäytetyön nimi: Automaattisen testaustyökalun valinta projektiin.
Työn ohjaaja(t): Tuula Harju ja Raili Simanainen
Työn valmistumislukukausi ja -vuosi: Kevät 2023 Sivumäärä: 51 + 7 liitettä

Opinnäytetyön aiheena on evaluoida erilaisia automaattiseen ohjelmistotestaukseen tarkoitettuja työkaluja spesifisesti määritettyjen arviointivaatimusten perusteella. Evaluaation tulosta on tarkoitus käyttää pohjana työkalun valinnalle Younite-AI-yrityksen asiakasprojektin käyttöön. Valittua työkalua on tarkoitus käyttää tulevaisuudessa asiakasprojektin ohjelmistotuotteiden testaukseen.

Työn lähtökohtana toimi Younite-AI-yrityksen kehitystiimin tarve kehittämänsä asiakasprojektin ohjelmistotuotteiden automaattiseen testaukseen, jotta tuotteiden kehitystyötä olisi mahdollista tehostaa ja nopeuttaa. Automaattiseen testauksen käyttöönottamiseksi tulisi löytää työkalu, joka soveltuisi mahdollisimman hyvin asiakasprojektin käyttöön.

Työ suoritettiin tutustumalla ensin yleisellä tasolla ohjelmistotestaukseen, eri testaustapoihin sekä testausautomaatioon. Soveltuviin työkaluihin ja niiden ominaisuuksiin tutustuttiin, ja evaluoitavaksi valikoitui kolme työkalua. Evaluaatiovaatimukset työkalun sopivuudeksi sovittiin yhdessä projektin kehitystiimin kanssa. Evaluoitavaksi valitut työkalut arvioitiin jokaisen vaatimuksen osalta. Itse evaluointi tapahtui tutustumalla dokumentaatioon sekä testaamalla työkaluja käytännössä. Lopuksi valittiin evaluaation perusteella sopivin työkalu.

Opinnäytetyö eteni hyvin ja aikataulun mukaisesti. Kaikkia evaluoitavia työkaluja onnistuttiin testaamaan käytännössä, ja evaluointien toteutus onnistui hyvin. Evaluointien perusteella valittiin sopivin työkalu, jota voitiin suositella projektin käyttöön.

Evaluoinnin tuloksen perusteella työkalua tarkasteltiin perusteellisemmin, jotta voitiin todeta sen soveltuvuus asiakasprojektin käyttöön. Työkalun perusteellisempi tarkastelu ja työkalun mahdollinen käyttöönotto asiakasprojektissa toteutetaan myöhemmässä vaiheessa, joten se ei kuulu tämän opinnäytetyön piiriin.

Asiasanat: ohjelmistotestaus, testaus, testausautomaatio, evaluointi, testaustyökalu

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Business Information Systems

Author: Aleksi Nokela
Title of thesis: Choosing the tool for automatic software testing for a project
Supervisor(s): Tuula Harju & Raili Simanainen
Term and year when the thesis was submitted: Spring 2023
Number of pages: 51 + 7 appendices

The subject of this thesis is to evaluate different automation testing tools for software testing. The results of the evaluation would form the basis for deciding whether one of the evaluated tools might be suitable for the needs of a customer project of Younite-AI software company. If so, in the future the tool would be taken into use for testing in the project.

The need for the evaluation arose from the company Younite-AI, who wanted to start automating their test cases for software products they are developing for a customer. With automated testing they would be able to improve and haste the development process. The automation tool to be taken into use should meet the requirements of the project as well as possible. Thus, evaluation of tools was required to find the most suitable one.

The thesis was accomplished by exploring documentation and familiarizing oneself with different automation tools in practice. At the start, one got acquainted with software testing and testing automation in general. Then three tools were prechosen for evaluation. The evaluation criteria for tools to meet were decided in co-operation with the development team. Each of the three tools were then evaluated criterion by criterion. Evaluation was conducted by exploring documentation and by testing the tools in practice. At the end of the process the most promising tool based on the evaluations was chosen.

The thesis progressed well and was completed in time. All the tools in the evaluation were successfully tested in practise and the evaluation processes were successful. At the end of the process, the most suitable tool was succeeded to be chosen for the best candidate for the tool to start using in the project.

The tool that was chosen could now be more extensively and more specifically researched and tested to really make sure that it undoubtedly suits for the needs of the project. This further testing was to be conducted at a later phase and was not part of this thesis.

Keywords: software testing, testing automation, evaluation, testing tool

SISÄLLYS

1 JOHDANTO

Ohjelmiston tai järjestelmän on täytettävä kaikki sille asetetut tavoitteet ja vaatimukset, eikä siinä saa olla toiminnan estäviä virheitä. Tämän varmistamiseksi tulee ohjelmistoa testata sen jokaisessa kehitysvaiheessa. Ohjelmistotestauksella tarkoitetaan niitä manuaalisesti tai automaattisesti tapahtuvia menetelmiä, joilla ohjelmiston toimivuus varmistetaan. (Hamilton 2023b.) Ohjelmistotestaus on erittäin tärkeä osa ohjelmiston kehitystä. Sen ansiosta ei ainoastaan varmisteta, että ohjelmisto toimii halutulla tavalla, vaan sen avulla voidaan myös vähentää sovelluksen kehitykseen kuluvaa aikaa ja siten kehityksen kustannuksia. Ohjelmistotestauksella voidaan parantaa sovelluksen laatua ja varmistua sen toimivuudesta jatkuvasti sen kehityksen edetessä. (Atlassian 2023.) Sovellustestauksen avulla voidaan parantaa myös tietoturvaa sekä sovelluksen käyttäjätyytyväisyyttä (Hamilton 2023b).

Automaattisessa testauksessa käytetään ohjelmistojen testaukseen eri työkaluja, joilla pystytään simuloimaan niitä toimia, joita käyttäjä tekisi ohjelmistolla, ilman että ihmisen tarvitsee niitä manuaalisesti toteuttaa. Automaattinen testaus on manuaaliseen testaukseen verrattuna nopeampaa ja tehokkaampaa, sekä säästää sovelluksen kehityksen resursseja. (Rehkopf 2023a.) Automaattiseen testauksen toteuttamiseksi on olemassa lukuisia eri työkaluja (Tricentis 2016).

Tämän opinnäytetyön toimeksiantajana toimii Younite-AI. Yrityksessä on jo vuosia kehitetty asiakasprojektina terveyspalveluihin kuuluvaa ohjelmistotuotetta, johon kuuluivat verkkosovellus, Android-sovellus sekä iOS-sovellus. Sovellusten testausta on automatisoitu yksikkötestien ja integraatiotestauksen osalta, mutta sovellusten käyttöliittymän kautta tapahtuva toiminnallinen testaus on koko projektin aikana tapahtunut lähes kokonaan manuaalisesti. Ohjelmistotuote on kuitenkin kasvanut vuosien aikana niin laajaksi, että tuotteen kehitystiimillä on tarve automaattisen testaustyökalun käyttöönotolle testauksen nopeuttamiseksi ja helpottamiseksi erityisesti käyttöliittymän kautta tapahtuvien, toiminnallisten päästä päähän -testien suorittamiseen. Automaattinen testaustyökalu nopeuttaa ja tehostaa ohjelmiston kehitystä ja vapauttaa keskittymään manuaalisessa testauksessa vain vaativimpiin ja spesifisimpiin testitapauksiin.

Tämän toiminnallisen opinnäytetyön tavoitteena on tutustua testaamiseen ja testausautomaatioon yleisesti, sillä aihealue yleensäkin on minulle uusi. Opinnäytetyön tarkoituksena on myös tutustua eri automaattisen testauksen työkaluihin, sekä niiden ominaisuuksiin. Varsinaisena

tutkimustehtävänä on kolmen työkalun evaluointi sopivimman automaatiotyökalun löytämiseksi projektin käyttöön. Evaluoinnin vaatimukset päätetään projektin tarpeiden mukaan yhteistyössä kehitystiimin kanssa, ja evaluointi suoritetaan tutustumalla eri työkalujen dokumentaatioon sekä tutustumalla työkaluihin käytännössä. Opinnäytetyön lopullisena tavoitteena on oman henkilökohtaisen oppimisen lisäksi, että se toimisi mahdollisena pohjana ja tukena projektin kannalta sopivimman työkalun löytämiselle ja valinnalle.

2 OHJELMISTOTESTAUS

Ohjelmistotestauksella tarkoitetaan menetelmiä, joiden avulla varmistutaan, että kehitetty ohjelmisto tai ohjelmistojärjestelmä täyttää sille asetetut vaatimukset ja ettei siinä ole virheitä tai puutteita (Hamilton 2023b). Ohjelmistotestaus toteutetaan joko manuaalisesti tai automaattisesti suorittamalla ohjelmiston tai ohjelmistojärjestelmän eri osa-alueita. Manuaalisessa testauksessa joko yksittäinen testaaja tai testaustiimi käyttää testattavaa ohjelmistoa ja varmistuu sen toimivuudesta. Automaattisessa testauksessa taas käytetään eri työkaluja tai menetelmiä, joilla voidaan luoda erilaisia testejä simuloimaan niitä samoja toimia, jotka ihmiset suorittaisivat manuaalisessa testauksessa. (Atlassian 2023.) Ohjelmistosta mahdollisesti löytyvien virheiden ja puutteiden lisäksi ohjelmistotestauksella on tarkoitus myös selvittää, että ohjelmistolla tai järjestelmällä on ne ominaisuudet, joita sillä on suunniteltukin olevan (Hamilton 2023b).

Ohjelmistotestaus on tärkeä osa ohjelmiston kehitystä, sillä sen ansiosta mahdolliset virheet onnistutaan havaitsemaan ja korjaamaan ennen ohjelmiston päätymistä loppukäyttäjän, kuten asiakkaan, käyttöön. Ohjelmointivirheet eivät aiheuta kehittäjälle ainoastaan turhia kuluja, vaan saattavat pahimmassa tapauksessa olla jopa vaarallisia, kuten esimerkiksi virhe lentokoneen ohjelmistossa. (Hamilton 2023b.)

Ohjelmistotestaus on taloudellisesti järkevää, sillä kehittäjä säästää ohjelmistoa testaamalla kehitykseen kuluvaa aikaa ja rahaa. Ennalta määritetyt testitapaukset ohjaavat sovelluksen kehittäjiä oikeaan suuntaan jo sovelluksen kehitysvaiheessa, mikä nopeuttaa kehitysprosessia ja auttaa välttämään virheitä. Täten myös kehityskulut alenevat.

Testaamalla saadaan lisäksi varmistettua, ettei jo olemassa olevat ominaisuudet lakkaa toimimasta, kun sovellusta edelleen kehitetään. (Atlassian 2023.) Tätä niin kutsuttua regressiotestausta tuleekin tehdä jatkuvasti ja aina, kun ohjelmiston koodissa tapahtuu joitain muutoksia (Hamilton 2023a). Ohjelmistotestauksella parannetaan myös tietoturvaa. Ennen kaikkea ohjelmiston kehityksessä on erityisen tärkeää ottaa huomioon myös tuotteen loppukäyttäjä, joten testaamalla käyttöliittymää ja käyttökokemusta saadaan testauksella parannettua myös käyttäjätyytyväisyyttä. (Hamilton 2023b.)

Ohjelmistotestaus jaetaan eri tasoihin. Jaottelu voi tapahtua hieman eri lailla ja eri nimillä riippuen käytetystä lähteestä. (Katara 2021; Hamilton 2023b; Atlassian 2023.) Esimerkkejä erityyppistä testeistä ovat muun muassa yksikkötestaus, integraatiotestaus, systeemitestaus, alfatestaus ja beetatestaus (Katara 2021). Atlassianin (2023) jaottelun mukaan testauksen perustasoja ovat muun muassa yksikkötestaus, integraatiotestaus ja päästä päähän -testaus.

2.1 Yksikkötestaus

Testauksen alinta tai pohjimmaista tasoa kutsutaan yksikkötestaukseksi (unit testing) (Atlassian 2023). Yksikkötestiä voidaan kuvata testiksi, jolla testataan vain pientä osaa toimivaa ohjelmistoa eli yksikköä. Yksikkötestit voidaan suorittaa nopeasti ja eristettyinä ohjelmiston kokonaisuudesta. (Khorikov 2021, Luku 2.1.)

Yksikkötestejä käytetään testaamaan yksittäisiä toimintoja tai tehtäviä, kuten esimerkiksi ohjelmointifunktiota, joille on määritetty tietty odotettu lopputulos. Yksikkötestissä testattavalle toiminnolle annetaan simuloituna syöte, jonka lopputuloksen tulee vastata odotettua tulosta. (Atlassian 2023.) Sovelluskehittäjät yleensä suorittavat yksikkötestit samalla, kun he kehittävät ohjelmistoa. Täten testien osoittamat virheet pystytään korjaamaan heti, jos niitä ilmenee. (Software Testing Fundamentals 2022b.)

2.2 Integraatiotestaus

Seuraavaa tasoa kutsutaan integraatiotestaukseksi (integration testing). Integraatiotesteiksi kutsutaan testejä, jotka koostuvat useammasta kuin yhdestä testattavasta osasta tai yksiköstä. (Atlassian 2023.) Yksinkertaisimmillaan integraatiotestausta voidaan käyttää testaamaan yksittäisten yksikköjen välistä toimintaa keskenään tai laajemmin kokonaisten systeemien välistä keskinäistä toimintaa (Software Testing Fundamentals 2022a).

Integraatiotestaus tarkoittaa esimerkiksi sovelluksen ja tietokannan välisen yhteyden toiminnan testaamista (Pittet 2023). Ohjelmoijat suorittavat komponenttien välisen integraatiotestauksen usein heti yksikkötestauksen perään. Integraatiotestit voidaan myös suorittaa automaatiolla esimerkiksi jatkuvan integraation (Continuous Integration, CI) palveluntarjoajan kautta. Testaajat

suorittavat systeemien välisen integraatiotestauksen yleensä komponenttien integraatiotestauksen jälkeen. (Software Testing Fundamentals 2022a.)

2.3 Päästä päähän -testaus

Seuraavaa tasoa kutsutaan päästä päähän -testaukseksi (end-to-end) tai toiminnalliseksi testaukseksi (functional testing) (Atlassian 2023). Päästä päähän -testauksella sovelluksen toimintaa testataan huomioimalla laajasti eri näkökulmat. Testausolosuhteiden, kuten esimerkiksi tietokannasta tulevien tietojen, tulee olla mahdollisimman samankaltaisia kuin oikeassa käyttöympäristössä. Testauksen tavoitteena onkin simuloida mahdollisimman tarkasti käyttäjän tosielämässä suorittamia, sovelluksen käyttökaaren eri vaiheissa tapahtuvia toimia. (Smartbear 2023.) Yksinkertainen esimerkkitapaus päästä päähän -testille on käyttäjän kirjautuminen sovellukseen tai palveluun syöttämällä käyttäjätunnus ja salasana ja painamalla kirjautumispainiketta.

Päästä päähän -testien tarkoituksena on varmistua, että ohjelmisto tai palvelu suoriutuu alusta loppuun kaikista käyttäjän suorittamista toimista odotetusti ja vaaditulla tasolla (Pittet 2023). Lisäksi varmistutaan myös siitä, että kaikki ohjelmistoon liitetyt alijärjestelmät, kuten esimerkiksi käyttöliittymä, ohjelmointirajapinnat ja ulkoiset tietokannat, toimivat oikein ja halutulla tavalla. Päästä päähän -testaus suoritetaan yleensä sovelluksen jokaisen päivityksen yhteydessä. Täten mahdolliset virheet saadaan havaittua ja korjattua nopeasti, ja virheiden mahdollinen pääsy ohjelmiston julkaistuun versioon on epätodennäköisempää. Näin ollen myös ohjelmiston julkaisutahtia voidaan nopeuttaa, mikä tehostaa sovelluskehitystä sekä vähentää kehityskustannuksia. (Smartbear 2023.)

Testaaja voi tehdä päästä päähän -testausta manuaalisesti, tai testit voidaan automatisoida. Mitä laajemmaksi testattava sovellus kasvaa, sitä hankalampaa ja enemmän aikaa vievää manuaalisesta testaamisesta tulee, joten testien automatisointi mahdollisimman laajasti on järkevää. (Schmitt 2022.) Tällöin vapautuu resursseja sovelluksen kehittämiseen, ja testaajat voivat keskittyä testaamaan manuaalisesti kaikkein haastavimpia tapauksia (Rehkopf 2023a).

3 TESTAUSAUTOMAATIO JA EVALUOITAVAKSI VALITUT TYÖKALUT

Testausautomaatiolla tarkoitetaan niitä menetelmiä, jotka mahdollistavat manuaalisesti tehtävän testauksen suorituksen koneellisesti. Testausautomaatiolla ei tarkoiteta pelkästään yksittäisen testin tai testisarjan suoritusta vaan kokonaista kokoelmaa niitä työkaluja, joita automaattisten testien suunnitteluun, suoritukseen, arviointiin ja raportointiin tarvitaan. (Baumgartner ym. 2022.)

Mikäli ohjelmistotestaus suoritetaan manuaalisesti, tarvitaan sen tekemiseen siihen erikoistunut testaaja tai kokonainen testaustiimi. Testaajien täytyy manuaalisesti suorittaa kaikki ennalta suunnitellut testitapaukset ohjelmiston muokkauksen tai päivityksen yhteydessä. Tämän jälkeen testien tulokset tulee erikseen esitellä sovelluskehittäjille, joiden tulee tarvittaessa korjata tai muokata ohjelmistoa testitulosten perusteella. Manuaalinen testaus on prosessina kallis, hidas sekä sisältää suuren virheiden mahdollisuuden. Testausautomaatio lisää koko kehitysprosessin tehokkuutta ja vähentää sen kustannuksia. Testaus voidaan esimerkiksi jatkuvan integraation (CI) työkalujen avulla automatisoida tapahtumaan jatkuvasti sovelluskehityskehityksen edetessä. Täten sovelluksen kehittäjät löytävät mahdolliset virheet ja pystyvät korjaamaan ne jo sovelluksen kehitysvaiheessa. Lisäksi varsinaiset testaajat pystyvät keskittymään pelkästään haastavimpiin testitapauksiin, mikä myös tehostaa toimintaa. (Rehkopf 2023a.)

Automaattisen testaukseen on olemassa hyvin laaja valikoima eri työkaluja (Tricentis 2016). Yhtenä haasteena oli löytää ja valita työkaluista sopivat vaihtoehdot evaluointia varten. Lopulta vaihtoehdoiksi valikoituivat projektin kehitystiimin ehdotuksesta Cypress ja Testim. Kolmanneksi työkaluksi haluttiin valita sellainen, joka ainakin alustavasti olisi potentiaalisesti mahdollisimman sopiva työkalu projektin tarpeisiin. Ei olisi järkevää evaluoida työkalua, joka ei pintapuolisen tarkastelun perusteella täyttäisi työkalulta vaadittuja kriteerejä. Tämän pintapuolisen esivalinnan perusteella vaikutti työkalu nimeltä Testsigma sopivan hyväksi evaluaatiokohteeksi. Näin ollen Testsigma päätyikin kolmanneksi evaluoitavaksi työkaluksi.

3.1 Cypress

Projektin kehitystiimin ehdotuksesta ensimmäiseksi evaluoitavaksi työkaluksi valikoitui Cypress. Cypress on nykyaikainen, avoimen lähdekoodin testausautomaatiotyökalu, joka soveltuu

työkaluksi erityisesti modernien JavaScript-ohjelmistokehyksiä käyttävien verkkosovellusten testaamiseen. Cypress-työkalulla pystytään toteuttamaan ennen kaikkea automaattisia päästä päähän -testejä, mutta se soveltuu myös yksikkötestaukseen, integraatiotestaukseen sekä funktionaaliseen testaukseen. Cypress-ohjelmisto on mahdollista integroida muihin palveluihin. (Cypress 2023j.)

3.2 Testim

Toiseksi evaluoitavaksi työkaluksi valikoitiin projektin kehitystiimin ehdotuksesta Testim. Testim on Tricentis-yrityksen omistama testausautomaatiotyökalu (Tricentis 2023). Testim-työkalu on tarkoitettu erityisesti pilviympäristössä toimivien verkkosovellusten ja mobiilisovellusten päästä päähän -testaukseen (Testim 2023f).

Alustan tarkoituksena on muun muassa auttaa sovelluskehittäjiä tehostamaan ja nopeuttamaan mobiili- ja/tai verkkosovellustensa kehitystyötä tarjoamalla helpon ja joustavan tavan testaamiseen. Testim-työkalun käyttö on suunniteltu helpoksi käyttää ilman laajempaa tietämystä tai kokemusta, mutta se mahdollistaa, että kokeneemmat käyttäjät pystyvät luomaan myös haastavampia testitapauksia. (Testim 2023f.) Testim on laajasti integroitavissa useiden eri kolmannen osapuolen palvelun kanssa (Tricentis 2023).

3.3 Testsigma

Kolmanneksi työkaluksi tutkittiin pintapuolisesti eri vaihtoehtoja. Tämän esivalinnan ajatuksena oli valita evaluoitavaksi työkaluksi sellainen, joka potentiaalisesti täyttäisi mahdollisimman hyvin työkalulta vaadittuja kriteerejä. Tarkastelun perusteella valikoitui kolmanneksi evaluoitavaksi työkaluksi Testsigma, joka on avoimen lähdekoodin testausautomaatiotyökalu. Testsigma-työkalu on tarkoitettu helposti käytettäväksi työkaluksi, jolla voidaan luoda nopeasti testejä verkkosovellusten, mobiilisovellusten ja ohjelmointirajapintojen (API) testaukseen. (Testisigma 2023.)

Työkalun tarkoituksena on mahdollistaa automaattisten testien luonti ilman vaativaa käyttöönottoa tai laitteistoa. Testsigma-työkalulla testien luomisen on tarkoitus olla nopeaa ja helppoa ja mahdollista ilman ohjelmointitaitoja. Testsigma-työkalu mahdollistaa kuitenkin tarvittaessa myös

vaativampien, personoitujen testien luonnin ohjelmoimalla. Testsigma on integroitavissa useisiin kolmannen osapuolen palveluihin ja työkaluihin. (Testsigma 2023g.)

4 TESTAUSTYÖKALUN VALINTAPERUSTEET

Sovelluksen kehitystiimissä käytyjen keskustelujen pohjalta oli mietitty 12 evaluaatiovaatimusta, jotka työkalun tulee täyttää kokonaan tai ainakin osittain. Kyseisten vaatimusten oli tarkoitus lopulta tukea ja antaa pohja sopivimman työkalun valinnalle tiimin käyttöön. Osa vaatimuksista oli lopullisen valinnan kannalta ehdottomampia kuin toiset. Vaatimukset käyttöympäristöistä ja testien uudelleenkäytettävyydestä olivat ehdottoman tärkeitä työkalun valinnan kannalta. Esimerkiksi se, että työkalu ei täytä vaatimusta mahdollisuudesta visuaaliseen vertailuun, ei todennäköisesti yksistään johda työkalun hylkäämiseen, jos suurin osa muista vaatimuksista täyttyy. Jos taas työkalu ei täytä vaatimusta vaadituista käyttöympäristöistä, joudutaan työkalu hylkäämään, vaikka se muuten osoittautuu sopivaksi.

Vaaditut käyttöympäristöt

Asiakasprojektin ohjelmistotuotteisiin kuuluvat verkko-, Android-, ja iOS-sovellus. Työkalulla tulee pystyä luomaan ja suorittamaan testejä kaikilla kolmella alustalla. Mikäli työkalulla ei ole suoraa tukea kaikille alustoille, on minimivaatimuksena, että jonkin lisäosan tai lisäohjelmiston avulla saadaan tuki kaikille alustoille lisättyä. Tämän vaatimuksen oli täytyttävä, jotta työkalu voitaisiin valita projektin käyttöön.

Helppokäyttöisyys

Valittavan työkalun käytön tulee olla mahdollisimman helppoa ja sujuvaa. On testauksen kannalta tehokasta, jos kuka tahansa kehitystiimin jäsen pystyy käyttämään työkalua. Täten on eduksi, mikäli valittavalla työkalun käyttäminen onnistuu helposti eikä vaadi syvällistä perehtymistä työkalun toimintaan. Lisäksi on hyvä, mikäli testien luominen työkalulla tapahtuu vain vähän (low-code) tai ei ollenkaan (no-code) ohjelmointia vaativilla menetelmillä. Mikäli työkalu on riittävän helppo oppia ja käyttää, on mahdollista esimerkiksi ottaa lyhyeksi aikaa projektin ulkopuolista henkilöstöä suorittamaan testausta.

Käyttöönoton helppous

Työkalun käyttöönoton tulee olla helppoa ja nopeaa. Käyttöönoton helppouteen vaikuttaa se, vaaditaanko esimerkiksi työkalun käyttämiseksi useiden ohjelmistojen asennusta vai tapahtuuko työkalun käyttö esimerkiksi suoraan verkossa selaimen käyttöliittymän kautta. Lisäksi helppouteen

vaikuttaa, millainen dokumentaatio käyttöönoton tueksi on olemassa ja miten hyvin dokumentaatio tukee käyttöönottoa.

Testien nopea luonti

Työkalulla tulee pystyä luomaan uusia testejä nopeasti ja vaivattomasti. Tämän vaatimuksen osalta ei testien luontitavalla ole merkitystä, vaan luonti voi tapahtua ohjelmoimalla tai ilman ohjelmointia. Evaluaatiovaatimuksen kannalta tärkeämpää onkin, että testien luontien aloitus olisi vaivatonta ja testien kirjoitus nopeaa ja intuitiivista. Lisäksi toteutumiseen vaikuttaa, millainen dokumentaatio testien luomisen tueksi löytyy.

Testien uudelleenkäytettävyys

Työkalulla luotuja testejä tai testin osia tulee pystyä käyttämään laajasti uudelleen, ilman että sama testi luodaan kokonaan erikseen jokaista testitapausta varten. Esimerkkinä voidaan luoda testi, jossa käyttäjä kirjautuu onnistuneesti palveluun. Tätä aiemmin luotua testiä taas tulee voida käyttää uudelleen osana toista testiä, jossa käyttäjä esimerkiksi ensin kirjautuu palveluun ja suorittaa kirjauduttuaan jonkin toisen toiminnon. Vaatimuksen testien uudelleenkäytettävyydestä tulee täyttyä, jotta työkalu voidaan valita projektin käyttöön.

Testien vakaus

Työkalulla luotujen testien tulee olla vakaita (robust). Testien tulee epäonnistua vain, mikäli virhe esiintyy jossain kohdassa, mitä testin tuleekin tarkistaa. Epäonnistuminen ei siis saa johtua mistään muusta, kuten esimerkiksi testausympäristöstä, muutoksista datassa, jostakin satunnaistapahtumasta tai ylipäänsä mistään varsinaisen testin ulkopuolisesta tekijästä. (Chahine 2020.) Testejä ei voida pitää vakaina, mikäli esimerkiksi jokaisen testattavan ohjelmiston päivityksen tai muutoksen jälkeen myös testejä joudutaan muokkaamaan tai päivittämään. Testien vakautta voidaan myös tarkastella siltä kantilta, miten niiden päivitys tai muokkaus onnistuu. Jos esimerkiksi samaa testauskomponenttia käytetään useissa eri testeissä, ei voida puhua vakaista testeistä, mikäli jokaista testiä joudutaan muokkaamaan tai korjaamaan erikseen. Vakaiden testien kohdalla riittää, että muutos tehdään vain jaettuun komponenttiin, jolloin korjaus toteutuu automaattisesti myös muissa komponenttia käyttävissä testeissä.

Ajastettujen testien mahdollisuus

Työkalussa tulee olla mahdollista ajastaa testien suoritus tapahtumaan automaattisesti. Testit tulee esimerkiksi pystyä ajastamaan tapahtumaan joka yö tiettyyn kellonaikaan. Evaluaatiovaatimuksen toteutumiseksi ei välttämättä vaadita, että työkalussa on olemassa suoraan käyttöliittymä, jossa testejä voidaan ajastaa, vaan testien ajastus voidaan suorittaa myös esimerkiksi integroimalla työkalu jatkuvan integraation- eli CI-palvelun kanssa (Rehkopf 2023b).

Testitulosten raportointi kolmannen osapuolen työkaluun

Työkalusta tulee löytyä tuki sille, että se pystytään integroimaan johonkin kolmannen osapuolen työkaluun, johon suoritettujen testien raportit voidaan lähettää automaattisesti. Raportit tulee pystyä ainakin lähettämään tiettyyn sähköpostiosoitteeseen. Paras vaihtoehto on mahdollisuus integraatioon jonkin pikaviestintäohjelman, kuten esimerkiksi Slackin, kanssa. Näin raportit voidaan julkaista suoraan kyseisessä palvelussa.

Mahdollisuus ohjelmoida omia testejä

Työkalulla tulee olla mahdollista luoda testejä ohjelmoimalla. Tämä koskee ennen kaikkea työkaluja, joissa testien luonti lähtökohtaisesti tapahtuu esimerkiksi nauhoittamalla (recording) käyttäjän toimia ilman ohjelmointia tai vain vähän ohjelmointia käyttäen. On hyvä, että työkalulla pystyy tarvittaessa itse ohjelmoimaan eri testejä käyttäen JavaScript-, Python- tai jotakin muuta ohjelmointikieltä.

Mahdollisuus visuaaliseen vertailuun

Työkalulla tulee pystyä tekemään käyttöliittymän visuaalista vertailua. Visuaalisessa vertailussa määritetään testiä varten kuva, jolta käyttöliittymän tulee näyttää ja johon käyttöliittymää testien aikana verrataan. Esimerkiksi mobiilisovellusten kohdalla tämä on erityisen hyödyllinen ominaisuus. Useiden erilaisien mobiililaitteitten takia saattaa eri kokoisilla ruuduilla esiintyä käyttöliittymän komponenteissa muutoksia esimerkiksi niiden sijoittumisessa vääriin kohtiin. Tämän kaltaiset virheet voidaan saada selville visuaalisen vertailun avulla.

Mahdollisuus CI-integraatioon/versionhallinnan integraatioon

Työkalu tulee pystyä integroimaan joko jatkuvan integraation (CI) palveluun, kuten Circle CI:hin, versionhallinnan palveluun, kuten Bitbucketiin, tai molempiin. Integrointi kyseisiin palveluihin mahdollistaa muun muassa testien automatisoinnin esimerkiksi aina, kun koodissa tapahtuu ohjelmointimuutoksia tai kun ohjelmistosta julkaistaan uusi versio (Rehkopf 2023b.)

Maksuttomuus

Yhtenä vaatimuksena valittavalle työkalulle oli sen ilmoitettu hinta. Työkalut joko ovat tai eivät ole maksullisia. Työkalun hintaa täytyy kuitenkin lopullisesti arvioida paljon laajemmin. Maksuton työkalu voi esimerkiksi olla maksullista työkalua huomattavasti työläämpi tai hitaampi käyttää, jolloin maksullinen työkalu saattaa itse asiassa tulla lopulta kustannustehokkaammaksi, kun työkalun käyttöön ja testien tekoon käytetään vähemmän työaikaa. Ilmaisesta työkalusta voi olla saatavilla myös maksullinen versio, jossa on enemmän ominaisuuksia ja jolla on parempi käytettävyys. Vaikka evaluointia varten päädyttiinkin yksinkertaisesti arvioimaan, onko työkalu maksuton vai ei, tulee työkalun lopullista valintaa tehtäessä hintaa tarkastella monipuolisemmin.

5 VALITTUJEN TESTAUSTYÖKALUJEN EVALUOINTI

Kukin työkalu evaluoitiin yksitellen, ja jokaisen kohdalla käytiin läpi kaikki edellä määritellyt evaluaatiovaatimukset. Jokaisen työkalun osalta tutustuttiin dokumentaatioon sekä kyseisen työkalun ominaisuuksiin ja mahdollisuuksiin. Näitä verrattiin työkalun evaluaatiovaatimuksiin ja tehtiin johtopäätökset haluttujen vaatimusten täyttymisestä. Vaatimusten voitiin todeta täyttyneen kokonaan tai osittain tai jääneen kokonaan täyttymättä.

Jokaisesta evaluoitavasta työkalusta hankittiin kokeiluversio, jotta sitä päästiin käytännössä testaamaan. Kaikille työkalun tukemille alustoille alustettiin projekti, jolle luotiin yksinkertainen testi käyttäjän sisäänkirjautumiselle testattavaan sovellukseen. Verkkosovelluksen testissä käyttäjä syöttää sähköpostiosoitteen ja salasanan niille varattuihin kenttiin, jonka jälkeen klikkaa painiketta sisäänkirjautumiseksi. Onnistuneen sisäänkirjautumisen jälkeen kirjautuneen käyttäjän nimen tulee näkyä sivulla. Mobiilialustojen testissä käyttäjä syöttää sähköpostiosoitteen ja näpäyttää kirjautumispainiketta. Tämän jälkeen käyttäjä syöttää 6-numeroisen PIN-koodin sekä varmistaa PIN-koodin syöttämällä sen toiseen kertaan. Tämän jälkeen käyttäjä hylkää sormenjälkitunnistuksen käyttöönoton kyselyn, mikäli sen käyttöönottoa ehdotetaan. Tämä riippuu siitä, tukeeko testauslaite sormenjälkitunnistusta. Tämän jälkeen ohjelmisto ilmoittaa ilmoitusten käyttöönotosta ja käyttäjä näpäyttää continue-painiketta jatkaakseen sisäänkirjautumisen loppuun. Testin lopuksi tulee sisään kirjautuneen käyttäjän nimen löytyä sivulta.

Valtaosaan valintaperusteista pystyttiin löytämään vastaus tutustumalla työkalujen ominaisuuksiin dokumentaation avulla, sekä testaamalla työkalua käytännössä. Vaatimusta testien kirjoittamisen helppoudesta evaluoitiin luomalla yksinkertaiset sisäänkirjautumisen testit kaikille työkalun tukemille alustoille ja testaamalla, miten testin suoritus onnistui. Jokaisen työkalun kohdalla kirjattiin taulukkoon, miten työkalu täytti kunkin vaaditun valintaperusteen.

Evaluaatioprosessin käytännön testauksiin käytettiin eri laitteita ja työkaluja. Testauslaitteena verkkosovellusten osalta toimi kannettava tietokone MacBook Pro (macOS Monteterey 12.6.3). Verkkosovelluksen testaukseen paikallisella tietokoneella käytettiin Chrome-selainta. Ohjelmointiympäristönä toimi Visual Studio Code. Mobiilisovellusten paikalliseen testaukseen käytettiin fyysistä Android-laitetta ja fyysistä iOS-laitetta. Testaukseen käytetty Android-laite oli

Samsung Galaxy S21FE, jonka Android-versio oli 13. Testaukseen käytetty iOS-laite oli iPhone 13 Pro, jonka iOS-versio oli 16.3.1.

Kun kaikki kolme työkalua oli evaluoitu, verrattiin evaluoinnin tuloksia keskenään. Tulosten perusteella voitiin valita evaluaation perusteella sopivin ehdokas, jota voitiin esittää lupaavaksi vaihtoehdoksi projektin testaustyökaluksi.

5.1 Cypress-työkalun evaluointi

Cypress-työkalu arvioitiin työkalun valinnalle asetettujen evaluointivaatimusten perusteella. Arviointi tapahtui tutustumalla ohjelmiston dokumentaatioon sekä hankkimalla itse ohjelmisto, jotta päästiin tekemään yksinkertaisia testejä ja siten kokeilemaan ohjelmistoa käytännössä. Cypress-työkalu evaluoitiin jokaisen vaatimuksen osalta ja tulokset kirjattiin taulukkoon. Evaluoinnin tulokset on esitetty taulukossa 1.

TAULUKKO 1. Cypress-työkalun evaluoinnin tulokset

Evaluoinnin vaatimus	Evaluoinnin tulos
Vaaditut käyttöympäristöt	Ei
Helppokäyttöisyys	Osittain
Käyttöönoton helppous	Kyllä
Testien nopea luonti	Kyllä
Testien uudelleenkäytettävyys	Kyllä
Testien vakaus	Osittain
Ajastettujen testien mahdollisuus	Kyllä
Testitulosten raportointi kolmannen osapuolen sovellukseen	Kyllä
Mahdollisuus ohjelmoida omia testejä	Kyllä
Mahdollisuus visuaaliseen vertailuun	Osittain
Mahdollisuus CI-integraatioon/versionhallinnan integraatioon	Kyllä
Maksuttomuus	Kyllä

Vaaditut käyttöympäristöt

Cypress-työkalu oli tarkoitettu selainpohjaisten sovellusten testaamiseen (Cypress 2023j). Sillä voitiin testata eri sovelluksia eri verkkoselaimilla. Cypress-työkalusta löytyi tuki Chrome-perheen

selaimille, muun muassa Chromelle, Edgelle ja Electronille sekä WebKitille (Safari) (Cypress 2023c). Mobiililaitteiden osalta se ei soveltunut kuin pelkästään selainpohjaisten sovellusten testaamiseen (Cypress 2023j). Tämä oli projektimme tavoitteiden kannalta hankalaa, sillä asiakasprojektin ohjelmistosta oli käytössä verkkosovelluksen lisäksi myös natiivit iOS- ja Android-sovellukset. Vaadittujen käyttöympäristöjen evaluointivaatimus ei siis toteutunut Cypress-työkalun osalta.

Helppokäyttöisyys

Cypress-työkalulla testit luotiin ohjelmoimalla JavaScript-ohjelmointikielellä. Testien ohjelmointiin käytettiin itse valittua ohjelmointiympäristöä, kuten esimerkiksi Visual Studio Codea. (Cypress 2023l.) Vaikka kaikki testit tulikin tehdä ohjelmoimalla, Cypress-työkalulle oli olemassa lisätyökaluja ohjelmoinnin helpottamiseksi. Työkalu sisälsi vielä kokeiluvaiheessa olevan lisätyökalun, Cypress Studion, jolla oli mahdollista luoda testejä nauhoittamalla käyttäjän toimia. Esimerkkinä voitiin luoda sisäänkirjautumisen testi verkkosovellukseen, jolloin Cypress Studio nauhoitti ja loi automaattisesti jokaista käyttäjän toimea vastaavan koodin. (Cypress 2023e.)

Cypress-työkalussa oli myös mahdollista hyödyntää Chromeen asennettavaa, kolmannen osapuolen tekemää laajennusta, Cypress Recorderia. Cypress Recorderilla voitiin myös nauhoittaa käyttäjän toimia, jotka aputyökalu muutti koodimuotoon. Luodut koodit voitiin jälkeenpäin kopioida sellaisenaan testiin. (Campbell ym. 2021.) Vaikka Cypress-työkalulla olikin mahdollista luoda testejä ohjelmoinnin lisäksi myös nauhoittamalla, oli tämä toiminnallisuus vasta kokeiluasteella tai sitten se vaati kolmannen osapuolen lisäosan toimiakseen. Tästä syystä Cypress-työkalu täytti evaluaatiovaatimuksen työkalun helppokäyttöisyydestä vain osittain.

Käyttöönoton helppous

Cypress-työkalun käyttöönottoa varten tuli sen ohjelmisto asentaa paikallisesti omalle tietokoneelle. Cypress asennettiin testattavaan projektiin, jonka sisällä se toimi niin kutsuttuna kehitystyöriippuvuutena (dev dependency). Seuraamalla Cypress-työkalun dokumentaatiota käyttöönotto sujui vaivattomasti ja helposti. (Cypress 2023g.) Työkalulla pystyi heti käyttöönoton jälkeen alkaa luomaan testejä. Osan toiminnoista, kuten esimerkiksi raportoinnin, käyttöön ottamiseksi, tuli luoda tunnukset Cypress Cloud -palveluun. Cypress Cloud -palveluun kirjautuminen onnistui helposti ja vaivattomasti dokumentaation avulla. (Cypress 2023d.) Selkeän dokumentaation ansiosta sekä testaamalla Cypress-työkalun asennusta ja käyttöönottoa

käytännössä voitiin kyseisen työkalun todeta täyttäneen evaluointivaatimuksen käyttöönoton helppoudesta.

Testien nopea luonti

Testien luonnin aloittaminen ja niiden suoritus Cypress-työkalulla osoittautui melko vaivattomaksi, ja molempiin löytyi hyvä dokumentaatio. Dokumentaation avulla pystyttiin nopeasti luomaan testausta varten yksinkertainen testi käyttäjän sisäänkirjautumiselle asiakasprojektin verkkosovellukseen (liite 1). (Cypress 2023k.) Testit Cypress-työkalulla kirjoitettiin käyttäen JavaScript-ohjelmointikieltä. Lisäksi jo aiemmin mainittu lisätyökalu, Cypress Studio, saattoi auttaa nopeuttamaan testien kirjoitusta entisestään. Sen avulla käyttäjän toimia pystyttiin nauhoittamaan, jolloin Cypress Recorder loi automaattisesti toimia vastaavan koodin. Luotuja testejä voitiin suorittaa paikallisesti suoraan Cypress-ohjelmiston paneelista (Cypress 2023i). Testien suoritus onnistui myös komentokehotteen (CLI) kautta (Cypress 2023b). Testit voitiin myös linkittää tiettyyn Cypress Cloudissa luotuun projektiin. Näin testien tulokset tallentuivat Cypress Cloudiin, missä niitä voitiin myöhemmin tarkastella ja tuloksia analysoida. (Cypress 2023d.) Dokumentaation ja käyttökokemuksen perusteella voitiin todeta, että Cypress-työkalu täytti evaluointiehdon testien nopeasta luonnista.

Testien uudelleenkäytettävyys

Cypress-työkalulla luodut testi olivat laajasti uudelleen käytettäviä. Yksinkertaisimmillaan samaa kirjoitettua koodia voitiin kopioida ja liittää useaan eri testitapaukseen. Työkalulla oli mahdollista luoda omia, muokattuja komentoja (Command). Luotuja komentoja voitiin käyttää useissa eri testeissä. (Cypress 2023l.) Esimerkiksi edellä luotu sisäänkirjautumisen testi (liite 1) voitiin luoda login-komennoksi, jolle syötettiin parametreina haluttu käyttäjätunnus ja salasana. Login-komentoa voitiin siten käyttää eri testien yhteydessä tai osana toista testiä. Esimerkkinä voisi olla testi, jossa käyttäjä ensin kirjautuu palveluun ja sen jälkeen suorittaa jonkin muun halutun toiminnon. Käytännön testauksen perusteella voitiinkin siis todeta, että Cypress-työkalu täytti evaluointiehdon testien uudelleenkäytettävyydestä.

Testien vakaus

Testeissä tapahtuvissa toiminnoissa, kuten esimerkiksi painikkeen painalluksessa, tarvitaan toiminnan kohteena olevalle elementille paikkaviite eli lokaattori. Lokaattorin avulla toiminta voidaan kohdentaa oikeaan käyttöliittymän elementtiin. Cypress-työkalulla luoduissa testeissä viitattiin eri käyttöliittymän elementteihin yksittäisillä ominaisuuksilla, kuten esimerkiksi elementin

id:llä, nimellä tai CSS-luokalla. Esimerkiksi sisäänkirjautumisen testissä (liite 1) oli Submit-painikkeen paikkaviitteenä eli lokaattorina, painikkeen id. Koska lokaattoreina käytettiin vain yksittäisiä ominaisuuksia, olivat testit mahdollisesti alttiita epäonnistumisille, mikäli kyseisissä lokaattoreissa tapahtuisi joitakin muutoksia. Jos esimerkiksi painikkeen id muuttuisi, epäonnistuisi sen painallus testiä suorittaessa. Pelkästään yhden ominaisuuden käyttö Cypress-työkalulla luoduissa testeissä saattoikin täten mahdollisesti heikentää testien vakautta. Oli kuitenkin myös todettava, että mahdollisten muutosten aiheuttamat virheet eivät kuitenkaan olleet erityisien työläitä korjata. Koska testeihin pystyttiin luomaan uudelleen käytettäviä komentoja, joita voitiin käyttää useissa testeissä, riitti, että mahdollinen korjaus tehtiin vain komennon koodiin. Täten virhe korjautui automaattisesti kaikissa kyseistä komentoa käyttävissä testeissä. Vaikka testit näin ollen olivatkin mahdollisesti alttiita epäonnistumaan, oli mahdollisten virheiden korjaaminen hyvän uudelleenkäytettävyyden vuoksi melko helppoa korjata. Tämän perusteella voitiin todeta, että Cypress-työkalu täytti osittain evaluointivaatimuksen testien vakaudesta.

Ajastettujen testien mahdollisuus

Integroimalla Cypress-työkalu johonkin jatkuvan integraation (CI) palveluun testit voidaan ajastaa tapahtumaan automaattisesti esimerkiksi aina silloin, kun koodissa tapahtuu muutoksia. CI-palveluissa kyetään luomaan myös ajastettuja toimintoja, joten täten integraation avulla voitiin testitkin ajoittaa tapahtumaan automaattisesti tiettynä aikana (Rehkopf 2023b). Itse Cypress-työkalussa ei ollut sisäänrakennettua ratkaisua testien ajastamiseen, mutta se voitiin kuitenkin toteuttaa CI-integraation kautta. Todettiin, että Cypress-työkalu täytti evaluaatiovaatimuksen ajastettujen testien mahdollisuudesta.

Testitulosten raportointi kolmannen osapuolen sovellukseen

Cypress-työkalu sisälsi tuen integraatiolle useisiin eri palveluihin (Cypress 2023d). Myös esimerkiksi Slack-pikaviestintäohjelma kuului integroitaviin ohjelmistoihin, joten testien tulosten automaattinen raportointi suoraan tietylle Slack-kanavalle voitiin tarvittaessa toteuttaa. (Cypress 2023h.) Jotta Slack-integraatio voitiin ottaa käyttöön, tuli olla kirjautuneena Cypress Cloud -palveluun (Cypress 2023d). Dokumentaatioihin tutustumisen perusteella voitiin todeta evaluointiehdon tulosten raporttien lähettämisestä kolmannen osapuolen palveluun täyttyneen.

Mahdollisuus ohjelmoida omia testejä

Kuten jo edellä todettiin, Cypress-työkalulla kaikki testit luotiin ohjelmoimalla. Tästä syystä voitiin Cypress-työkalulla luoda erilaisia testejä hyvin monipuolisesti. Näin ollen Cypress-työkalun voitiin todeta täyttäneen evaluaatioehdon mahdollisuudesta ohjelmoida omia testejä.

Mahdollisuus visuaaliseen vertailuun

Cypress-työkalussa ei ollut suoraan mahdollisuutta luoda testejä, joissa voitiin verrata näyttökuvia visuaalisesti keskenään. Työkaluun oli saatavilla Cypress-image-diff-lisäosa (plugin), jonka avulla erojen testaaminen näyttökuvien välillä onnistui (Elusoji 2022). Vaikka olisikin työkalun helppokäyttöisyyden kannalta ollut parempi, jos ominaisuus olisi löytynyt työkalusta sisäänrakennettuna, oli lisäosaa käyttämällä kuitenkin mahdollista toteuttaa vertailu näyttökuvien välillä. Cypress-työkalun voitiin todeta täyttäneen evaluaatioehdon mahdollisuudesta visuaaliseen vertailuun osittain.

Mahdollisuus CI-integraatioon/versionhallinnan integraatioon

Cypress-työkalu tarjosi tuen useille jatkuvan integraation (CI) palveluille. Näille kaikille integroitaville CI-palveluille, kuten esimerkiksi CircleCI:lle, GitHub Actionsille ja Bitbucketille, oli olemassa yksityiskohtainen dokumentaatio, jolla opastettiin palveluiden integraatio Cypress-työkalun kanssa. (Cypress 2023a.) Dokumentaation perusteella Cypress-työkalun voitiin todeta täyttäneen evaluaatiovaatimuksen integraatiomahdollisuudesta CI-palvelun/versionhallinnan palvelun kanssa.

Maksuttomuus

Cypress-työkalun käyttö ja testaaminen paikallisesti omalla koneella oli ilmaista, eikä sovelluksen lataaminen ja asennus vaatinut kustannuksia. Myös Cypress Cloudin käyttöönotto onnistui ilmaiseksi, mutta sen käyttöön löytyi myös maksullisia tilausmalleja. Cypress Cloud -palvelusta löytyi neljä erilaista tilausmallia, joista yksi oli ilmainen. Eri tilausmalleissa eroina olivat muun muassa käytettävissä olevien ominaisuuksien määrä, sallitut yhtäaikaiset käyttäjät sekä suoritettavien testien kuukausittainen lukumäärä. Maksulliset tilausmallit olivat Team, jonka hinta oli 75 dollaria kuukaudessa, Business, jonka hinta oli 300 dollaria kuukaudessa, ja Enterprise, jolle ei ollut kiinteää hintaa, vaan hinta neuvoteltiin käyttäjän kanssa tapauskohtaisesti. (Cypress 2023f.) Cypress Cloudin ilmainen tilausmalli ei todennäköisesti olisi riittävä muuten kuin pienen projektin tarpeisiin, ja isommassa projektissa olisikin todennäköisesti harkittava maksullisen tilausmallin käyttöä. Cypress Cloudin käyttö ei kuitenkaan ollut välttämätöntä, ja Cypress-työkalun käyttö oli mahdollista ilman sitä. Cypress-työkalu voitiin esimerkiksi integroida CI-palveluun, jonka kautta on mahdollista myös tehdä raportteja suoritetuista testeistä, vaikka Cypress Cloud ei olisikaan käytössä (Hiltunen 2023). Voitiin todeta, että Cypress-työkalu täytti evaluaatiovaatimuksen maksuttomuudesta.

5.2 Testim-työkalun evaluointi

Testim-työkalu arvioitiin työkalun valinnalle asetettujen evaluointivaatimusten perusteella. Arviointi tapahtui tutustumalla Testim-työkalun dokumentaatioon sekä hankkimalla pääsy työkaluun, jotta päästiin tekemään yksinkertaisia testejä ja siten kokeilemaan sitä käytännössä. Testim-työkalu evaluoitiin jokaisen vaatimuksen osalta ja tulokset kirjattiin taulukkoon. Evaluoinnin tulokset on esitetty taulukossa 2.

TAULUKKO 2. Testim-työkalun evaluoinnin tulokset

Evaluoinnin vaatimus	Evaluoinnin tulos
Vaaditut käyttöympäristöt	Kyllä
Helppokäyttöisyys	Kyllä
Käyttöönoton helppous	Kyllä
Testien nopea luonti	Osittain
Testien uudelleenkäytettävyys	Kyllä
Testien vakaus	Osittain
Ajastettujen testien mahdollisuus	Osittain
Testitulosten raportointi kolmannen osapuolen sovellukseen	Kyllä
Mahdollisuus ohjelmoida omia testejä	Osittain
Mahdollisuus visuaaliseen vertailuun	Osittain
Mahdollisuus CI-integraatioon/versionhallinnan integraatioon	Kyllä
Maksuttomuus	Osittain

Vaaditut käyttöympäristöt

Testim-työkalu sopi sekä selainpohjaisten verkkosovellusten että mobiilisovellusten testaamiseen (Testim 2022b; Testim 2023f). Verkkosovellusten osalta työkalu sisälsi tuen useilla eri verkkoselaimilla testaamiseen. Tuettuja selaimia olivat Chrome, Firefox, Safari ja Edge. (Testim 2023e.) Verkkosovellusten lisäksi Testim-työkalusta löytyi tuki natiivien Android- ja iOS-sovelluksien testaamiseen (Testim 2023a). Testim-työkalu tukikin projektimme kaikkia kolmea käyttöympäristöä, joten voitiin todeta, että se täytti evaluointivaatimusten vaadituista käyttöympäristöistä.

Helppokäyttöisyys

Testim-työkalulla voitiin luoda testejä joko nauhoittamalla (recording) käyttäjän toimia tai kirjoittamalla ne JavaScript-ohjelmointikielellä. Testien nauhoittaminen tapahtui Chrome-selaimessa ja tarvitsi erityisen laajennusosan, Testim Editorin. Testien luonti nauhoittamalla oli mahdollista sekä verkkosovellusten että mobiilisovellusten osalta. Testien nauhoitusta varten täytyi käyttäjän ensin kirjautua verkkoselaimen kautta Testim-tililleen. Sitten nauhoitus käynnistettiin ja suoritettiin verkkosovelluksessa tai mobiilisovelluksessa jokin käyttäjän suorittama toimi, kuten esimerkiksi sisäänkirjautuminen. Testim Editor -laajennusosa tallensi automaattisesti käyttäjän suorituksen jokaisen vaiheen (step). Nauhoituksen loputtua oli testi luotu, ja sitä voitiin tarkastella ja tarvittaessa muokata. Nauhoittaminen oli hyvin helppoa, eikä testien luonti sen avulla vaatinut lainkaan ohjelmointitaitoja. Ohjelmointitaitoisten oli mahdollista luoda testejä myös ohjelmoimalla testit kokonaan itse. Oli myös mahdollista luoda testi nauhoittamalla ja sitten siirtää testi koodimuodossa ohjelmointiympäristöön (esimerkiksi Visual Studio Codeen) ja muokata testiä halutun kaltaiseksi. (Testim 2022b.) Testien luominen Testim-työkalulla oli tehty monipuoliseksi ja käteväksi, eikä tämä välttämättä vaatinut ollenkaan ohjelmointitaitoja. Voitiinkin todeta, että Testim-työkalu täytti evaluaatioehdon työkalun helppokäyttöisyydestä.

Käyttöönoton helppous

Testim-työkalua käytettiin verkkoselaimen kautta, ja käyttöönottoa varten oli palveluun rekisteröidyttävä ja luotava tunnukset. Evaluointia varten ohjelmiston maksullisesta tilausmallista saatiin käyttöön 14 vuorokauden testijakso. Chrome-selaimeen asennettiin Testim Editor -laajennusosa verkkosovellusten testien nauhoituksia varten (Testim 2022b). Testien luontiin nauhoittamalla ei muuta vaadittu, ja testejä verkkosovelluksille kyettiin luomaan välittömästi (Testim 2022c). Verkkosovellusten testien luomiseksi ohjelmoimalla sekä nauhoitettujen testien muuttamiseksi koodimuotoon vaadittiin Testim Dev Kit -ohjelmistokehyksen asennus (Testim 2022b). Asennuksen tueksi löytyi selkeä dokumentaatio, jonka perusteella se onnistui vaivattomasti. Dokumentaation avulla myös ensimmäisten testien kirjoitus ja suoritus onnistui helposti. (Testim 2021d.) Verkkosovelluksien testejä voitiin suorittaa paitsi paikallisesti omalla tietokoneella myös Testim-työkalun omassa testauslaitekirjastossa, joka oli nimeltään Testim cloud grid. Testim-työkalu oli mahdollista integroida myös eri kolmannen osapuolen testauslaitekirjastoihin etätestausta varten. (Testim 2023b.)

Testien luonti mobiilialustoilla vaati erillisen Tricentis Mobile Agent -lisäosan asennuksen. Lisäosa vaadittiin, jotta työasemaan kytketyt fyysiset tai työasemalle asennetut simuloidut mobiililaitteet

voitiin yhdistää Testim-työkaluun. Tricentis Mobile Agentin asennus suoritettiin Testim-työkalun verkkoselainkäyttöliittymän kautta noudattamalla dokumentaation ohjeita. Lisäosan avulla voitiin testejä mobiilisovelluksille alkaa luomaan nauhoittamalla. (Testim 2023a.) Mobiilisovellusten testejä pystyi suorittamaan paikallisesti fyysisillä tai simuloiduilla mobiililaitteella. Testattava sovellus voitiin ladata (upload) Testim-työkalun käyttöliittymään, tai se voitiin valita suoraan testauslaitteesta. (Testim 2023d). Mobiililaitteiden etätestausta varten vaadittiin integraatio kolmannen osapuolen testauslaitekirjaston, HeadSpin-palvelun, kanssa (Testim 2023b). Android-laitteiden kohdalla oli käyttöönotto todella helppoa, eikä siihen vaadittu oikeastaan muuta kuin laitteen yhdistämistä tietokoneeseen. Käyttöönotto iOS-laitteilla oli työläämpää. Ennen kuin iOS-laitteilla pääsi luomaan testejä, tuli käytettävä laite linkittää projektin Apple-kehitystiimiin ja luoda testausta varten lupasertifikaatteja. (Testim 2023d.)

Vaikka Testim-työkalun käyttöön ottamiseksi tulikin asentaa kaksi lisäosaa, löytyi niiden asennuksen ja käyttöönoton tueksi hyvä dokumentaatio. Verkkosovellusten ja Android-sovellusten osalta käyttöönotto oli todella vaivatonta. IOS-sovellusten osalta jouduttiin tekemään enemmän asetusten säätöjä ja käyttöönotto oli hankalampaa, mutta tämä johtui enemmänkin Applen omista käytännöistä ja vaatimuksista kuin itse Testim-työkalusta. Testim-työkalun todettiin täyttäneen vaatimuksen käyttöönoton helppoudesta.

Testien nopea luonti

Testim-työkalulla kyettiin luomaan testejä verkkosovelluksille sekä mobiilisovelluksille nauhoittamalla käyttäjän toimia. Testien luonti verkkosovelluksille onnistui kokonaan itse ohjelmoimalla tai muuttamalla nauhoitettu testi koodimuotoon ja muokkaamalla sitä omiin tarpeisiin sopivaksi. Mobiilisovellusten testit voitiin luoda pelkästään nauhoittamalla. Mobiililaitteille ei ollut mahdollista ohjelmoida testejä itse eikä muuttaa nauhoitettuja testejä koodimuotoon.

Evaluointia varten luotiin yksinkertainen sisäänkirjautumisen testi asiakasprojektin verkkosovellukseen (liite 2), Android-sovellukseen (liite 3) sekä iOS-sovellukseen (liite 4) sekä kokeiltiin niiden suoritusta. Havainnollistamista varten verkkosovelluksen testi muutettiin koodimuotoon, mutta mobiilisovelluksien testit esitetään, kuten ne ovat Testim-työkalun käyttöliittymässä. Verkkosovelluksen ja Android-sovelluksen osalta testien luonti ja suoritus onnistui helposti. Näin ei kuitenkaan ollut iOS-sovelluksen kohdalla. Testi iOS-sovellukselle onnistuttiin kyllä helposti nauhoittamaan, mutta testin suoritusta ei kyetty onnistuneesti toteuttamaan. Syytä ongelmaan tiedusteltiin Testim-työkalun tekniseltä tuelta, ja ongelma johtui

ilmeisimmin virheestä Testim Mobile Agent -lisäosassa. Virheeseen ei kuitenkaan saatu evaluaation aikana korjausta, joten toimivaa testiä ei onnistuttu luomaan kuin verkkosovellukselle ja Android-sovellukselle. IOS-sovellusten osalta vaatii Testim-työkalu päivitystä tai korjausta. Näin ollen Testim-työkalu täytti vaatimuksen testien nopeasta luonnista vain osittain.

Testien uudelleenkäytettävyys

Testim-työkalulla testien uudelleenkäytettävyys oli helposti toteutettavissa. Nauhoittamalla luotujen testien vaiheet tai kokonaiset testit voitiin yhdistää ryhmiksi (group). Ryhmät taas voitiin määrittää jaetuiksi ryhmiksi (shared groups). Näitä jaettuja ryhmiä voitiin hyödyntää eri testeissä. (Testim 2022d.) Esimerkiksi 6-numeroisen PIN-koodin näppäilyn vaiheet voitiin yhdistää yhdeksi jaetuksi ryhmäksi, jota voitiin käyttää uudelleen joko samassa tai kokonaan eri testissä. Myös itse ohjelmoitujen testien uudelleenkäytettävyys oli toteutettu hyvin. Testeissä käytettävien elementtien paikkatiedot eli lokaattorit voitiin tallentaa erilliseen tiedostoon, mistä niitä voitiin kutsua eri testeihin. Täten esimerkiksi saman painikkeen painallus voitiin nopeasti ohjelmoida useaan testiin. Tämä oli mahdollista, koska viite kyseisen painikkeen lokaattoriin oli jo olemassa, joten siihen voitiin viitata eri testeistä. Testejä ei kuitenkaan voitu käyttää uudelleen eri alustojen välillä. Esimerkiksi Android-alustalle luotuja testejä tai ryhmiä ei voitu käyttää iOS- tai verkkoalustalla. Koska testien uudelleenkäytön mahdollisuus oli kuitenkin saman alustan sisällä laajasti mahdollista, voitiin todeta, että Testim-työkalu täytti evaluaatioehdon testien uudelleenkäytettävyydestä.

Testien vakaus

Testeissä tapahtuvissa toiminnoissa tarvitaan toiminnan kohteena olevalle elementille paikkaviite eli lokaattori. Lokaattorin avulla toiminta, kuten esimerkiksi painikkeen painallus, kohdistuu oikeaan käyttöliittymän elementtiin. Testim-työkalulla luoduissa testeissä hyödynnettiin eri elementtien tunnistamisessa niin kutsuttuja älylokaattoreita (Smart locators). Lokaattorien luonnissa hyödynnettiin tekoälyä, minkä ansiosta lokaattoreita ei määritetty pelkästään yksittäisen ominaisuuden, kuten esimerkiksi id:n, vaan jopa yli sadan eri ominaisuuden perusteella, mikä paransi elementin tunnistamista. Tekoälyä hyödynnettiin lokaattorien luonnin lisäksi myös niiden ylläpidossa. Mikäli yksittäinen ominaisuus mahdollisesti muuttuisi, pystyttiin tekoälyn avulla automaattisesti hyödyntämään elementin tunnistamiseen muita elementtiin viittaavia ominaisuuksia. Näin ollen lokaattori tunnisti yhä siihen viittaavan elementin, eikä testi epäonnistunut. Älylokaattoreita käytettiin kaikilla alustoilla nauhoittamalla luoduissa testeissä. Lisäksi ne toimivat verkkosovellusten testeissä, jotka nauhoitettiin ja muutettiin koodimuotoon.

(Testim 2022b.) Älylokaattorien lisäksi testien vakauden parantamiseen vaikutti testien hyvä uudelleenkäytettävyys. Mahdollisen virheen sattuessa virhe voitiin korjata jaetussa ryhmässä, jolloin se korjautui automaattisesti kaikissa ryhmää käyttävissä testeissä.

Testim-työkalun testit olivat teoriassa erittäin vakailta, ja käytännön kokeilun perusteella niissä havaittiin puutteita. Verkkosovelluksen ja Android-sovelluksen osalta testien luonti ja suoritus onnistuivat moitteetta, mutta iOS-sovelluksen kohdalla oli ongelmia, eikä onnistunutta testiä onnistuttu luomaan eikä suorittamaan. Kuten jo aiemmin evaluaatiovaatimuksen testien nopeasta luonnista kohdalla todettiin, johtui tämä virheestä Testim Mobile Agent -lisäosassa. Virheeseen ei evaluaatiojakson aikana saatu korjausta, eikä toimivia testejä iOS-sovellukselle kyetty luomaan eikä suorittamaan. Näin ollen ei iOS-testejä voitu myöskään kuvailla vakaiksi. Tämän takia jouduttiinkin toteamaan, että Testim-työkalu täytti evaluaatiovaatimuksen testien vakaudesta vain osittain.

Ajastettujen testien mahdollisuus

Testim-työkalulla testejä oli mahdollista ajastaa eri tavoilla. Verkkosovellusten testejä voitiin ajastaa tapahtumaan suoraan työkalun käyttöliittymän kautta. Ajastuksessa voitiin tarkasti määrittää muun muassa, mitkä testit tai testikokoelmat suoritetaan, minä viikonpäivinä, mihin aikaan, missä testiympäristössä sekä miten testien tulokset raportoidaan. (Testim 2022e.) Testim-työkalu tarjosi verkkosovelluksia varten oman Testim cloud grid -testauslaitekirjaston, jossa ajastettuja testejä voitiin suorittaa etänä erilaisilla laitteilla ja kokoonpanoilla. Testim-työkalu oli mahdollista integroida myös kolmannen osapuolen testauslaitekirjastoihin, joiden laitteilla testit voitiin ajastaa tapahtumaan. (Testim 2023b.) Testim-työkalusta löytyi tuki integraatiolle useisiin eri CI-palveluihin (Testim 2021b). Näin verkkosovellusten ajastaminen oli mahdollista myös CI-palvelun kautta. Mobiililaitteiden testien ajastamista ei voitu tehdä suoraan Testim-työkalun käyttöliittymän kautta. Mobiililaitteiden testien ajastus oli mahdollista ainoastaan CI-palvelun kautta. Lisäksi mobiililaitteiden testien ajastus vaati integroinnin kolmannen osapuolen palvelun, HeadSpinin, testauslaitteistokirjaston kanssa. (Testim 2023b.) Koska Testim-työkalu tuki testien ajastusmahdollisuutta laajasti verkkosovellusten osalta mutta mobiilisovellusten osalta vaadittiin integrointi kolmannen osapuolen palveluun, voitiin todeta Testim-työkalun täyttäneen evaluaatiovaatimuksen ajastettujen testien mahdollisuudesta vain osittain.

Testitulosten raportointi kolmannen osapuolen sovellukseen

Testim-työkalulla suoritettujen testien raportit oli mahdollista nähdä paitsi työkalun käyttöliittymästä, mutta ne oli myös mahdollista lähettää haluttuihin sähköpostiosoitteisiin. Työkalusta löytyi myös integraatiomahdollisuus Slack-pikaviestintäohjelmaan, joten raporttien automaattinen lähetys halutulle Slack-kanavalle oli mahdollista (Testim 2022a). Testim-työkalun voitiin siis todeta täyttäneen evaluaatiovaatimuksen testitulosten raportoinnista kolmannen osapuolen sovellukseen.

Mahdollisuus ohjelmoida omia testejä

Testien luonnit Testim-työkalulla tapahtuivat kaikilla tuetuilla alustoilla ensisijaisesti nauhoittamalla. Verkkosovellusten osalta testejä oli kuitenkin mahdollista luoda ohjelmoimalla ne kokonaan itse. Testien ohjelmointia varten tuli asentaa Testim Dev Kit -ohjelmistokehys, jonka jälkeen ohjelmointi tapahtui JavaScript-ohjelmointikielellä. Testejä oli mahdollista luoda myös siten, että ensin testi nauhoitettiin, jonka jälkeen se muutettiin koodimuotoon, ja tarvittaessa testiä voitiin muokata. Jotta Testim-työkalun omia älylokaattoreita pääsi hyödyntämään, tulikin noudattaa tätä niin kutsuttua hybridimallia. (Testim 2022b.) Työkalun käytännön kokeilun perusteella havaittiin, että omien testien ohjelmoiminen oli kuitenkin mahdollista vain verkkosovellusten kohdalla. Mobiilisovelluksien testejä voitiin luoda vain nauhoittamalla. Evaluaatiovaatimuksen mahdollisuudesta ohjelmoida omia testejä voitiin todeta Testim-työkalun kohdalla toteutuneen vain osittain.

Mahdollisuus visuaaliseen vertailuun

Testim-työkalulla luoduissa testeissä oli mahdollista vertailla visuaalisia eroja määritellyn perustilan ja testien välillä. Tämä toiminnallisuus oli kuitenkin kolmannen osapuolen, Applitoolin, tarjoama ja vaati toimiakseen integroinnin Applitool Eyes -sovelluksen kanssa. Tämä toiminnallisuus oli lisäksi käytettävissä vain Testim-työkalun kalleimman tilausmallin kanssa. (Testim 2022f.) Koska toiminnallisuus visuaaliselle vertailulle oli olemassa mutta vaati toimiakseen integraation kolmannen osapuolen palveluun, voitiin todeta evaluaatiovaatimuksen mahdollisuudesta visuaaliseen vertailusta toteutuneen vain osittain.

Mahdollisuus CI-integraatioon/versionhallinnan integraatioon

Testim-työkalu tuki integraatiota usean CI-palvelun kanssa. Esimerkkeinä tuetuista palveluista olivat muun muassa Circle CI, Codeship, GitlabCI ja Azure Devops build pipeline. CI-integraation tueksi löytyi kaikkien kyseisten palveluiden osalta hyvä dokumentaatio. (Testim 2021b.) Testim voitiin myös integroida versionhallinnan palvelun, kuten esimerkiksi Bitbucketin, kanssa (Testim 2021a). Integrointi oli mahdollista myös tehtävänhallintaohjelmiston, kuten esimerkiksi Jiran,

kanssa (Testim 2021c). Voitiinkin siis todeta Testim-työkalun täyttäneen evaluaatiovaatimuksen mahdollisuudesta CI-integraatioon/versionhallinnan integraatioon.

Maksuttomuus

Testim-työkalun maksullisesta tilausmallista saatiin kokeilemista ja testaamista varten ilmainen 14 vuorokauden testiversio. Testin jälkeen työkalun käyttöä oli mahdollista jatkaa käyttäen kolmea erilaista tilausmallia. Eroja eri tilausmallien välillä olivat muun muassa käytettävissä olevien toiminnallisuuksien määrä, sallitut yhtäaikaiset käyttäjät, suoritettavien testien määrät ja niin edelleen. Eri tilausmallit olivat ilmainen Community, vähintään 450 dollaria kuukaudessa maksava Essentials, ja vähintään 1 000 dollaria kuukaudessa maksava Pro. (Testim 2023c.) Testim-työkalusta voitiin ottaa käyttöön ilmainen Community-tilausmalli, mikä mahdollisti työkalun ilmaisen käytön tietyin rajauksin. Oli kuitenkin todettava, että ilmaisen tilausmallin rajatut ominaisuudet ja testauksien määrä eivät tulisi riittämään kuin hyvin pienen projektin tarpeisiin. Vähänkin suuremman projektin tulisi harkita maksullisen tilausmallin käyttöä. Näin ollen voitiinkin todeta, että Testim-työkalu täytti evaluaatiovaatimuksen maksuttomuudesta vain osittain.

5.3 Testsigma-työkalun evaluointi

Testsigma-työkalua arvioitiin työkalun valinnalle asetettujen evaluointivaatimusten perusteella. Arviointi tapahtui tutustumalla työkalun dokumentaatioon sekä testaamalla sitä käytännössä muun muassa luomalla yksinkertaisia testejä. Testsigma-työkalu evaluoitiin jokaisen vaatimuksen osalta ja tulokset kirjattiin taulukkoon. Evaluoinnin tulokset on esitetty taulukossa 3.

TAULUKKO 3. Testsigma-työkalun evaluoinnin tulokset

Evaluoinnin vaatimus	Evaluoinnin tulos
Vaaditut käyttöympäristöt	Kyllä
Helppokäyttöisyys	Kyllä
Käyttöönoton helppous	Kyllä
Testien nopea luonti	Kyllä
Testien uudelleenkäytettävyys	Kyllä
Testien vakaus	Kyllä
Ajastettujen testien mahdollisuus	Kyllä
Testitulosten raportointi kolmannen osapuolen sovellukseen	Osittain
Mahdollisuus luoda omia testejä	Kyllä
Mahdollisuus visuaaliseen vertailuun	Kyllä
Mahdollisuus CI-integraatioon/versionhallinnan integraatioon	Kyllä
Maksuttomuus	Osittain

Vaaditut käyttöympäristöt

Testsigma-työkalu soveltui sovellusten testaukseen useilla eri alustoilla (Testsigma 2023g). Verkkosovellusten testauksen (Testsigma 2023b) lisäksi se soveltui ohjelmointirajapintojen (API) testaukseen (Testsigma 2023c). Työkalulla kyettiin lisäksi testaamaan sekä natiiveja Android- ja iOS-mobiilisovelluksia että hybridimobiilisovelluksia (Testsigma 2023a). Testausta voitiin suorittaa paikallisesti omilla laitteilla tai hyödyntää pilvipalveluna tarjottavaa testauslaitekirjastoa. Testauslaitekirjasto tarjosi tuen yli 800 erilaiselle selainversiolle ja yli 1 000 eri laitteelle (Testsigma 2023g). Testsigma-työkalua voitiin käyttää joko kokonaan pilvipohjaisena palveluna verkkoselaimen kautta (Testsigma Cloud) tai paikallisen asennuksen vaativana versiona (Testsigma Community Edition) (Testsigma 2022k). Sekä Testsigma Cloud että Testsigma Community Edition tukivat testien suorittamista kaikilla vaadituilla käyttöympäristöillä. Testsigma-työkalun todettiin täyttäneen evaluaatiovaatimuksen vaadituista käyttöympäristöistä.

Helppokäyttöisyys

Testsigma-työkalulla testit luotiin kirjoittamalla yksinkertaisia englanninkielisiä komentoja. Näitä komentoja kutsuttiin toiminnoiksi (action). Yhtenä esimerkkinä toiminnosta oli klikkaa elementtiä (Click on element), jossa elementiksi asetettiin haluttu UI-elementti, kuten painike. Testejä oli myös mahdollista toteuttaa nauhoittamalla (recording) eli suorittamalla käyttäjänä haluttu tapahtuma, jolloin Testsigma-työkalu loi automaattisesti käyttäjän toimia vastaavat toiminnot. (Testsigma

2023g.) Nauhoitettuja testejä oli mahdollista tarkastella ja muokata jälkeenpäin halutun kaltaiseksi. Nauhoittaminen onnistui Testsigma Cloudissa kaikilla alustoilla, mutta Testsigma Community Editionissa tuki nauhoittamiselle oli vain verkkosovellusten osalta (Testsigma 2023f). Testsigma Communty Editionissa testien luonti mobiilisovelluksille oli kuitenkin mahdollista manuaalisesti kirjoittamalla. Testsigma-työkalu sisälsi laajan kirjaston valmiita toimintoja, joita testien luontiin tarvitaan, mutta toimintoja oli mahdollista myös ohjelmoida itse käyttämällä Java-ohjelmointikieltä (Testsigma 2022a). Testsigma-työkalulla kyettiinkin luomaan testejä monipuolisesti, eikä ohjelmointitaitoa välttämättä vaadittu. Todettiin, että Testsigma-työkalu täytti evaluaatiovaatimuksen helppokäyttöisyydestä.

Käyttöönoton helppous

Testsigma-työkalu voitiin ottaa käyttöön kahdella eri tavalla. Ensimmäinen tapa oli maksullinen, pilvipohjainen, verkkoselaimen kautta käytettävä palvelu Testsigma Cloud. Toinen oli ilmainen Testsigma Community Edition, joka vaati asennuksen paikalliselle tietokoneelle tai palvelimelle. (Testsigma 2022k.) Testsigma Cloudista saatiin käyttöön 30 päivän ilmainen testijakso, jonka aikana palvelua oli mahdollista laajasti koekäyttää. Käyttöönottoa varten palveluun rekisteröidyttiin ja luotiin tunnukset, jonka jälkeen palveluun voitiin kirjautua verkkoselaimen kautta.

Sisäänkirjautumisen jälkeen testien luominen manuaalisesti kirjoittamalla onnistui välittömästi. Testien nauhoittamista varten täytyi kuitenkin Chrome-selaimeen asentaa Testsigma Recorder -laajennusosa, jonka jälkeen testien luonti myös nauhoittamalla onnistui (Testsigma 2022d). Luotuja testejä pystyttiin välittömästi suorittamaan Testsigma Labissa, joka oli Testsigma-työkalun oma, pilvipalveluna toteutettu, laaja testauslaitteistokirjasto (Testsigma 2023d). Jotta testejä pystyttiin suorittamaan paikallisesti omalla tietokoneella tai paikallisilla mobiililaitteilla, täytyi ensin asentaa Testsigma Agent -lisäosa. Asennuksen jälkeen testaukseen käytettävät paikalliset laitteet pystyttiin rekisteröimään työkalussa. (Testsigma 2022l.) Mobiililaitteilla testausta varten täytyi testattava sovellus ladata (upload) työkaluun sen käyttöliittymän kautta, jonka jälkeen laitteella voitiin alkaa nauhoittamaan ja suorittamaan testejä (Testsigma 2022b; Testsigma 2022c). Android-laitteiden osalta käyttöönotto onnistui vaivattomasti, mutta iOS-laitteita varten täytyi tehdä lisätoimia. Jotta testausta paikallisilla iOS-laitteilla päästiin tekemään, tuli testauksessa käytettävä laite linkittää projektin Apple-kehitystiimiin ja luoda lupasertifikaatteja. (Testsigma 2022e.) Molempien mobiilialustojen sovelluksia pystyttiin kuitenkin testaamaan myös Testsigma Labissa olevilla laitteilla ilman mitään lisäasetusten määritystä.

Testsigma Community Editionin käyttämiseksi vaadittiin hieman enemmän toimia (Testsigma 2022c). Dokumentaatiota apuna käyttäen Testsigma-työkalun palvelin ja MySQL-tietokanta ladattiin ja asennettiin paikalliselle tietokoneelle, jossa ne pystytettiin toimimaan Docker-konteissa. Tämän jälkeen Testsigma-työkalu voitiin käynnistää paikallisesti tietokoneen selaimessa, jonka jälkeen luotiin käyttäjätunnukset ja kirjauduttiin palveluun. (Testsigma 2022n.) Testsigma Community Editionia käyttäessä voitiin paikallinen palvelu yhdistää pilvipalveluun (Testsigma Community Cloud), joka oli ominaisuuksiltaan kuitenkin maksullista, Testsigma Cloudin tarjoamaa palvelua suppeampi. Esimerkiksi testien suorittaminen Testsigma Labissa onnistui molemmissa palveluissa samalla tavalla, mutta testaukseen käytettäviä laitteita oli Testsigma Community Cloudissa vähemmän. (Testsigma 2022m.) Muutoin Testsigma Community Editionin käyttö ei juurikaan eronnut Testsigma Cloudin käytöstä, vaan testejä luotiin ja suoritettiin lähes samalla tavalla. Poikkeuksena olivat mobiililaitteiden testit, joita ei voitu Testsigma Community Editionissa luoda nauhoittamalla. Tarvittavat lisäosat, Testsigma Agent ja Testsigma Recorder, vaadittiin molempia palveluita käytettäessä.

Kaiken kaikkiaan Testsigma-työkalun käyttöönotto onnistui vaivattomasti ja saatavilla oli hyvä dokumentaatio. Testsigma Cloudin käyttöönottoon ei yksinkertaisimmillaan tarvittu muuta kuin tunnusten luonti ja kirjautuminen palveluun. Testsigma Community Editionin käyttöönotto onnistui dokumentaation avulla vaivattomasti, vaikka siihen tarvittiinkin asennus paikalliselle koneelle. Tarvittavien lisäosien asennus oli helppoa, ja niiden tueksi löytyi hyvä dokumentaatio. Testsigma-työkalun voitiin todeta toteuttaneen evaluaatiovaatimuksen käyttöönoton helppoudesta.

Testien nopea luonti

Testsigma-työkalulla testejä luotiin joko kirjoittamalla yksinkertaisia englanninkielisiä komentoja tai nauhoittamalla käyttäjän toimia. Testien luomiseksi ei tarvittu ohjelmointitaitoja. (Testsigma 2023g.) Testien luonti nauhoittamalla oli Testsigma Cloudissa mahdollista verkkosovellusten (Testsigma 2022d), Android-sovellusten (Testsigma 2022b) ja iOS-sovellusten (Testsigma 2022c) osalta. Käyttökokeilun perusteella kuitenkin huomattiin, että Testsigma Community Edition ei tukenut testien automaatista nauhoitusta mobiilisovellusten osalta vaan testit tuli luoda manuaalisesti kirjoittaen. Evaluointia varten luotiin yksinkertainen sisäänkirjautumisen testi asiakasprojektin verkkosovellukselle (liite 5), Android-sovellukselle (liite 6) ja iOS-sovellukselle (liite 7) sekä testattiin niiden suoritusta. Testien luonti kaikille kolmelle alustalle onnistui helposti, ja kullakin alustalla testi myös suoritettiin onnistuneesti. Testien luonnin tueksi löytyi verkkosovelluksille (Testsigma 2022d), Android-sovelluksille (Testsigma 2022b) sekä iOS-sovelluksille (Testsigma 2022c) hyvä

dokumentaatio. Testsigma-työkalun voitiin todeta täyttäneen evaluaatiovaatimuksen testien nopeasta luonnista.

Testien uudelleenkäytettävyys

Testsigma-työkalulla testien uudelleenkäytettävyys oli helposti toteutettavissa. Nauhoitettujen testien vaiheet voitiin yhdistää niin kutsutuiksi askelryhmiksi (step group), joita voitiin käyttää uudelleen eri testeissä (Testsigma 2023g). Esimerkiksi verkkosovelluksen sisäänkirjautumisen testi (liite 5) voitiin muodostaa omaksi askelryhmäksi, jota voitiin käyttää uudelleen eri testeissä. Askelryhmät olivat uudelleen käytettävissä ainoastaan saman alustan sisällä. Esimerkiksi Android-alustalla luodut ryhmät eivät olleet käytettävissä iOS-alustalla, eikä verkkosovelluksien ryhmiä voitu käyttää mobiilialustojen testeissä. Koska Testsigma-työkalulla oli kuitenkin mahdollista luoda uudelleen käytettäviä askelryhmiä saman alustan sisällä, voitiin todeta Testsigma-työkalun täyttäneen evaluaatiovaatimuksen testien uudelleenkäytettävyydestä.

Testien vakaus

Testeissä tapahtuvat toiminnot, kuten esimerkiksi painikkeen painallus, kohdistuvat aina tiettyyn käyttöliittymän elementtiin. Toiminnan kohteena olevalle elementille tarvitaan paikkaviite eli lokaattori varmistamaan, että toiminta kohdistuu oikeaan elementtiin. Testsigma-työkalulla luotujen testien lokaattorien ominaisuuksina käytettiin muun muassa elementin id:tä, CSS-luokkaa (class) tai XPathia (XML Path Language). Luotaessa testi nauhoittamalla muodostuivat lokaattorit automaattisesti tarkimman elementtiin viittaavan ominaisuuden mukaan. Lokaattoreita oli lisäksi myös mahdollista määrittää manuaalisesti. (Testsigma 2022i.) Lokaattorien luonti automaattisesti oli Testsigma Cloudilla mahdollista kaikkien alustojen kohdalla. Testsigma Community Editionissa ei ollut tukea mobiilisovellusten luomiseen automaattisesti nauhoittamalla, vaan testit oli kirjoitettava manuaalisesti. Täten myös elementtien lokaattorit oli määritettävä manuaalisesti. Nauhoitustyökalua pystyi kuitenkin tässäkin tapauksessa hyödyntämään lokaattorien määritykseen. Nauhoitustyökalun avulla voitiin katsoa, mitä elementtien ominaisuuksia työkalu ehdotti lokaattoreiksi, ja näitä ehdotuksia voitiin sitten hyödyntää testien kirjoituksessa. (Testsigma 2023f.)

Testsigma-työkalussa hyödynnettiin tekoälyä lokaattorien eheyden ja siten testien vakauden ylläpitämiseksi. Tämä niin kutsuttu itsekorjaus (Self-Healing) perustui tekoälyn havaitsemiin muutoksiin ohjelmiston koodissa, joita se pyrki automaattisesti korjaamaan. Esimerkiksi elementin ominaisuuden, kuten id:n, muuttuessa, pyrki tekoäly automaattisesti tekemään vastaavan

muutoksen myös kyseisen elementin lokaattoriin, jolloin testit edelleen onnistuisivat. (Testsigma 2023e.)

Testien vakauteen vaikutti myös mahdollisuus uudelleen käytettävien askelryhmien (step group) käyttöön. Koska samoja askelryhmiä pystyi käyttämään useissa testeissä, voitiin myös mahdollisten virheiden korjaus toteuttaa keskitetysti. Virheen sattuessa riitti, että se korjattiin askelryhmässä, jolloin se korjautui kaikissa sitä käyttävissä testeissä. Edellä mainittujen havaintojen perusteella voitiinkin todeta, että Testsigma-työkalu täytti evaluaatiovaatimuksen testien vakaudesta.

Ajastettujen testien mahdollisuus

Testsigma-työkalulla oli mahdollista ajastaa testit tapahtumaan automaattisesti haluttuna ajankohtana. Testien ajastusta varten luotiin testaussuunnitelma (test plan), jossa määritettiin, mikä tai mitkä testisarjat (test suite) ovat mukana suunnitelmassa. Ajastusta varten määritettiin myös testaukseen käytettävät laitteet ja laitekokoonpanot, kuten esimerkiksi se, mitä käyttöjärjestelmää, selainta ja selainversiota käytetään. Suoritettavaksi voitiin määrittää testisarjan kaikki testit tai vain tietyt, yksittäiset testit (partial test run). Ajastukselle määritettiin haluttu päivänmäärä, kellonaika ja se, kuinka useasti testien ajo toistettiin. (Testsigma 2022j.) Testien ajastus oli mahdollista verkkosovelluksille, Android-sovelluksille sekä iOS-sovelluksille. Testit suoritettiin hyödyntäen Testsigma-työkalun omaa testauslaitteistokirjastoa, Testsigma Labia. Testsigma-työkalu oli myös mahdollista integroida kolmannen osapuolen testauslaitekirjastojen, kuten esimerkiksi BrowserStackin tai Sauce Labsin, kanssa. (Testsigma 2023d.) Käyttökokemuksen perusteella havaittiin, että testien ajastus oli mahdollista sekä maksullisella Testsigma Cloudilla että ilmaisella Testsigma Community Editionilla. Lisäksi havaittiin, että testitulosten raportointi esimerkiksi sähköpostiin tai kolmannen osapuolen palveluun onnistui kuitenkin vain Testsigma Cloudin kautta, eikä Testsigma Community Editionista löytynyt tukea tälle ominaisuudelle. Testien ajastaminen oli Testsigma-työkalulla mahdollista myös CI-palvelun kautta, sillä Testsigmasta löytyi tuki integraatioon usean eri CI-palvelun kanssa (Testsigma 2022h). Testsigma-työkalun todettiin täyttäneen vaatimuksen ajastettujen testien mahdollisuudesta.

Testitulosten raportointi kolmannen osapuolen sovellukseen

Käytännön kokeilun perusteella havaittiin, että suoritettujen testien raportit olivat helposti nähtävissä Testsigma Cloudin sekä Testsigma Community Editionin käyttöliittymästä. Maksullisen Testsigma Cloudin osalta oli raportit mahdollista lähettää automaattisesti esimerkiksi sähköpostiin.

Testsigma Cloud oli lisäksi mahdollista integroida esimerkiksi Slack-pikaviestintäohjelman kanssa, jolloin raporttien lähetys automaattisesti jollekin Slack-kanavalle oli mahdollista (Testsigma 2022g).

Käytännön kokeilun perusteella havaittiin, ettei raporttien automaattista lähetysmahdollisuutta ollut Testsigma Community Editionissa ja ettei integraatio Slackiin ollut mahdollista. Todettiin, että tulosten raportoinnin tuen löydyttyä vain Testsigma-työkalun maksullisesta versiosta toteutui evaluaatiovaatimus testitulosten raportoinnista kolmannen osapuolen työkaluun vain osittain.

Mahdollisuus ohjelmoida omia testejä

Testsigma-työkalu sisälsi laajan kirjaston valmiita toimintoja (action), joilla voitiin kuvata käyttäjien toimia testejä kirjoitettaessa. Tämän lisäksi Testsigma-työkalussa oli mahdollisuus luoda omia toimintoja erityisten ohjelmalaajennuksien (Add-ons) avulla. Uusi ohjelmalaajennus luotiin Testsigma-työkalun käyttöliittymästä. Luotua ohjelmalaajennusprojektia pystyttiin muokkaamaan itse haluamallaan ohjelmointiympäristöllä. Ohjelmointi tapahtui Java-ohjelmointikielellä. Uusia toimintoja pystyttiin ohjelmoimaan paitsi verkkosovelluksille myös Android- ja iOS-sovelluksille. Muutoksien tai lisäyksien jälkeen ohjelmanlaajennus ladattiin Testsigma-työkalun käyttöliittymään, jolloin uusia toimintoja voitiin käyttää testien kirjoituksessa. Ohjelmanlaajennuksien luomiseen ja toimintojen ohjelmoinnin aloittamiseen oli olemassa hyvä dokumentaatio esimerkkeineen. (Testsigma 2022a.) Käyttökokeilun perusteella todettiin, että ohjelmalaajennuksien luonti onnistui sekä Testsigma Cloudissa että Testsigma Community Editionissa. Todettiin, että Testsigma-työkalu täytti evaluaatiovaatimuksen mahdollisuudesta ohjelmoida omia testejä.

Mahdollisuus visuaaliseen vertailuun

Testsigma-työkalulla luoduissa testeissä oli mahdollista vertailla testien eri vaiheissa visuaalisia eroja määritellyn perustilan ja testien väillä. Ominaisuus visuaaliseen vertailuun löytyi sisäänrakennettuna Testsigma-työkalusta, eikä käyttöönottoon esimerkiksi tarvittu mitään kolmannen osapuolen palvelua. (Testsigma 2022o.) Käyttökokemuksen perusteella havaittiin, että mahdollisuus visuaaliseen vertailuun löytyi sekä maksullisesta Testsigma Cloudista, että ilmaisesta Testsigma Community Editionista. Testsigma-työkalun todettiin täyttäneen evaluaatiovaatimuksen mahdollisuudesta visuaaliseen vertailuun.

Mahdollisuus CI-integraatioon/versionhallinnan integraatioon

Testsigma-työkalusta löytyi mahdollisuus integraatioon usean CI-palvelun kanssa. Tuettuja palveluita olivat muun muassa Circle CI, Codeship, Bamboo, Azure DevOps, CodeShip CI ja Jenkins. Kaikkien tuettujen palveluiden osalta löytyi integraation tueksi dokumentaatio (Testsigma 2022h). Testsigma-työkalu kyettiin integroimaan myös versionhallinnan, kuten esimerkiksi Bitbucketin, kanssa (Testsigma 2022f). CI-palvelujen ja versionhallinnan lisäksi löytyi Testsigma-työkalusta tuki eri tehtävienhallintaohjelmien, kuten muun muassa Jiran, kanssa (Testsigma 2022h). Käyttökokeilun perusteella todettiin, että integraatiotuet eri työkaluille löytyivät Testsigma Cloudin lisäksi myös Testsigma Community Editionista. Voitiinkin siis todeta Testsigma-työkalun täyttäneen evaluaatioehdon mahdollisuudesta CI-integraatioon/versionhallinnan integraatioon.

Maksuttomuus

Testsigma-työkalusta voitiin ottaa käyttöön ilmainen Testsigma Community Edition, tai maksullinen Testsigma Cloud. Testsigma Cloudille oli olemassa kaksi tilausmallia, Pro sekä Enterprise, jotka erosivat paitsi hinnaltaan myös muun muassa sisältämiensä ominaisuuksien perusteella. Ilmainen Testsigma Community Edition sisälsi vähemmän käytettäviä ominaisuuksia kuin Testsigma Cloudin Pro- tai Enterprise-tilausmallit. Tilausmalli Pron hinta oli vähintään 349 dollaria kuukaudessa, kun taas Enterprise-tilausmallin hinnasta oli sovittava tapauskohtaisesti, ja riippui muun muassa käyttöön halutuista ominaisuuksista. Evaluointia varten Testsigma-työkalusta otettiin käyttöön ilmainen Testsigma Community Edition, ja Testsigma Cloudin Enterprise-tilausmallista saatiin käyttöön 30 päivän mittainen, ilmainen testijakso. (Testsigma 2023f.)

Vaikka Testsigma-työkalusta olikin olemassa ilmainen tilausmalli, oli kuitenkin huomioitava, etteivät esimerkiksi ilmaisen tilausmallin testausminuutit (200 min/kk) tulisi todennäköisesti riittämään kovin laajaan käyttöön. Testien luonti mobiilisovelluksille oli maksullisessa Testsigma Cloudissa nopeampaa, sillä ilmaisessa Testsigma Community Editionissa ei mobiilisovellusten testien luominen nauhoittamalla ollut mahdollista. Täten käytännössä vähänkin isomman projektin tuleekin todennäköisesti ottaa käyttöön maksullinen tilausmalli. Voitiin todeta, että Testsigma-työkalu täytti evaluaatiovaatimuksen maksuttomuudesta vain osittain.

5.4 Työkalun valinta evaluoinnin perusteella

Kaikki kolme työkalua evaluoitiin kaikkien evaluaatiovaatimuksen osalta. Evaluaatiovaatimukset olivat: vaaditut käyttöympäristöt (1), helppokäyttöisyys (2), käyttöönoton helppous (3), testien nopea luonti (4), testien uudelleenkäytettävyys (5), testien vakaus (6), ajastettujen testien mahdollisuus (7), testitulosten raportointi kolmannen osapuolen työkaluun (8), mahdollisuus ohjelmoida omia testejä (9), mahdollisuus visuaaliseen vertailuun (10), mahdollisuus CI-integraatioon/versionhallinnan integraatioon (11) ja maksuttomuus (12). Evaluaation tulokset jokaisen työkalun osalta on nähtävissä taulukossa 4.

TAULUKKO 4. Työkalujen evaluointien tulokset

Evaluaatiovaatimus	Cypress	Testim	Testsigma
1	Ei	Kyllä	Kyllä
2	Osittain	Kyllä	Kyllä
3	Kyllä	Kyllä	Kyllä
4	Kyllä	Osittain	Kyllä
5	Kyllä	Kyllä	Kyllä
6	Osittain	Osittain	Kyllä
7	Kyllä	Osittain	Kyllä
8	Kyllä	Kyllä	Osittain
9	Kyllä	Osittain	Kyllä
10	Osittain	Osittain	Kyllä
11	Kyllä	Kyllä	Kyllä
12	Kyllä	Osittain	Osittain

Taulukosta 4 havaitaan, miten eri työkalut suoriutuivat kunkin evaluaatiovaatimuksen kohdalla. Tuloksista voitiin havaita, että kaikki työkalut täyttivät suurimmilta osin lähes kaikki evaluaation vaatimukset, ainakin osittain. Kaikista työkaluista jäi myös käytännön kokeilun perusteella suurimmilta osin hyvä kuva. Jokaisessa työkalussa oli omat hyvät puolensa, ja jokaisesta löytyi jotakin parannettavaa. Työn tarkoituksena oli kuitenkin evaluoida ja löytää työkaluista se parhaiten sopiva, joka mahdollisesti voisi tulla käyttöön asiakasprojektimme testaamiseen.

Evaluaatioiden tulokset voitiin pisteyttää siten, että Kyllä saa yhden pisteen, Osittain saa puoli pistettä ja Ei saa nolla pistettä. Maksimipistemäärä oli näin ollen 12 pistettä. Cypress sai kokonaispistemääräkseen 9,5 pistettä, Testim sai 9 pistettä, ja Testsigma sai 11 pistettä. Valintaa ei ollut järkevää kuitenkaan tehdä pelkästään tällaisen pisteytyksen perusteella, vaan eri evaluaatiokriteereitä tuli tarkastella yksityiskohtaisemmin.

Cypress-työkalu

Cypress-työkalu osoittautui helposti käyttöön otettavaksi työkaluksi, jolla oli helppo luoda testejä JavaScript-ohjelmointikielellä. Vaikka testien luontiin oli olemassa aputyökaluja, joilla testien vaiheita voitiin luoda nauhoittamalla (recording), olivat ne vielä kokeiluasteella tai kolmannen osapuolen tarjoamia. Testien luonti Cypress-työkalulla tapahtui käytännössä ohjelmoimalla ja vaati ohjelmointitaitoja. Työkalulla pystyttiin luomaan laajasti omia testitapauksia, ja testien uudelleenkäytettävyys toteutui hyvin. Työkalulla luotujen testien vakaus todettiin melko hyväksi, sillä vaikka testien vaiheissa viitattiinkin eri käyttöliittymän elementteihin vain yksittäisillä ominaisuuksilla, onnistui mahdollisten virheiden korjaus kuitenkin testien hyvän uudelleenkäytettävyyden ansiosta helposti. Cypress-työkalun integrointi onnistui muun muassa CI-palveluiden kanssa, ja CI-palvelun avulla myös testien ajastaminen oli mahdollista. Työkalun integrointi oli lisäksi mahdollista Slack-pikaviestintäohjelman kanssa, jonka kautta testiraportit voitiin julkaista. Cypress-työkalulla oli myös mahdollista tehdä visuaalista testausta, vaikkakin tämä ominaisuus vaati kolmannen osapuolen tekemän lisäosan käyttöä. Cypress-työkalu oli mahdollista ottaa käyttöön kokonaan ilman kustannuksia, joten se täytti myös evaluaatiovaatimuksen maksuttomuudesta.

Cypress-työkalun valinnan suurimmaksi esteeksi osoittautui kuitenkin se, ettei työkalu tukenut halutuista käyttöympäristöistä kuin verkkosovelluksia, eikä se soveltunut Android-, eikä iOS-sovellusten testaukseen. Tämän takia Cypress-työkalun osalta päädyttiin tulokseen, että se ei tule valituksi työkaluksi, sillä asiakasprojektimme vaatii, että työkalulla tulee pystyä testaamaan kaikkia kolmea alustaa.

Testim-työkalu

Testim-työkalu vaikutti lähtökohtaisesti lupaavalta työkalulta, sillä se tarjosi tuen kaikkien kolmen vaaditun alustan testaukseen. Työkalun käyttöönotto oli nopeaa ja vaivatonta, ja testien luonti nauhoittamalla käyttäjän toimia oli helppoa eikä välttämättä vaatinut ollenkaan ohjelmointitaitoja. Testejä verkkosovelluksille pystyi kuitenkin luomaan myös ohjelmoimalla. Testit tai testien osat

voitiin luoda laajasti uudelleenkäytettäviksi, milloin mahdollisten virheidenkin korjaus oli helppoa. Testim-työkalun älylokaattorit, sekä tekoälyn hyödyntäminen lokaattorien ylläpidossa lisäsi ainakin teoriassa testien kestävyyttä. Testim-työkalu pystyttiin integroimaan useisiin CI- ja versionhallinnan palveluihin sekä myös esimerkiksi Slack-pikaviestintäohjelmaan raporttien lähettämistä varten. Testien ajastaminen oli mahdollista, mutta ilman kolmannen osapuolen testauslaitekirjastoa ajastus ei onnistunut kuin verkkosovellusten osalta. Testim-työkalulla oli mahdollista tehdä visuaalista vertailua, mutta ominaisuus vaati integraation kolmannen osapuolen palveluun. Vaikka Testim-työkalua voitiinkin teoriassa käyttää ilman kustannuksia, vaatii vähänkin suuremman projektin testaaminen todennäköisemmin maksullisen tilausmallin hankintaa.

Testim-työkalun valintaan eniten vaikuttavaksi seikaksi osoittautui kuitenkin se, että vaikka testien luonti onnistui helposti sekä verkkosovellukselle, että Android-sovellukselle, ei iOS-sovelluksen osalta onnistuttu luomaan toimivaa testiä. Tämä johtui viasta Testim Mobile Agent -lisäosassa, eikä vikaa saatu korjattua evaluaatiojakson aikana. Se, ettei iOS-sovelluksen testausta saatu toimimaan, ja se, että monet ominaisuudet, kuten esimerkiksi visuaalinen testaus tai mobiililaitteiden etätestaus vaativat toimiakseen jonkin kolmannen osapuolen integraation, vaikutti lopulta siihen, ettei Testim tullut valituksi työkaluksi.

Testsigma-työkalu

Testsigma-työkalu täytti arvioidusta työkaluista valitut vaatimukset parhaiten. Työkalu soveltui kaikkien kolmen testausalustan testaukseen, ja sen käyttöönotto oli helppoa. Testien luonti työkalulla tapahtui helposti. Testien vaiheet voitiin luoda joko englanniksi kirjoittamalla tai nauhoittamalla käyttäjä toimia, jolloin ne muodostuivat automaattisesti. Ohjelmointitaitoa ei testien luomiseen tarvittu. Testejä ja testien vaiheita pystyttiin uudelleenkäyttämään helposti useissa eri testeissä. Testisigma-työkalu hyödynsi tekoälyä ylläpitämään ja korjaamaan mahdollisia virheitä testien lokaattoreissa, ja mahdollisten virheiden korjaus voitiin tarvittaessa tehdä helpoksi testien hyvän uudelleenkäytettävyyden myötä. Testien ajastus onnistui kätevästi kaikilla testausalustoilla, sillä testejä pystyttiin suorittamaan hyödyntäen työkalun omaa testauslaitekirjastoa. Vaikka testien luomiseen ei tarvittu ohjelmointitaitoja, oli Testsigma-työkalulla kuitenkin mahdollista luoda yksilöityjä toimintoja myös ohjelmoimalla. Työkalusta löytyi tuki myös visuaaliseen testaukseen. Testsigma-työkalu kyettiin integroimaan usean eri CI-palvelun tai versionhallinnan kanssa. Vaikka työkalua oli mahdollista käyttää ilman kustannuksia, oli kuitenkin huomioitava, että ilmaisversion suppeamman sisällön vuoksi joudutaan projektin käyttöön todennäköisimmin valitsemaan maksullinen tilausmalli. Lisäksi eräät ominaisuudet, kuten esimerkiksi mahdollisuus

testausraporttien lähetykseen Slack-pikaviestintäohjelmaan, olivat käytössä vain maksullisessa tilausmallissa.

Testsigma-työkalu täytti evaluoiduista työkaluista parhaiten eri evaluaatiovaatimukset ja näyttää täten soveltuvan hyvin testaustyökaluksi asiakasprojektin käyttöön. Testsigma-työkalu päädyttiin evaluaation perusteella valitsemaan lupaavaksi vaihtoehdoksi asiakasprojektin testausautomaatiotyökaluksi.

6 YHTEENVETO JA POHDINTA

Opinnäytetyön tarkoituksena oli evaluoida erilaisia testausautomaatioon tarkoitettuja työkaluja spesifisesti määritettyjen vaatimusten perusteella. Tätä evaluaation tulosta tullaan käyttämään pohjana testausautomaatiotyökalun valinnalle Younite-AI-yrityksen asiakasaprojektin käyttöön. Valittu työkalu saattaa tulevaisuudessa olla potentiaalinen vaihtoehto asiakasprojektin ohjelmistotuotteiden testaukseen. Evaluaatioon perusteella tultiin lopputulokseen, että yksi kolmesta evaluoiduista työkalusta, Testsigma, onnistui selvästi parhaiten täyttämään työkalulle asetetut vaatimukset. Näin ollen Testsigma-työkalua voitiin evaluaation perusteella suositella potentiaaliseksi työkaluksi asiakasprojektin ohjelmistotuotteiden testaukseen.

En ollut aiemmin päässyt kovin syvällisesti tutustumaan ohjelmistotestaukseen enkä testausautomaatioon, joten evaluoinnin lisäksi opinnäytetyön tarkoituksena oli oppia ja syventää omaa ymmärrystäni kyseisistä aiheista. Ohjelmistotestaus on valtavan tärkeä osa ohjelmiston kehitystä, sillä sen avulla muun muassa varmistetaan, että sovellus toimii halutulla tavalla eikä siinä ole virheitä. Testauksen ansiosta sovelluksen kehitykseen kuluva aika ja kustannukset vähenevät, sekä sovelluksen laatu paranee. (Atlassian 2023.) Myös sovelluksen tietoturva sekä käyttäjätyytyväisyys paranevat (Hamilton 2023b). Oli hyvä, että pääsin tämän opinnäytetyön ansiosta tutustumaan ohjelmistotestaukseen tarkemmin. Uskon, että olen oppinut opinnäytetyön teon ansiosta aiheesta paljon. Aihe osoittautui kuitenkin valtavan laajaksi, joten pääsin opinnäytetyössäni tutustumaan vasta perusteisiin. Minun täytyy vielä tulevaisuudessa jatkaa perehtymistä ja oppini syventämistä aiheesta.

Opinnäytetyössäni pääsin myös perehtymään testausautomaatioon ja eri työkaluihin, mikä oli minulle hyvin opettavaista. Minulle tuli suurena yllätyksenä eri automaatiotyökalujen suuri määrä. Eräs haaste oli päättää, mitkä kolme työkalua päädyttiin valitsemaan varsinaiseen evaluaatioprosessiin. Itse evaluointiprosessi onnistui hyvin kaikkien työkalujen osalta. Kaikkia kolmea työkalua päästiin testaamaan käytännössä sekä kaikista löytyi kattava dokumentaatio, joiden avulla onnistuttiin selvittämään, miten työkalut suoriutuivat kunkin evaluaatiovaatimuksen kohdalla.

Vaikka evaluaation perusteella päädyttiin suosittelemaan Testsigma-työkalua, ei tätä kuitenkaan pelkästään tämän evaluaation perusteella voida valita asiakasprojektin käyttöön. Työkalua tulee

vielä jatkossa testata tarkemmin ennen lopullista valintaa. Työkalulla tulee luoda useampia todellisiin käyttötapauksiin perustuvia testejä ja testata, miten työkalu suoriutuu näistä haastavimmista testeistä. Testejä tulee myös suorittaa tiheään ja useita, jolloin saadaan tarkempaa selvyyttä, kuinka kestäviä (robust) testit ovat.

Mikäli jatkotestauksenkin perusteella voidaan todeta työkalun todella sopivan projektin käyttöön, voidaan työkalua esitellä asiakkaalle, joka päättää työkalun käyttöönottamisesta. Tässä vaiheessa varsinkin työkalun käyttökustannus saattaa osoittautua käyttöönoton kannalta merkittäväksi tekijäksi. Vaihtoehtona voidaan myös harkita, että työkalu otetaan käyttöön pelkän asiakasprojektin sijaan Younite-AI-yrityksessä. Tällöin työkalua voidaan käyttää myös yrityksen muiden projektien ohjelmistotuotteiden automaattiseen testaukseen. Tässä vaihtoehdossa työkalun kustannukset jäävät kuitenkin yrityksen omalle vastuulle.

LÄHTEET

Atlassian 2023. Software testing in continuous delivery. Hakupäivä 20.2.2023. https://www.atlassian.com/continuous-delivery/software-testing.

Baumgartner, Manfred; Steirer, Thomas; Wendland, Marc-Florian; Gwihs, Stefan; Hartner, Julian & Seidl, Richard 2022. Test Automation Fundamentals. USA: Rocky Nook. Hakupäivä 1.3.2023. O'Reilly Online Learning. Vaatii käyttöoikeuden.

Campbell, Abbey; Morgan, Bradley; Sakuma, Ken & Stover, Adam 2021. Readme. Hakupäivä 24.2.2023. https://github.com/KabaLabs/Cypress-Recorder/#readme.

Chahine, Marc Hage 2020. How to improve the robustness of a test? Hakupäivä 22.2.2023. https://www.agilitest.com/blog/reasons-for-test-automation-failure-lack-of-robustness.

Cypress 2023a. CI Provider Examples. Hakupäivä 24.2.2023. https://docs.cypress.io/guides/continuous-integration/ci-provider-examples.

Cypress 2023b. Command Line. Hakupäivä 17.3.2023. https://docs.cypress.io/guides/guides/command-line.

Cypress 2023c. Cross Browser Testing. Hakupäivä 27.2.2023. https://docs.cypress.io/guides/guides/cross-browser-testing.

Cypress 2023d. Cypress Cloud. Hakupäivä 24.2.2023. https://docs.cypress.io/guides/cloud/introduction.

Cypress 2023e. Cypress Studio. Hakupäivä 20.3.2023. https://docs.cypress.io/guides/references/cypress-studio.

Cypress 2023f. Flexible pricing for Cypress Cloud. Hakupäivä 24.2.2023. https://www.cypress.io/pricing/.

Cypress 2023g Installing Cypress. Hakupäivä 24.2.2023. https://docs.cypress.io/guides/getting-started/installing-cypress.

Cypress 2023h. Integration for Slack. Hakupäivä 24.2.2023. https://docs.cypress.io/guides/cloud/slack-integration.

Cypress 2023i. Opening the App. Hakupäivä 17.3.2023. https://docs.cypress.io/guides/getting-started/opening-the-app.

Cypress 2023j. Why Cypress? Hakupäivä 20.2.2023. https://docs.cypress.io/guides/overview/why-cypress.

Cypress 2023k. Writing and Organizing Tests. Hakupäivä 29.3.2023. https://docs.cypress.io/guides/core-concepts/writing-and-organizing-tests.

Cypress 2023l. Writing Your First E2E Test. Hakupäivä 24.2.2023. https://docs.cypress.io/guides/end-to-end-testing/writing-your-first-end-to-end-test.

Elusoji, Sodeeq 2022. Comparing screenshots in Cypress. Hakupäivä 23.2.2023. https://reflect.run/articles/comparing-screenshots-in-cypress/.

Hamilton, Thomas 2023a. What is Regression Testing? Test Cases (Example). Hakupäivä 28.3.2023. https://www.guru99.com/regression-testing.html.

Hamilton, Thomas 2023b. What is Software Testing? Definition. Hakupäivä 20.2.2023. https://www.guru99.com/software-testing-introduction-importance.html.

Hiltunen, Simo 2023. Lead developer. Younite-AI. Haastattelu 27.2.2023.

Katara, Lakhman 2021. Levels of Testing in Software Testing. Hakupäivä 21.2.2023. https://qacraft.com/levels-of-testing-in-software-testing/.

Khorikov, Vladimir 2021. Unit testing: principles, practises, and patterns. First edition. New York: Manning. Hakupäivä 24.2.2023. O'Reilly Online Learning. Vaatii käyttöoikeuden.

Pittet, Sten 2023. The different types of software testing. Hakupäivä 24.2.2023. https://www.atlassian.com/continuous-delivery/software-testing/types-of-software-testing.

Rehkopf, Max 2023a. Automated software testing. Hakupäivä 25.1.2023. https://www.atlassian.com/continuous-delivery/software-testing/automated-testing.

Rehkopf, Max 2023b. Continuous integration tools. Hakupäivä 29.3.2023. https://www.atlassian.com/continuous-delivery/continuous-integration/tools.

Schmitt, Jacob 2022. What is end-to-end testing? Hakupäivä 10.3.2023. https://circleci.com/blog/what-is-end-to-end-testing/.

Smartbear 2023. What is End-To-End Testing? Hakupäivä 6.3.2023. https://smartbear.com/learn/automated-testing/what-is-end-to-end-testing/.

Software Testing Fundamentals 2022a. Integration Testing. Hakupäivä 3.3.2023. https://softwaretestingfundamentals.com/integration-testing/.

Software Testing Fundamentals 2022b. Unit Testing. Hakupäivä 2.3.2023. https://softwaretestingfundamentals.com/unit-testing/.

Testim 2021a. Bitbucket Integration. Hakupäivä 10.3.2023. https://help.testim.io/docs/bitbucket-integration.

Testim 2021b. CI integrations. Hakupäivä 10.3.2023. https://help.testim.io/docs/integrate-testim-to-your-ci.

Testim 2021c. Connecting Testim to Jira. Hakupäivä 10.3.2023. https://help.testim.io/docs/connecting-testim-to-jira.

Testim 2021d. Creating your First Coded Test using TDK. Hakupäivä 1.3.2023. https://help.testim.io/docs/getting-started.

Testim 2022a. Connecting Testim to Slack. Hakupäivä 9.3.2023. https://help.testim.io/docs/connecting-testim-to-slack.

Testim 2022b. Core concepts – Testim Automate. Hakupäivä 28.2.2023. https://help.testim.io/docs/core-concepts.

Testim 2022c. Creating your First Codeless Test in Testim Visual Editor. Hakupäivä 27.2.2023. https://help.testim.io/docs/creating-your-first-codeless-test.

Testim 2022d. Groups. Hakupäivä 14.4. 2023. https://help.testim.io/docs/groups.

Testim 2022e. Scheduler. Hakupäivä 8.3.2023. https://help.testim.io/docs/scheduler.

Testim 2022f. Visual Validation (element, viewport, full-page). Hakupäivä 10.3.2023. https://help.testim.io/docs/pixel-validation-and-pixel-wait-for.

Testim 2023a. Creating your First Mobile Test in Testim Visual Editor. Hakupäivä 27.2.2023. https://help.testim.io/docs/creating-your-first-mobile-test-in-testim-visual-editor-edited.

Testim 2023b. Grid Management. Hakupäivä 22.3.2023. https://help.testim.io/docs/grid-management.

Testim 2023c. Pricing. Hakupäivä 10.3.2023. https://www.testim.io/pricing/.

Testim 2023d. Recording a Mobile test. Hakupäivä 30.3.2023. https://help.testim.io/docs/recording-a-mobile-test.

Testim 2023e. Running tests on multiple browsers. Hakupäivä 27.2.2023. https://help.testim.io/docs/running-tests-on-multiple-browsers.

Testim 2023f. Testim overview. Hakupäivä 27.2.2023. https://help.testim.io/docs.

Testsigma 2022a. Create a Testsigma add-on. Hakupäivä 20.3.2023. https://testsigma.com/docs/addons/create/.

Testsigma 2022b. Get Started with Automating Android Apps. Hakupäivä 16.3.2023. https://testsigma.com/tutorials/getting-started/automate-android-applications/.

Testsigma 2022c. Get Started with Automating iOS Apps. Hakupäivä 16.3.2023. https://testsigma.com/tutorials/getting-started/automate-ios-applications/.

Testsigma 2022d. Get Started with Automating Web Applications. Hakupäivä 16.3.2023. https://testsigma.com/tutorials/getting-started/automate-web-applications/.

Testsigma 2022e. How to automate tests for iOS Apps using local devices. Hakupäivä 22.3.2023. https://testsigma.com/tutorials/test-cases/mobile-apps/build-tests-using-local-ios-devices/.

Testsigma 2022f. Integrate Testsigma with Bitbucket CI. Hakupäivä 23.3.2023. https://testsigma.com/docs/continuous-integration/bitbucket-ci/.

Testsigma 2022g. Integrating Slack with Testsigma for test run notifications. Hakupäivä 22.3.2023. https://testsigma.com/docs/integrations/collaboration/slack/.

Testsigma 2022h. Integrations in Testsigma. Hakupäivä 22.3.2023. https://testsigma.com/docs/integrations/overview/.

Testsigma 2022i. Overview of Elements for a Web Project. Hakupäivä 20.3.2023. https://testsigma.com/docs/elements/web-apps/overview/.

Testsigma 2022j. Schedule Test Plans. Hakupäivä 21.3.2023. https://testsigma.com/docs/test-management/test-plans/schedule-plans/.

Testsigma 2022k. Setup Testsigma. Hakupäivä 22.3.2023. https://testsigma.com/docs/getting-started/setup/overview/.

Testsigma 2022l. Testsigma Agent – Overview. Hakupäivä 22.3.2023. https://testsigma.com/docs/agent/overview/.

Testsigma 2022m. Testsigma Community Cloud. Hakupäivä 22.3.2023. https://testsigma.com/docs/getting-started/testsigma-community-cloud/.

Testsigma 2022n. Testsigma Docker Setup. Hakupäivä 22.3.2023. https://testsigma.com/docs/getting-started/setup/docker/.

Testsigma 2022o. Visual Testing – Configure Test Steps. Hakupäivä 23.3.2023. https://testsigma.com/docs/visual-testing/configure-test-steps/.

Testsigma 2023a. Automated mobile testing that works for you, not the other way around. Hakupäivä 16.3.2023. https://testsigma.com/automated-mobile-app-testing.

Testsigma 2023b. Automated web app testing that scales with you. Hakupäivä 16.3.2023. https://testsigma.com/automated-web-application-testing.

Testsigma 2023c. Cloud-based end-to-end automated API testing tool. Hakupäivä 16.3.2023. https://testsigma.com/automated-api-testing.

Testsigma 2023d. Run Tests on Desktop and Mobile Browsers, Real iOS, and Android Devices. Hakupäivä 22.3.2023. https://testsigma.com/test-lab.

Testsigma 2023e. Self Healing Test Automation | Tests that do Not Require Maintenance. Hakupäivä 20.3.2023. https://testsigma.com/blog/self-healing-tests-maintenance-testsigma/.

Testsigma 2023f. Tiered to fit your team's automation needs. Scalable to grow as you do. Hakupäivä 21.3.2023. https://testsigma.com/pricing.

Testsigma 2023g. Testsigma verkkosivut. Hakupäivä 15.3.2023. https://testsigma.com/.

Tricentis 2016. Blog: 100+ Best Software Testing Tools Reviewed (Research Done for You!) Hakupäivä 15.3.2023. https://www.tricentis.com/blog/100-plus-best-software-testing-tools?utm_source=google&utm_medium=paidsearch&utm_campaign=Blog_Search_DSA_High_EMEA_EN&utm_term=&gclid=Cj0KCQjw2cWgBhDYARIsALggUhoiw-8WfAZX93Y5QUhmSDsm5zR4FIm4xR3KuSWwerwMMcAHOs4XYZgaAvAPEALw_wcB.

Tricentis 2023. Tricentis Testim Datasheet. Hakupäivä 20.2.2023. https://be.tricentis.com/media-assets/pdf/Tricentis-data-sheet_Testim.pdf.

LIITTEET

Verkkosovelluksen sisäänkirjautumistesti Cypress-työkalulla liite 1

Verkkosovelluksen sisäänkirjautumistesti Testim-työkalulla liite 2

Android-sovelluksen sisäänkirjautumistesti Testim-työkalulla liite 3

iOS-sovelluksen sisäänkirjautumistesti Testim-työkalulla liite 4

Verkkosovelluksen sisäänkirjautumistesti Testsigma-työkalulla liite 5

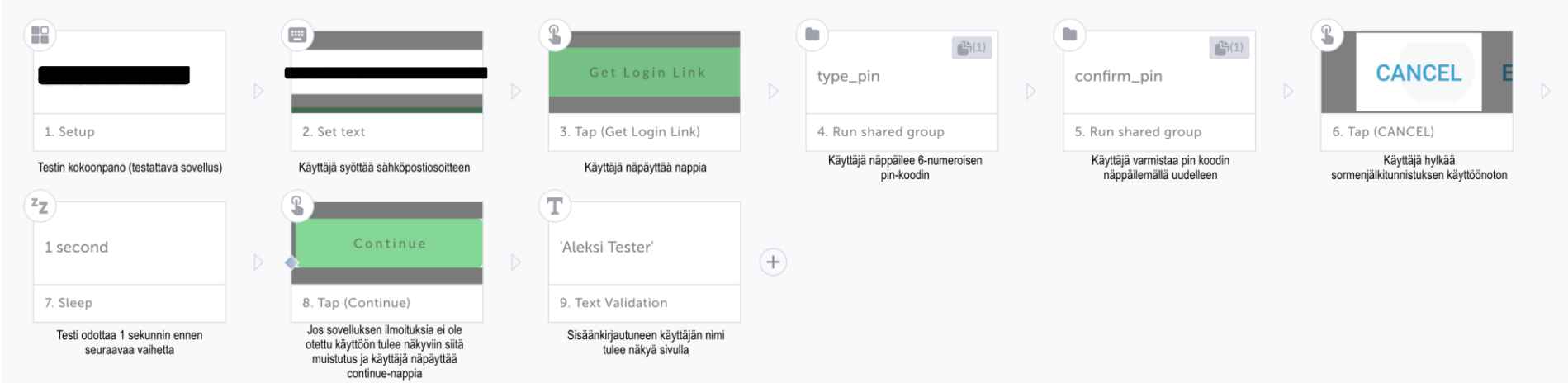
Android-sovelluksen sisäänkirjautumistesti Testsigma-työkalulla liite 6

iOS-sovelluksen sisäänkirjautumistesti Testsigma-työkalulla liite 7

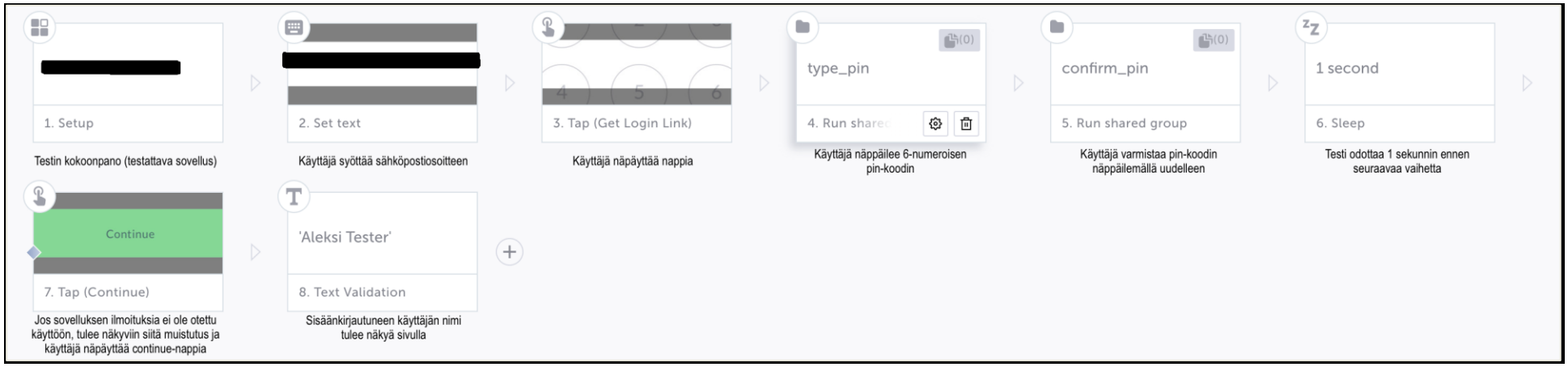
```
25
26  it('User logged in succesfully', () => {
27    cy.visit([redacted])
28
29    // find input for username and type value
30    cy.get('input[id="loginUsername"]').type('aleksi.nokela@faketestmail.com')
31
32    // find input or password and type password
33    cy.get('input[id="loginPassword"]').type('faketestpswd')
34
35    // find login button and and click the button
36    cy.get('button[id="loginSubmitButton"]').click()
37
38     // check if the name of logger is found on page
39    cy.get('span[id="Dashboard-header-welcome-name"]').contains('Aleksi')
40  })
```

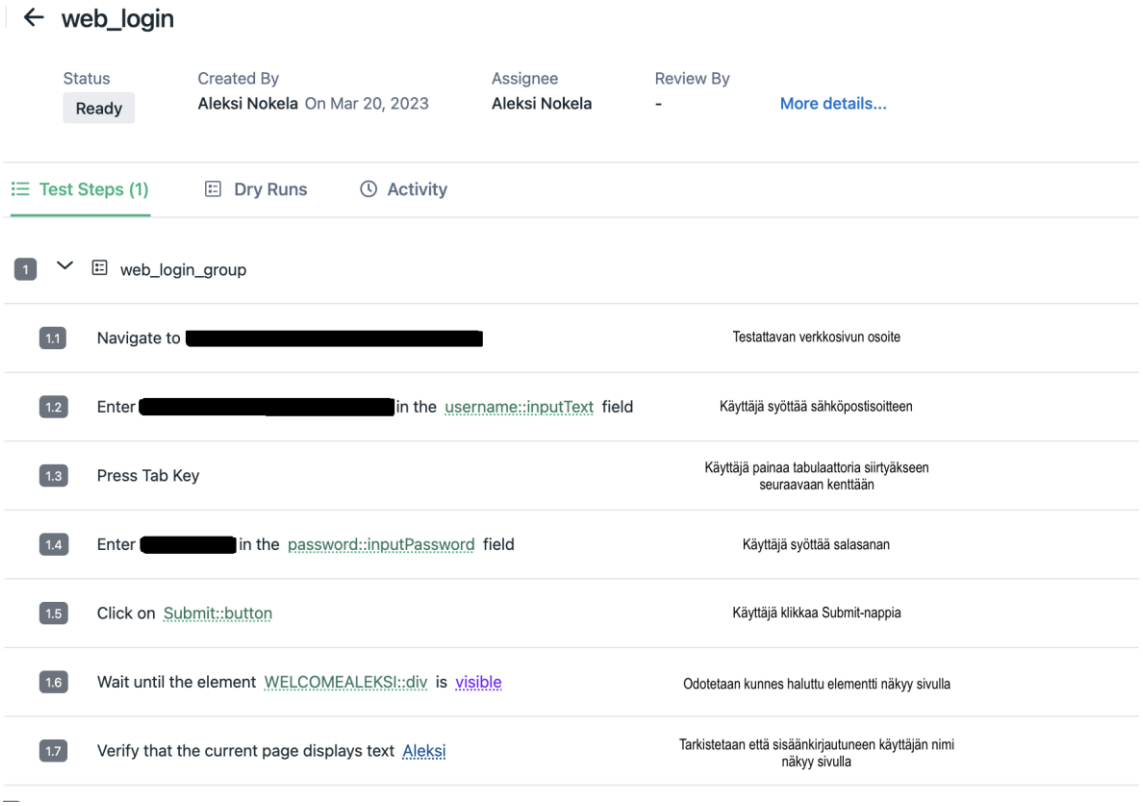
```
"use strict";

const { Locator, test, go, resize, click, l, type, sendCharacter, exists } = require('testim');

Locator.set(require('./locators/locators.js'));

test("login_test", async () => {
  await go("[redacted]");
  await resize({width: 1024, height: 768});
  await web_login();
});

// shared steps \\
async function web_login() {

  // user clicks the input field for email
  await click(l("Email"));

  // user types in email address
  await type(l("Email"), 'aleksi.nokela@faketestmail.com');

  // user moves to password input by pressing tab
  await sendCharacter(l("Email"), '\t');

  // user could also click the password input
  // await click(l("Password"));

  // user types in the password
  await type(l("Password"), 'faketestpswd');

  //user clicks the submit-button
  await click(l("Submit"));

  // check if the namew of the logger is found on the page
  await exists(l("ALEKSI"));
}
```

ANDROID-SOVELLUKSEN SISÄÄNKIRJAUTUMISTESTI TESTIM-TYÖKALULLA LIITE 3



IOS-SOVELLUKSEN SISÄÄNKIRJAUTUMISTESTI TESTIM-TYÖKALULLA LIITE 4





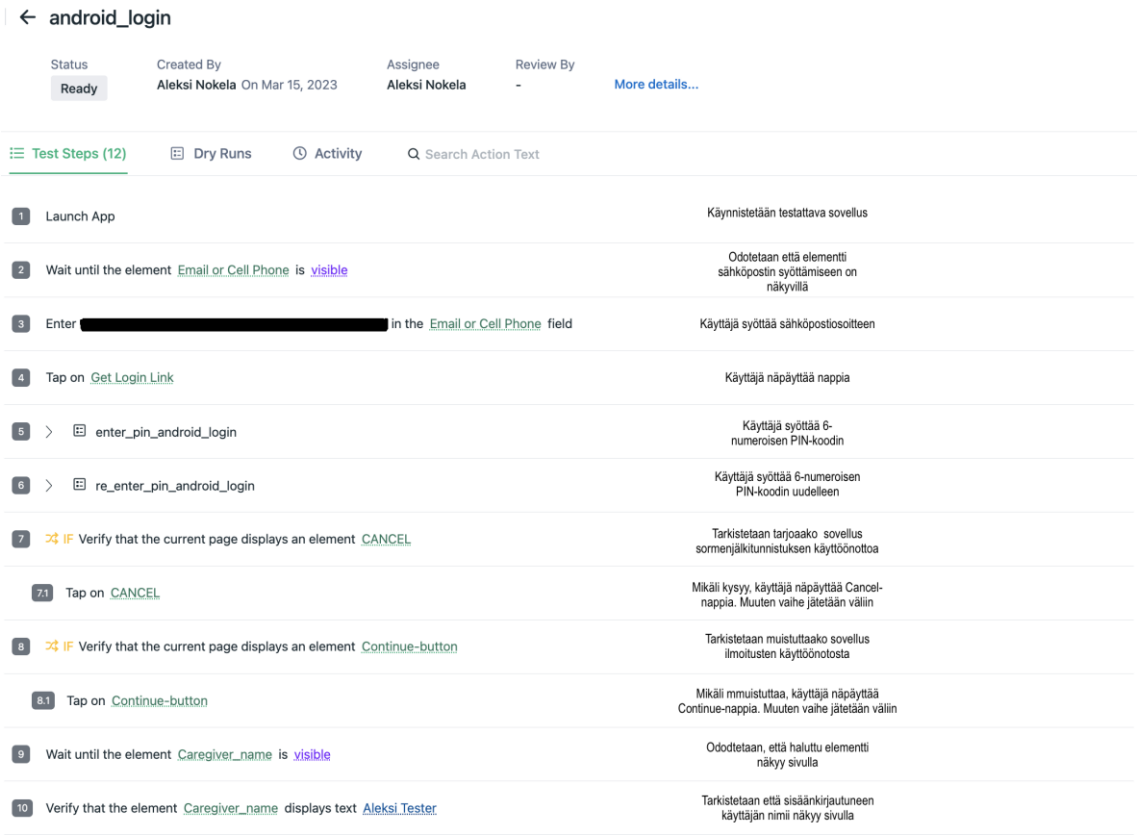
← web_login

Status	Created By	Assignee	Review By	
Ready	Aleksi Nokela On Mar 20, 2023	Aleksi Nokela	-	More details...

Test Steps (1) | Dry Runs | Activity

1 web_login_group

1.1	Navigate to [redacted]	Testattavan verkkosivun osoite
1.2	Enter [redacted] in the username::inputText field	Käyttäjä syöttää sähköpostisoitteen
1.3	Press Tab Key	Käyttäjä painaa tabulaattoria siirtyäkseen seuraavaan kenttään
1.4	Enter [redacted] in the password::inputPassword field	Käyttäjä syöttää salasanan
1.5	Click on Submit::button	Käyttäjä klikkaa Submit-nappia
1.6	Wait until the element WELCOMEALEKSI::div is visible	Odotetaan kunnes haluttu elementti näkyy sivulla
1.7	Verify that the current page displays text Aleksi	Tarkistetaan että sisäänkirjautuneen käyttäjän nimi näkyy sivulla

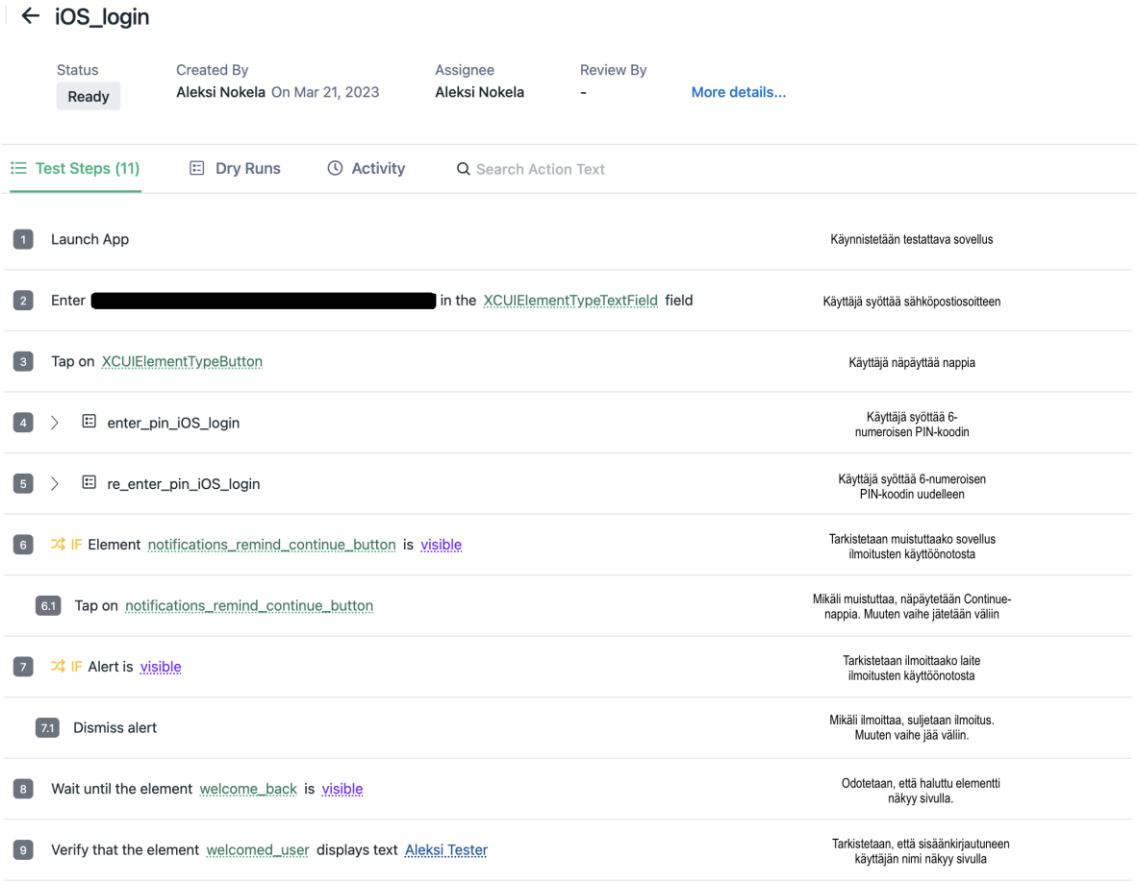


← android_login

Status	Created By	Assignee	Review By	
Ready	Aleksi Nokela On Mar 15, 2023	Aleksi Nokela	-	More details...

Test Steps (12) | Dry Runs | Activity | Search Action Text

1	Launch App	Käynnistetään testattava sovellus
2	Wait until the element Email or Cell Phone is visible	Odotetaan että elementti sähköpostin syöttämiseen on näkyvillä
3	Enter [redacted] in the Email or Cell Phone field	Käyttäjä syöttää sähköpostiosoitteen
4	Tap on Get Login Link	Käyttäjä näpäyttää nappia
5	enter_pin_android_login	Käyttäjä syöttää 6-numeroisen PIN-koodin
6	re_enter_pin_android_login	Käyttäjä syöttää 6-numeroisen PIN-koodin uudelleen
7	IF Verify that the current page displays an element CANCEL	Tarkistetaan tarjoaako sovellus sormenjälkitunnistuksen käyttöönottoa
7.1	Tap on CANCEL	Mikäli kysyy, käyttäjä näpäyttää Cancel-nappia. Muuten vaihe jätetään väliin
8	IF Verify that the current page displays an element Continue-button	Tarkistetaan muistuttaako sovellus ilmoitusten käyttöönotosta
8.1	Tap on Continue-button	Mikäli mmuistuttaa, käyttäjä näpäyttää Continue-nappia. Muuten vaihe jätetään väliin
9	Wait until the element Caregiver_name is visible	Ododtetaan, että haluttu elementti näkyy sivulla
10	Verify that the element Caregiver_name displays text Aleksi Tester	Tarkistetaan että sisäänkirjautuneen käyttäjän nimii näkyy sivulla


iOS_login
Status
Ready
Created By
Aleksi Nokela On Mar 21, 2023
Assignee
Aleksi Nokela
Review By
-
More details...
Test Steps (11)
Dry Runs
Activity
Search Action Text
1 Launch App
Käynnistetään testattava sovellus
2 Enter [redacted] in the XCUIElementTypeTextField field
Käyttäjä syöttää sähköpostiosoitteen
3 Tap on XCUIElementTypeButton
Käyttäjä näpäyttää nappia
4 enter_pin_iOS_login
Käyttäjä syöttää 6-numeroisen PIN-koodin
5 re_enter_pin_iOS_login
Käyttäjä syöttää 6-numeroisen PIN-koodin uudelleen
6 IF Element notifications_remind_continue_button is visible
Tarkistetaan muistuttaako sovellus ilmoitusten käyttöönotosta
6.1 Tap on notifications_remind_continue_button
Mikäli muistuttaa, näpäytetään Continue-nappia. Muuten vaihe jätetään väliin
7 IF Alert is visible
Tarkistetaan ilmoittaako laite ilmoitusten käyttöönotosta
7.1 Dismiss alert
Mikäli ilmoittaa, suljetaan ilmoitus. Muuten vaihe jää väliin.
8 Wait until the element welcome_back is visible
Odotetaan, että haluttu elementti näkyy sivulla.
9 Verify that the element welcomed_user displays text Aleksi Tester
Tarkistetaan, että sisäänkirjautuneen käyttäjän nimi näkyy sivulla