OHJELMISTON TESTAUSSUUNNITELMA

Case: Acrelec Finland Oy

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Hämeenlinnan korkeakoulukeskus
Tietojenkäsittelyn koulutusohjelma

Syksy, 2020

Milka Ukkonen

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutusohjelma
Hämeenlinnan korkeakoulukeskus

Tekijä Milka Ukkonen **Vuosi** 2020

Työn nimi Ohjelmiston testaussuunnitelma

Työn ohjaaja/t Lauri Salminen

TIIVISTELMÄ

Tänä päivänä yhä useampiin tuotteisiin sisältyy runsaasti erilaisia ohjelmia, ohjelmistoja ja pilviratkaisuja. Jotta nämä ratkaisut toimivat parhaiten yhdessä, on ohjelmat ja ohjelmistot testattava säännöllisesti. Erilaisia testausmenetelmiä ja tapoja on monenlaisia ja onkin tärkeää löytää ohjelmistolle sopivin testausmenetelmä. Testausprosessi alkaa kuitenkin ensin testaussuunnitelman laatimisesta.

Tämän opinnäytetyön tarkoituksena oli luoda kohdeyritys Acrelec Finland Oy:lle heidän kioskisovelluksensa testaussuunnitelma. Testausympäristönä toimi Fafa's DOT-sovellus, joka on Acrelecin luoma Direct Order Taker-sovellus ja on käytössä Fafa's ravintoloissa Suomessa. Sovelluksen avulla asiakas voi tehdä itse tilauksensa Fafa's ravintolassa. Acrelec toivoi selkeää, täsmällistä ja järjestelmällistä step-by-step-testaussuunnitelmaa, jotta jatkossa sovellus testattaisiin aina samalla tavalla. Samalla tutkittiin, onko mahdollista automatisoida jotkin testausvaiheet.

Testaussuunnitelmaan luotiin 10 erilaista testitapausta, jotka kattoivat asiakkaan näkökulmasta kaikki oleellisimmat tilauksentekovaiheet. Testaaminen tapahtui manuaalisesti ja se oli pitkälti toiminnallista- ja käytettävyystestaustausta. Testaussuunnitelmalla varmistetaan, että ohjelmisto toimii aina toivotulla tavalla ja testaus suoritetaan loppukäyttäjän, eli asiakkaan, näkökulmasta. Testitapausten automatisoiminen jouduttiin jättämään työstä pois, koska ajallisesti resurssit eivät riittäneet. Jokainen testitapaus on kuitenkin mahdollista automatisoida käyttäen esimerkiksi Robot Framework-ohjelmaa.

Avainsanat Testaussuunnitelma, ohjelmistotestaus, testausmenetelmät, testitapaus

Sivut 46 sivua, joista liitteitä 12 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Hämeenlinna University Centre

Author Milka Ukkonen **Year** 2020

Subject A Software test plan

Supervisors Lauri Salminen

ABSTRACT

Nowadays, more and more products contain a wide variety of programs, software, and cloud solutions. In order for these solutions to work together, programs and software must be tested regularly. There are many different test methods, and it is important to find the most suitable test method for the software. However, the testing process should begin first with a test plan development.

The purpose of this thesis was to create a test plan for a target company's, Acrelec Finland Ltd, Kiosk application. The testing environment was Fafa's DOT application, which is a Direct Order Taker-application created by Acrelec and it is used in Fafa's restaurants in Finland. The application allows the customer to place their order on their own at Fafa's restaurants. Acrelec hoped for a clear, precise, and systematic step-by-step test plan so that in the future the Kiosk application would always be tested in the same way. It was also investigated if it is possible to automatize some of the test steps.

Ten different test cases were created in the test plan, covering all the most relevant ordering steps for the customer's point of view. Testing was done manually, and it was mainly functional and usability testing. The test plan ensures that the application always works as desired and that the testing is performed from the customers perspective. The automatization of the test cases had to be left out of the work because it was not possible timewise. However, it is possible to automatize each test case by using, for example, The Robot Framework program.

Keywords Test plan, software testing, test methods, test case

Pages 46 pages including appendices 12 pages

SISÄLLYS

1 JOHDANTO

Tänä päivänä useimpiin tuotteisiin sisältyy runsaasti erilaisia ohjelmistoja ja pilviratkaisua, usein myös kolmansien osapuolten sovelluksia. Jotta nämä erilaiset ratkaisut toimisivat parhaiten yhdessä, on ohjelmistot testattava huolellisesti. Mitä laajemmasta ohjelmistosta on kyse, sitä tärkeämpää sitä on testata säännöllisesti.

Tämän opinnäytetyön tarkoituksena on luoda Acrelec Finland Oy:lle kioskisovelluksen testaussuunnitelma. Testausympäristönä tulee toimimaan Fafa's DOT-sovellus, joka on Acrelecin luoma Direct Order Taker-sovellus. Fafa's-ravintolan kioskitilauskehys on luotu tähän Direct Order Taker-sovellukseen. DOT on tällä hetkellä yksi maailman johtavista kioskitilauskehyksistä. (Acrelec Finland, n.d.)

Tavoitteena on luoda yritykselle selkeä, täsmällinen ja järjestelmällinen step-by-step-testaussuunnitelma, jotta jatkossa ohjelmisto testattaisiin aina samalla tavalla. Testaussuunnitelma tulee kattamaan kaikki Fafa's DOT-sovelluksen kautta tehdyn tilauksen tekovaiheet aloituksesta maksamiseen saakka, asiakkaan näkökulmasta. Testaussuunnitelmassa tutkitaan samalla, mitkä testausvaiheet olisi mahdollista automatisoida käyttäen Robot Frameworkia.

Opinnäytetyö tulee vastaamaan seuraaviin kysymyksiin:

- Miten ohjelmisto kannattaa testata ennen kuin se luovutetaan asiakkaan käyttöön?
- Mitä vaiheita testaamisessa on otettava huomioon?
- Missä järjestyksessä testaaminen kannattaa tehdä?
- Mitkä testivaiheet voidaan automatisoida?

2 ACRELEC FINLAND OY

Acrelec on maailmanlaajuinen teknologiayritys, joka on keskittynyt ravintola- ja vähittäiskauppabrändien asiakaskokemuksiin ja niiden uudelleenkehittämiseen. Yrityksellä on vuosikymmenien kokemus ohjelmisto-, laitteisto- ja palveluosaamisesta, jonka avulla yritys kehittää ja integroi uusia alustaratkaisuja, jotka lisäävät asiakkaiden sitoutumista, optimoivat tehokkuutta ja parantavat toimintaa. Yrityksellä on maailmanlaajuisesti 800 työntekijää. (Acrelec, n.d.)

Acrelec Finland Oy on Hollolassa sijaitseva, digitaalisia polkuja asiakkaille tarjoava yritys, jossa työskentelee 25 henkilöä. Acrelec Groupin pääkonttori sijaitsee Ranskassa. Samalla alueella sijaitsee myös tehdas, jossa valmistetaan laitteistot asiakkaiden brändien mukaisesti. Acrelec Finland tekee tiiviisti yhteistyötä pohjoismaisten Acrelec Ruotsin ja Acrelec Tanskan toimistojen kanssa. Acrelecilla on myös omat myyjät, projektipäälliköt, ohjelmistokehittäjät, asentajat ja tukipalvelut. (Acrelec Finland, n.d.)

Acrelecin tuottamia digitaalisia ratkaisuja ovat mm. verkkokauppatilausten noutopalvelu Click & Collect, itsepalvelukioskit, näytöt ja kolmannen osapuolen järjestelmät (esim. mobiilisovellukset). Acrelec myös integroi myyntipisteiden keskeisten järjestelmien tiedot Acrelec Bridgeen ja välittää ne asiakaskokemuksen kanaviin. Acrelecilla on maailmanlaajuisesti yli 200 suurta asiakasta ja yli 20 000 myyntipistettä eri digitaalisten ratkaisujen koko elinkaaren ajan. (Acrelec Finland, n.d.)

2.1 Acrelec DOT-Kioskiohjelmisto

Kuluttajat haluavat käyttää tänä päivänä yhä useammin digitaalisia kanavia. Tämän takia Acrelec on kehittänyt uniikin itsepalvelukioskin, jolla asiakas kykenee tekemään tilauksensa itse. Acrelec tarjoaa tähän sekä ohjelmiston, että laitteen asiakkaan käyttöön. (Acrelec Kiosk, n.d.)

Acrelecin luoma DOT (Digital Order Taker) on yksi maailman johtavista kioskitilauskehyksistä, joka on liitetty useimpiin jo olemassa oleviin POS-ratkaisuihin. Se valjastaa tekoälyn räätälöimään jokaisen ”matkan” tuotesuosituksilla, jotka perustuvat säähän, liikenteeseen, aikaan ja ostoskoriin lisättyihin tuotteisiin. DOT on joustava ja mahdollistaa testattujen sisältöstrategioiden ja käyttöliittymäanalyysien parantaa asiakaskokemuksia. (Acrelec Kiosk, n.d.)

Acrelecin tarjoamat sovellusliittymät (API) auttavat kehittäjiä hallitsemaan, optimoimaan, näyttämään ja lisäämään tilauksia kaikille digitaalisille laitteille ja kanaville. Puhutaan ns. Bridge-palvelusta. Bridge tarjoaa laajan POS-integraatiokirjaston, jossa on yli 50 esirakennettua laajennusta. Bridge:sta on olemassa Bridge Loyalty, Experiment ja Personalise – moduulit, joiden avulla asiakas saa ainutlaatuiset kokemukset brändista. (Acrelec Kiosk, n.d.)

ATP (Acrelec Transformation Platform) mahdollistaa koko laitekannan etähallinnan. Se mahdollistaa laitteiston ja oheislaitteiden valvonnan, ohjainten hallinnan, ohjelmistojen versioinnin ja tietoturvapalvelut sekä Acrelecin omistamille, että kolmansien osapuolien sovelluksille. (Acrelec Kiosk, n.d.)

3 OHJELMISTOTESTAUS

Lyytisen (2019) mukaan ohjelmistotestaus on ensisijaisesti tuntemattoman tutkimista. Jokainen sovellus ja ohjelmisto on lukuisien erilasten riippuvuuksien, sisäisten tilojen, tilamuutosten, käyttöjärjestelmän ja muiden koneella samanaikaisesti ajettavien ohjelmistojen välisten vuorovaikutusten kompleksinen sekoitus. Ohjelmistokehittäjän tehtävä on tuottaa koodia ja tämä tapahtuu usein hyvin tiukoissa aikarajoissa. Tämän takia kehittäjä ei aina pysty ottamaan kaikkia vaikuttavia tekijöitä huomioon. Tässä avuksi tulee testaajat, joiden tehtävänä on tutkia näitä erilaisia tekijöitä, vuorovaikutuksia ja niiden yhdistelmiä, sekä selvittää minkälaisia ei-toivottuja vaikutuksia niistä voi aiheutua. (Lyytinen, 2019)

Tärkeä osa ohjelmistotestaajan työtä on tukea muuta ohjelmistotiimiä pyrkimällä tunnistamaan ohjelmiston puutteita, ristiriitaisuuksia ja väärinkäsityksiä. Ongelmia voi olla esimerkiksi ohjelmiston koodissa tai suunnittelussa. Jotkut ongelmat voivat myös olla täysin inhimillisten tekijöiden, kuten esimerkiksi automaattisten oletusten, aiheuttamia. Ohjelmistotestauksessa onkin hyvin pitkälti kyse kriittisestä ajattelusta ja sellaisten kysymysten kysymisestä, joilla pyritään selvittämään, miten ohjelmisto saattaisi toimia ei-toivotulla tavalla, tai olla toimimatta kokonaan. Minkälaisia riskejä esimerkiksi jokin tietty suunnittelu- tai toteutuspäätös voi tuoda mukanaan? Miten ohjelmistoa saatettaisiin väärinkäyttää? (Lyytinen, 2019)

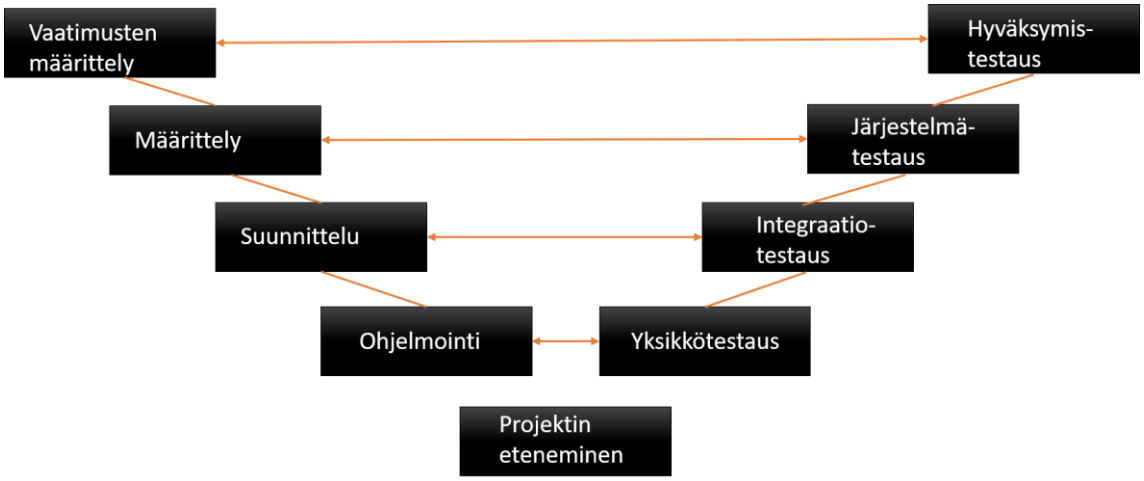
Ohjelmistotestauksen tärkein tehtävä on tuottaa tietoa. Sen tehtävä on siis selvittää mikä ohjelmiston nykytila oikeasti on verrattuna siihen, mitä jokin määrittely sanoo, että sen pitäisi olla. Testaaminen ei rajoitu aina pelkästään ohjelmistototeutuksen tutkimiseen. On mahdollista testata myös ideoita, suunnitelmia, odotuksia tai määrittelyjä. (Lyytinen, 2019)

Ohjelmistotestaus on sitä, että varmistetaan, että tehdään oikeaa tuotetta ja että tuote on tehty oikein. Ohjelmistotestaus on siis vertailua siitä, että onko tuote tehty kuten suunniteltu ja täyttääkö se asiakkaan tarpeet. (Salminen, 2020)

Ohjelmistotestausta voidaan tehdä monella tasolla, mutta juuri kehitysvaiheessa tehtävä mekaaninen testaustyö on keskeisin osa koko testaustyötä. Testaus voidaan jakaa seuraaviin tasoihin:

- Yksikkötestaus, jossa testitapaus rajoittuu yhteen komponenttiin. Testeissä testataan yksiköitä, osia koodista. Siinä varmistetaan nimenomaan näiden pienien osien oikeellisuus. Nämä testit usein automatisoidaan, ja luodaan lähes aina samaan tahtiin muun kehitystyön kanssa.
- Integraatiotestaus, jossa testaus liittyy muutamaan komponenttiin ja niiden väliseen toimintaan, eli integraatioon.
- Järjestelmätestaus, joka on koko järjestelmän kattavaa testausta. Siinä keskitytään koko ohjelmistoon ja sen variaatioihin käytössä. Tätä edeltää kehityksen aikana tehdyt yksikkötestit.
- Hyväksymistestaus, joka suoritetaan kohdeympäristössä tai sitä simuloiden. (Salminen, 2020; Valagroup, n.d.)

Edellä mainittu, neljään vaiheeseen jaettu ohjelmistotestaus tunnetaan myös nimellä v-malli, jota havainnolistetaan kuvassa 1 (Kudjoi, 2018, s. 8).



Kuva 1. V-malli (Kudjoi, 2018, s. 8).

Kokonaisuudessa perinteiseen testausprosessiin kuuluu siis monia vaiheita. Näitä työvaiheita ovat ainakin testauksen suunnittelu, testitapausten luominen, testitapausten suorittaminen, testiajojen tulosten arviointi ja niistä raportointi. (Katara, 2015)

3.1 **Testauksen tasot**

Testausmenetelmistä tulee ensimmäiseksi hahmottaa ja ymmärtää termit staattinen ja dynaaminen. Ne määrittävät millaista testaamista kohdejärjestelmälle ollaan tekemässä. Vaikka ohjelmistotestaus on periaatteessa hyvinkin yksinkertaista, kokonaisuuden hallitseminen on kuitenkin laaja paketti, ja se vaatii monenlaista erityisosaamista. (Salminen, 2020; Valagroup, n.d.)

Staattinen testaaminen tarkoittaa sitä, että testattavaa ohjelmistoa ei suoriteta, vaan järjestelmää tutkitaan esim. koodianalysaattorilla tai koodiarvioinnilla ja yritetään löytää virheitä. Se on siis perustason ja sisäisen logiikan tarkastelua, jolla pyritään syntaksivirheiden poistoon. (Salminen, 2020; Katara, 2015)

Menetelmillä pyritään jo alkuvaiheessa poistamaan mahdolliset ilmenevät ongelmat, ennen kuin tarkempi testaus aloitetaan. Näin voidaan alkuvaiheessa jo saavuttaa hyviä tuloksia pienemmillä kustannuksilla. Staattinen testaus voidaan myös jo aloittaa, vaikka mitään toimivaa osaa kohdejärjestelmästä ei ole vielä toteutettu ja staattisessa testauksessa löydetyt viat ovat myös halpoja korjata. (Salminen, 2020)

Dynaaminen testaaminen on staattisen testaamisen vastakohta. Dynaaminen testaaminen tarkoittaa ohjelmiston ajamista sopivilla syötteillä. Testattavaa ohjelmistoa käytetään testaamisessa ja tutkitaan järjestelmän reagointia syötteisiin. Tätä voi olla esimerkiksi yksikkötestaus, integrointitestaus tai kuormitustestaus. (Salminen, 2020; Katara, 2015)

3.1.1 Yksikkötestaus

Yksikkötestaus tarkoittaa ohjelmiston pienimmän mahdollisen osan toiminnan testaamista. Näillä yksikkötesteillä varmistetaan, että ohjelman pienimmätkin osat toimivat odotetulla tavalla, ja että mahdolliset virhetilanteet on ennakoitu. On tärkeää, että kun yksikkötestejä kirjoitetaan ja ajetaan, tarkistellaan testitapauksen lopputulosta kokonaisvaltaisesti. Yksikkötestaamisen hyödyt näkyvät kehitysprosessin aikana erityisesti silloin, kun koodiin joudutaan tekemään muutoksia. Mikäli testit on automatisoitu, voidaan nopeasti todeta, aiheuttavatko muutokset virheitä. Yksikkötestit ovat tärkeässä roolissa erityisesti ketteriä menetelmiä käytettäessä, joissa uusia koontiversioita julkaistaan usein. (Smart education, n.d.)

3.1.2 Integraatiotestaus

Integraatiotestauksessa yritetään löytää virheitä, jotka eivät tulleet esiin yksikkötesteissä. Integraatiotestauksessa testataan monien ohjelman komponenttien yhteistoimintaa. Testeissä suoritetaan tietynlaisia polkuja, jotka hyödyntävät useita eri yksiköitä tai laajempia komponentteja, ja lopuksi tarkastellaan tuloksia niiden toiminnasta. Kun projekti etenee ja yksiköt valmistuvat, testattavien polkujen määrä lisääntyy ja samalla ne pitenevät. Tällöin on hyvä vaihtaa regressiotestaukseen, eli suoritetaan uudelleen aiemmin suoritettuja testejä, joilla pyritään varmistamaan, että tehdyt muutokset ja lisäykset eivät aiheuta ongelmia aiemmin toteutettujen elementtien toiminnassa. Regressiotestausta voidaan suorittaa joko manuaalisesti tai automatisoidusti tai näiden yhdistelmänä. Regressiotestaus voidaan jakaa kolmeen ryhmään:

- Testit, jotka hyödyntävät ohjelman kaikkia jo toteutettuja ominaisuuksia
- Testit, jotka tehdään toiminnoille, joihin muutokset todennäköisimmin vaikuttavat
- Komponenttien testit, jotka ovat muutoksien alla. (Smart education, n.d.)

3.1.3 Järjestelmätestaus

Järjestelmätestaus tarkoittaa kokonaisen ohjelmiston testaamista, jossa tarkastellaan vastaako ohjelmisto sille asetettuja vaatimuksia ja käyttötarkoitusta. Aitoon ympäristöön kuuluvat mm. käytettävä laitteisto, tietokannat ja ohjelmiston käyttäjät. Järjestelmätestausta voidaan pitää ns. tulikokeena, jossa sitä pyritään käyttämään siinä käyttötarkoituksessa ja ympäristössä, johon se on suunniteltu. (Smart education, n.d.)

Järjestelmätestausvaiheessa tulisi keskittyä ohjelmiston toiminnan varmistamiseen ja hienosäätöön. Mikäli testaamisen aiemmissa vaiheissa ei ole kiinnitetty tarpeeksi huomiota itse testaamiseen ja virheiden ehkäisyyn, huomataan se yleensä tässä vaiheessa. Ohjelmointivirheiden korjaaminen tässä vaiheessa on yleensä erittäin aikaa vievää ja kallista. Siksi järjestelmätestaukseen siirryttäessä, keskitytään mieluummin olemassa olevien ominaisuuksien hiomiseen, ja jätetään mahdollisten lisäominaisuuksien toteuttaminen myöhemmäksi. (Smart education, n.d.)

Mikäli tehdään korjauksia ja muutoksia, on olennaista, että ohjelmistoa voidaan testata kattavasti eritasoisten automatisoitujen testien avulla. Näin voidaan ilman manuaalista testausta varmistua siitä, että tarvittavat korjaukset eivät riko olennaisia toiminnallisuuksia ohjelmassa. (Smart education, n.d.)

Koska järjestelmätestaus on testaamisen tasoista laajin, jaetaan se useisiin osa-alueisiin. Jokaisella osa-alueella on omat tavoitteensa. Näitä testattavia osa-alueita voivat olla esimerkiksi rasituksen sieto, tietoturva, kapasiteettia koettelevien rasituspiikkien sieto, käytettävyys, suorituskyky, asennus ja luotettavuus. (Smart education, n.d.)

3.1.4 Hyväksymistestaus

Hyväksymistestauksella varmistetaan, että järjestelmä vastaa vaatimuksia. Hyväksymistestaukselle pyritään selvittämään kriittisten liiketoimintaprosessien toiminnan lisäksi esim. integraatioiden toimivuus, suorituskyky ja mahdolliset virhetilanteet. Se on, toisin sanoen, ohjelmistokokonaisuuden vaatimusten ja käyttötarkoituksen testaamista. (Testimate, n.d.)

Hyväksymistestaus perustuu siis pitkälti asiakkaan vaatimuksiin. Hyväksymistestaussuunnitelma on pidettävä ajan tasalla, sillä mikäli ohjelmistoon tilataan uusia ominaisuuksia kesken kaiken, tulee nekin hyväksymistestata. Hyväksymistestauksessa testataan siis kokonaista valmista tuotetta ja testaajina kannattaa käyttää tuotteen loppukäyttäjiä. Testiympäristön tulisi olla myös mahdollisimman lähellä todellista loppukäyttäjän ympäristöä. Hyväksymistestaus on ajallisesti melko lyhyt vaihe, koska sen tarkoituksena on vain osoittaa, että vaatimukset on täytetty. Tässä kohtaa virheiden löytyminen ei ole enää ensisijaisena tavoitteena. (Katara, 2011)

Hyväksymistestin tulisi olla lähinnä demonstraatio. Mikäli tässä vaiheessa löydetään virheitä, voi niiden korjaaminen tulla kalliiksi. Mikäli käyttäjät pääsevät tutustumaan ja käyttämään tuotetta vasta kun se on valmis, suurella todennäköisyydellä virheitä löydetään. Idealistisesti ajatellen, käyttäjien tulisi olla jossain muodossa mukana ohjelmistoprosessin vaiheissa. Suurten muutosten tekemistä projektin loppuvaiheessa pitää harkita tarkkaan. Muutostarpeiden priorisoinnin jälkeen, on järkevintä jättää virheiden korjaaminen vasta seuraaviin versioihin. Mikäli muutoksiin kuitenkin ryhdytään, pitää niihin liittyvät riskit kartoittaa uudelleen, esimerkiksi sen arvioimiseksi, että pitäisikö jokin ominaisuus jättää versiosta kokonaan pois tai sen testausta on vähennettävä. (Katara, 2011)

Erot testiympäristön ja loppukäyttäjän ympäristön välillä ovat helposti asioita, joita ei tule testattua. Mahdolliset erot ovat muun muassa siinä, että käytetäänkö todellisia tietokantoja tai muita testattavaan ohjelmistoon liittyviä järjestelmiä? Vai simuloidaanko niitä jotenkin? Onko testidata generoitua vai todellista dataa? Onko käyttäjän laitteistossa muita mahdollisia ohjelmia tai onko laitteiston konfiguraatio realistinen? (Katara, 2011)

Ideaalitapauksissa hyväksymistestauksen tekijät ovat loppukäyttäjiä tai heidän edustajiaan. Käyttäjät ymmärtävät parhaiten haettavia vaatimuksia ja omaan liiketoimintaansa liittyviä riskejä. Käyttäjät voivat myös toimittaa realistista testidataa, toimittaa käyttötapauksia, määritellä hyväksymistestin lopetusehdon, suorittaa ja tarkastaa ja katselmoida testiraportteja ja muita projektissa syntyneitä dokumentteja. (Katara, 2011)

3.2 **Muut testausmenetelmät**

Ohjelmistokehityksen V-mallia ei yleensä sovelleta sellaisenaan, mutta sen sisältämät testaamisen tasot ovat kuitenkin yleisesti käytössä. Termejä käytetään jonkin verran myös ristiin, mikä aiheuttaa helposti ristiriitoja. Yleisimmin testaaminen jaetaan kuitenkin yksikkö-, integraatio-, järjestelmä- ja hyväksymistestaukseen. (Smart education, n.d.)

Testaustasoja voi kuitenkin olla useita. Muita testaustasoja ovat muun muassa toiminnallinen- ja ei-toiminnallinen testaus. Toiminnallinen testaus on testausta, jonka tarkoituksena on varmistaa, että ohjelmisto toimii sille tarkoitetulla tavalla ja se täyttää asetetut vaatimukset toiminnallisuutensa puolesta. Toiminnallinen testaus tehdään usein loppukäyttäjän näkökulmasta ja se suoritetaan itse käyttöliittymää käyttäen. Ei-toiminnallinen testaus on vuorostaan taas testausta, jossa testataan nimenomaan muuta kuin näkyvää toiminnallisuutta, esim. suorituskykyä, vikasietoisuutta tai resurssien käyttöä. (Valagroup, n.d.)

Regressiotestaus pitää sisällään sekä toiminnallisen testauksen, että ei-toiminnallisen testauksen, mutta on luonteeltaan kuitenkin erilaista. Regressiotestauksessa tarkistetaan, että uudet muutokset ohjelmistossa eivät ole rikkoneet toimivia ominaisuuksia. Samalla tarkistetaan, että jo löydetyt virheet ovat edelleen korjattuja. (Valagroup, n.d.)

Käytettävyystestauksessa tutkitaan kuinka hyvin suunniteltu ja toiminnassa oleva ohjelmisto toimii. Testattavia ominaisuuksia ovat esimerkiksi ohjelmiston helppokäyttöisyys, ovatko värit, kontrastit ja siirtymät hyväksyttäviä tai joutuuko käyttäjä odottamaan toiminnallisuuksia ja niiden latauksia liian kauan. (Valagroup, n.d.)

Tutkiva testaus ei ole kovin suunnitelmallista, vaan se perustuu usein testaajan ammattitaitoon. Yleensä painopisteet on etukäteen sovittu ja testauksen aikana pidetään kirjaa asioista, jotka on tehty ja mahdollisista löydöksistä. Tutkiva testaus voi olla tehokas tapa käydä läpi asioita ilman suurta valmistelua tai suurta dokumentoinnin määrää. Tulokset voivat olla yllättäviäkin. (Valagroup, n.d.)

Mustalaatikkotestaus, eli black box testing, on perinteisin testausmuoto. Tässä testausmuodossa järjestelmälle annetaan syötteitä ja periaatteessa katsotaan mitä tapahtuu. Siitä mitä järjestelmän sisällä tapahtuu, ei välitetä. Normaalisti järjestelmälle annettavat syötteet tai tehtäväsarjat määritellään jo testitapauksissa, joissa kuvataan miten testattavaa järjestelmää/laitetta tulisi käyttää ja miten sen tulisi reagoida tehtäväsarjaan tai syötteeseen. Mustalaatikkotestaus on menetelmänä yksinkertainen ja sitä voidaan käyttää missä tahansa työvaiheessa. (Salminen, 2020)

Lasilaatikkotestaus, eli white box testing, on järjestelmän sisäisen toiminnan tarkastelua. Tässä testaamistavassa testaaja ns. ”näkee ohjelman sisälle”, eli miten annettua syötettä käsitellään. Tämä on paljon kattavampaa testaamista mustalaatikkotestaamiseen verrattuna ja vaatii testaajaksi järjestelmän hyvin tuntevan henkilön. (Salminen, 2020)

Harmaalaatikkotestaus, eli grey box testing, on yhdistelmä lasilaatikko- ja mustalaatikko -testaamisesta. Siinä tarkoituksena on yhdistää molemmista parhaimmat puolet. Se sopii parhaiten testeihin, joissa järjestelmää ei voida testata kokonaisvaltaisesti lasilaatikon avulla, mutta järjestelmän komponentteihin päästään kuitenkin käsiksi. (Salminen, 2020)

Validaatiotesteillä pyritään varmistamaan ohjelman toimivuus vaatimusten mukaisesti käytännön tilanteissa. Testitapaukset kirjoitetaan puhtaasti ohjelman vaatimusmäärittelyn pohjalta, ja niiden ohjenuorana toimivat käyttötapaukset, jotka on luotu suunnitteluvaiheessa. Näistä käyttötapauksista luodaan testejä, joiden avulla on mahdollista todeta ohjelman toimivuus käyttötilanteissa, ja yritetään löytää mahdollisia ongelmakohtia. Validaatiotesteillä pyritään siis ennen kaikkea varmistamaan, että ohjelmisto kykenee vastaamaan sille asetettuihin vaatimuksiin, ennen kuin siirrytään järjestelmätestaamiseen. Validaatiotestit kannattaa yrittää automatisoida, sillä kaikkia muutosten yhteydessä syntyneitä virheitä ei välttämättä ole kyetty löytämään alemman tason testeillä. (Smart education, n.d.)

3.3 **Manuaalinen testaus**

Manuaalinen testaus on tärkeää. Manuaalisesti testaamalla, ohjelmistosta löytyy enemmän virheitä, verrattuna automatisoituun testaamiseen. Virhe yleensä löytyykin ensin manuaalisella testillä, joka tämän jälkeen automatisoidaan regressiotestausta varten. (Katara, 2015)

Manuaaliset tehtävät testit eivät ole yleensä monimutkaisia, vaan testit voivat olla hyvinkin yksinkertaisia. Mikäli testitapauksia joutuu ajamaan useaan kertaan, kannattaa testin automatisointia harkita. Manuaalinen testaus on hyvää harjoitusta vasta-aloittaville testaajille, jotka vasta opettelevat mistä testaamisessa on kyse ja mitä sillä halutaan tavoitella. (Suomalainen, 2017, s. 9)

Manuaalinen testaus ei vaadi kummoisia IT-taitoja, sillä manuaaliseen testaukseen ei tarvitse välttämättä käyttää erikoisempia sovelluksia, vaan testitapauksia voi luoda esimerkiksi Microsoft Exceliin. Manuaalitestauksen hyvä puoli on kuitenkin, että virheet löytyvät manuaalisella testauksella helpommin. Manuaalinen testaus on parhaimmillaan myös tutkivaa testausta ja manuaalista testausta tarvitaan ainakin kerran jokaisella testauksen osa-alueella. Aloitteleva testaaja oppii myös testaamisen perusteet hyvin manuaalisella testaamisella. (Suomalainen, 2017, s.9)

Manuaalinen testaus on kuitenkin aikaa vievää ja voi viedä liian kauan. Se koetaan myös helposti liian helpoksi ja tylsäksi. Manuaalisella testauksella ei myöskään pystytä suorittamaan kaikkia testejä, esimerkiksi ohjelmiston kuormitustestausta. (Suomalainen, 2017, s. 9)

3.4 **Testauksen automatisoiminen**

Järjestelmätestaus vaatii usein sekä manuaalisia, että automatisoituja testejä. Näin testausprosessi on kattava ja oikein suoritettu. Testiautomaatio tarkoittaa erityisesti tapaa ja toimintaa, jossa sopivia työkaluja käyttämällä saadaan tietokone suorittamaan testitapaukset, ilman, että ihmisen on puututtava testaukseen. Testausautomaatiota voidaan käyttää esimerkiksi moduulien rajapinnoissa ja moduulien yksikkötestauksessa tehtävissä tarkastuksissa. (Salminen, 2020; Valagroup, n.d.)

On hyvä muistaa, että testiautomaatio on ennen kaikkea työkalu, joka oikein käytettynä voi tuoda merkittäviä hyötyjä. Se ei ole testausstrategia vaan testiautomaatioprosessin lopputulemana tulisi olla testattavalle ohjelmistolle räätälöity testiautomaatiotyökalu, joka koostuu erilaisista työkaluista. (Valagroup, n.d.)

Testauksen automatisoinnilla tavoitellaan uudelleen käytettävän koodin määrää ja laatua. Testauksen automatisoinnilla manuaalisia testejä ajavien henkilöiden määrää pystytään vähentää ja testaustiimin tehokkuutta kasvattamaan. Tuote saadaan myös nopeammin markkinoille ja testikattavuus paranee. Automatisoinnilla pystytään myös ajamaan suuri määrä testejä läpi lyhyessä ajassa. Testausympäristöjen rakentamiseen kuluva aika pienenee merkittävästi ja resurssit saadaan parempaan hyötykäyttöön. (Salminen, 2020)

Testausautomaatiota on syytä harkita, mikäli testiä toistetaan 4-20 kertaa projektin aikana. Kaikki testit olisi syytä suorittaa vähintään viisi kertaa ja 25% testeistä useammin kuin 20 kertaa. Otollista maaperää automatisoinnille on esimerkiksi yksikkötestit ja integraatiotestit. (Salminen, 2020)

Kaikkia testejä ei kuitenkaan kannata automatisoida. Automatisointi kannattaa ainoastaan niissä tapauksissa, joista on odotettavissa riittävästi lisäarvoa. (Valagroup, n.d.)

Automatisointiin sopivia testejä ovat muun muassa:
- Testit, jotka vaativat ihmiseltä paljon aikaa
- Testit, joihin sisältyy paljon datan validointia
- Testit, jotka tehdään jokaisessa regressiosyklissä
- Testit, joita ei voida tehdä muulla tapaa (esim. ei-toiminnallinen testaus, suorituskykytestaus, benchmarkkaus, stressitestaus) (Valagroup, n.d.)

Automatisointiin sopimattomia testejä ovat esimerkiksi:
- Testit, joiden vaatimukset muuttuvat usein
- Testit, jotka luokitellaan käytettävyystestaukseksi
- GUI-testaus (GUI, Graphical User Interface), mikäli GUI muuttuu usein
- Testit, joita ei ole verifioitu
- Ad-Hoc random testit (ns. ”apinatestaus”, eli täysin satunnaisuuteen perustuva automatisoitu testaus, jossa työkalu antaa satunnaisia syötteitä järjestelmälle ja tallentaa tilanteet, joissa virheitä ilmenee). (Salminen, 2020)

Testauksen automatisointiin on käytössä monia eri ohjelmia. Yksi niistä on Robot Framework. Robot Framework on testauksen automatisointiohjelma, jonka on kehittänyt Pekka Klärck. Robot Frameworkin kehys kehitettiin alun perin Nokia Networks:issa ja kesäkuussa 2008 se julkaistiin avoimen lähdekoodin ohjelmistona. Se on ytimeltään Python-ohjelmointikielellä toteutettu järjestelmän hyväksyntätestaukseen tarkoitettu sovelluskehys ja on käytössä mm. NASA:lla. Robot Framework hyödyntää avainsanapohjaisen testauksen lähestymistapaa ja se on laajennettavissa testikirjastojen avulla. Käyttäjät voivat myös kirjoittaa omia testikirjastojaan omiin tarpeisiinsa Pythonin avulla. Valmiista avainsanoista voidaan myös luoda uusia korkeamman tason avainsanoja. (Salminen, 2020; Robot Framework, n.d.)

Robot Framework-projektia isännöidään GitHubissa, josta löytyy lisädokumentaatiota, lähdekoodin ja julkaisunseurannan. Latauksia ylläpidetään PyPI:ssä. Robot Framework on käyttöjärjestelmästä ja -sovelluksesta riippumaton. Sen ydinkehys toteutetaan Pythonilla ja se toimii myös Jython:illa (JVM) ja IronPython:illa (.NET). Robot Framework julkaistaan Apache License 2.0:n alla, ja suurin osa ekosysteemin kirjastoista ja työkaluista ovat myös avoimen lähdekoodin alla. (Robot Framework, n.d.)

Robot Framework on hyvin laajennettava ja sen takia, sitä voidaan käyttää hyvin moni eri tavoin. Yleisin käyttötapa on Robot Frameworkin käyttö verkkosivustojen automatisointiin yhdessä Selenium-ohjelmiston kanssa. Robot Framework tuottaa tulosteena HTML-muotoisen raportin ja lokitiedoston, jotka sisältävät tiedot tehdyistä testeistä. (Salminen, 2020)

Koska Robot Framework on avoimen lähdekoodin ohjelmisto, sitä on helppo käyttää myös testausautomaatioon ja robotiikkaprosessien automatisointiin (RPA). Sitä käytetään myös laajalti ohjelmistokehityksessä. Se pystytään laajentamaan ja integroimaan käytännössä katsoen mihin tahansa muuhun työkaluun, tehokkaiden ja joustavien automaatiojärjestelmien luomiseksi. Avoimen lähdekoodin käyttäminen tarkoittaa myös sitä, että Robot Framework:in käyttö on ilmaista, ilman lisenssimaksuja. (Robot Framework, n.d.)

4 TESTAUSSUUNNITELMAN LUOMINEN

Testaussuunnitelman (kuva 2) rakentaminen ja testien valmistelu tulee aina aloittaa aikaisessa vaiheessa projektia. Testaussuunnitelmaan vaikuttavia tekijöitä ovat organisaation testauskäytänteet, tavoitteet, riskit ja projektin rajoitteet. On myös hyvä muistaa mitä vaatimuksia suunnitelmalle on ja mihin tarkoitukseen se tarvitaan. On myös pidettävä mielessä, kenelle suunnitelma tehdään ja miten tehdä suunnitelma niin, että se on ymmärrettävä. Kun testaussuunnitelmaa aletaan laatimaan, tarvitaan tätä varten määrittelydokumentti, kokemustarkastuslistat, suunnitteludokumentti, laatujärjestelmän ohjeistus ja vanha testaussuunnitelma. (Salminen, 2020; Katara, 2015)

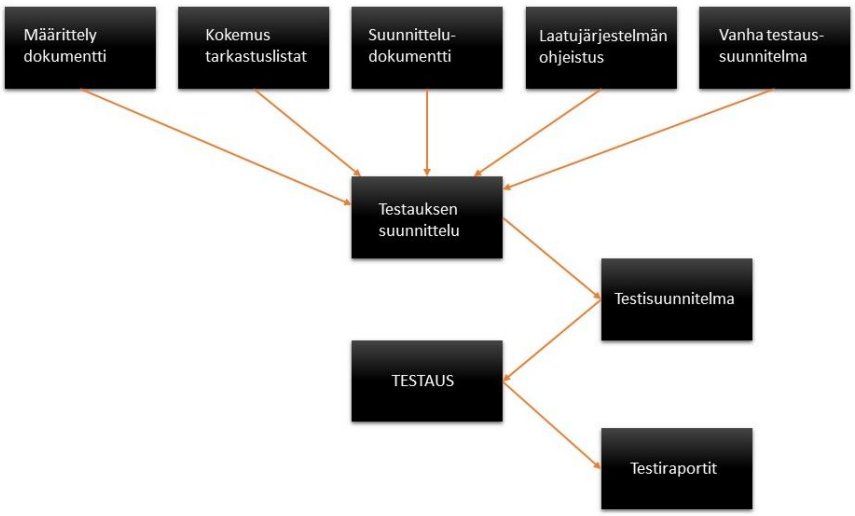
Yllä mainittujen dokumenttien avulla laaditaan uusi testaussuunnitelma, jonka avulla sovellus/ohjelmisto lopulta testataan. Tehdyistä testeistä laaditaan lopuksi testiraportit. Testaussuunnitelma pitää kuitenkin suunnitella projektin koon, riskitason ja testattavan ohjelmiston perusteella. Pienelle projektille riittää yleensä yksi yleinen testaussuunnitelma, joka kattaa integrointi- ja järjestelmätestauksen. Suuremmille projekteille on hyvä tehdä päätestaussuunnitelma, joka kuvaa testauksen kokonaisuudessaan, eri testaustasot ja jokaiselle testityypille laaditaan oma suunnitelma. (Salminen, 2020; Katara, 2015)

Testaussuunnitelmaan on tärkeää myös määritellä testauksen lopettamiskriteerit. Mahdollisia lopettamiskriteerejä ovat muun muassa varatun ajan päättyminen. Testaaminen on syytä myös lopettaa, kun kaikki testitapaukset on tehty ja analysoitu tai kaikki mahdolliset via on löydetty ja korjattu. (Salminen, 2020)

Testaussuunnitelman tulisi sisällyttää ainakin seuraavat aiheet:

- Johdanto
- Testauksen kohde ja tavoitteet
- Testausympäristö
- Testauksen organisointi ja raportointi
- Testausstrategia ja integrointisuunnitelma
- Toimintojen testitapaukset, hyväksymiskriteerit
- Ei-toiminnallisten ominaisuuksien testaaminen
- Erikoistilanteet
- Ominaisuudet, joita ei testata. (Salminen, 2020)

Testaussuunnitelma on loppujen lopuksi dokumentti, jossa linjataan mitä ohjelmasta testataan, missä vaiheessa testataan, millä menetelmällä testataan ja millaisia lopputuloksia testistä odotetaan. (Salminen, 2020)



Kuva 2. Testaussuunnitelma (Salminen, 2020).

Testaussuunnitelma voidaan siis määritellä olevan dokumentti, joka linjaa mitä ohjelmistosta testataan, missä vaiheessa mikäkin osa testataan ja millä menetelmällä testaus suoritetaan. Lopuksi vielä määritellään, millaisia lopputuloksia testauksesta odotetaan. (Salminen, 2020)

Ohjelmiston täydellisesti kattava testaaminen on kuitenkin mahdotonta, joten kaikista mahdollisista testitapauksista muodostetaan testijoukko, eli rajattu kokonaisuus. Testijoukko muodostuu testitapauksista, joiden tehtävänä on varmistaa ohjelmiston täyttävän sille asetetut toiminnan odotukset. (Pollari, 2014)

Jokaisella testitapauksella tulee olla ID, eli testitapauksen yksilöivä tunnus. Testitapauksen kuvauksesta on tultava ilmi testi-ID ja testin tarkoitus, joka antaa testaajalle perspektiivin, jonka perusteella tutkia testin tuloksia. Lisäksi testitapauksella tulee olla ennakkoehdot, eli alkutilanne. Kun testitapaus suoritetaan, ennakkoehdot alustetaan joka kerta testiä suorittaessa. Testitapauksella tulee näiden lisäksi olla test step:it, joiden mukaan testi suoritetaan. Jokaisella test step:illä on myös omat odotetut tuloksensa. Testitapauksen kuvauksessa tulee myös määrittää läpäisykriteerit. (Pollari, 2014)

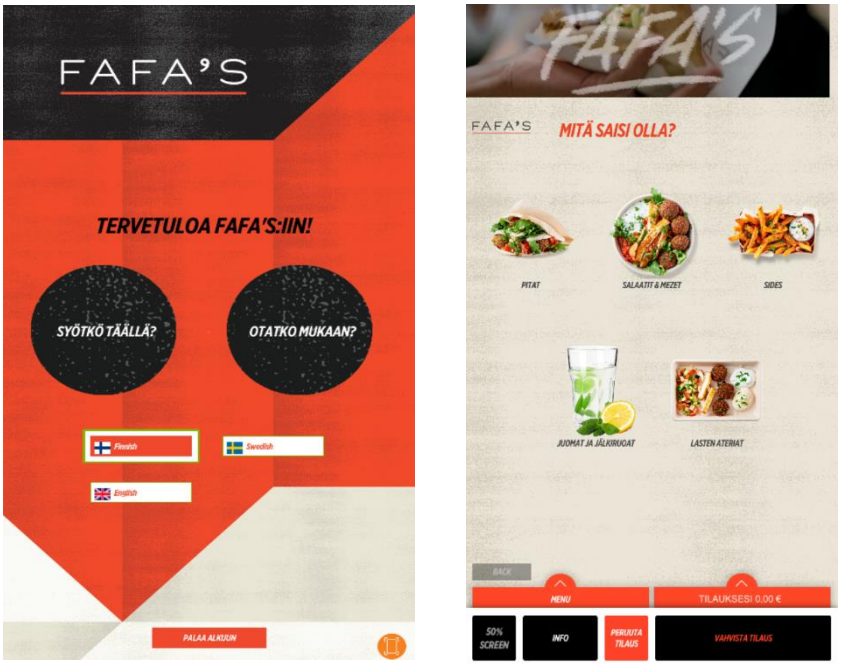
Testitapauksen tulos voi olla joko pass tai fail, eli hyväksytty tai hylätty. Se määräytyy vertaamalla testauksen tapahtumia testitapauksen odotettuihin tuloksiin. Samalla tarkastetaan, täyttääkö ohjelmiston toiminta sille testitapauksessa määrätyt läpäisykriteerit. (Pollari, 2014)

Uusi testitapaus laaditaan aina ohjelmiston käyttötapauksen testaamista varten. Testitapauksen sisältö määräytyy siis pitkälti ohjelmiston käyttötapausten mukaan. Ensin tulee tutustua ohjelmiston käyttötapauksiin, joiden toimivuus testitapauksella tullaan testaamaan. Testitapaus onkin hyvä nimetä siten, että se kuvaa testitapauksessa testattavia toimintoja. (Pollari, 2014)

Itse testijoukko koostuu useista testitapauksista, jotka testattavalla ohjelmistolle suoritetaan. Suurissa projekteissa, testijoukot toimivat helposti myös työnjakajina, jolloin kunkin testijoukon tehtävänä voi olla esimerkiksi tietyn toiminnon varmistaminen. Testijoukon koko riippuu siitä, kuinka luotettavaan toimintaan ohjelmiston kanssa pyritään. Testijoukon tarkoitus vaikuttaa myös sen kokoon. Kun testijoukkoa suunnitellaan, pyritään siitä tekemään mahdollisimman pieni, mutta kuitenkin tarpeeksi kattava, jotta kaikki oleellinen toiminta tulee testatuksi. Testijoukkoon on hyvä sisällyttää kaikki aiemmilla testikerroilla läpäisemättömät testitapaukset. Toinen tärkeä testitapausryhmä testijoukoissa on ydintoiminnan varmistavat testitapaukset. Aivan viimeisenä testijoukkoon lisätään kaikki loput tarpeelliset testitapaukset, jotka käsittelevät samaa aihealuetta. (Pollari, 2014)

5 TESTAUSSUUNNITELMAN SUUNNITTELU

Testaussuunnitelman pohjana käytetään Acrelecin luomaa Fafa's-ravintolan kioskitilauskehystä (kuva 3). Acrelecin edustajien pyynnöstä, testaussuunnitelma pyritään tekemään asiakkaan näkökulmasta, jotta asiakkaan kannalta keskeisimmät toiminnot tilausjärjestelmästä tulee testattua aina samalla tavalla.



Kuva 3. Fafa's kioskitilausjärjestelmän etusivu (Fafa's DOT)

Ennen testaussuunnitelman luontia, on Fafa's kioskitilausjärjestelmä käyty yhdessä Acrelecin edustajien kanssa läpi ja heidän kanssaan päätetty järjestelmän keskeisimmät ja kriittisimmät toiminnot. Tämän pohjalta on yhdessä päätetty testaussuunnitelmaan tulevat toiminnot.

Testaussuunnitelmassa nämä testattavat toiminnot tulevat olemaan muun muassa:

- Onko kaikilla tuotteilla kuvat?
- Onko kuvat ja tekstit asemoitu oikein (Ovatko kuvat ja tekstit suunnilleen samoilla etäisyyksillä toisiinsa nähden eivätkä tekstit mene kuvien päälle?)?
- Toimiiko tilauksen muokkaussivu?
- Toimiiko lisämyynti-sivu?
- Toimivatko kaikki painikkeet?
- Pystyykö ostoskoria muokkaamaan?
- Toimiiko "Vahvista tilaus"-sivu?
- Toimiiko pyörätuolitoiminto?
- Miten ohjelmisto toimii eri kieliversioilla?
- Onnistuuko tuotteiden tilaaminen eri kombinaatioilla?
- Toimiiko ohjelmisto oikein, kun sillä tehdään euro- ja kappalemääräisesti suuri tilaus?

Tässä testaussuunnitelmassa testaaminen tulee olemaan manuaalista käsin tehtyä testaamista. Yrityksen toive on, että testaaminen tehdään niin sanotusti kuluttajan näkökulmasta, joten testaaminen tulee tämän takia tehdä jäljitellen asiakkaan mahdollista ostokäyttäytymistä. Tällä tavalla asiakkaan näkökulmasta oleellisimmat vaiheet tulee testattua.

Testaussuunnitelmassa testaaminen tulee olemaan pitkälti toiminnallista- ja käytettävyystestausta. Testaussuunnitelmalla tullaan varmistamaan, että ohjelmisto toimii aina toivotulla tavalla ja testaus suoritetaan loppukäyttäjän, eli asiakkaan, näkökulmasta. Lisäksi testauksessa varmistetaan, että ohjelmisto toimii, on helppokäyttöinen, intuitiivinen, looginen ja, että käyttäjä ei joudu odottamaan toiminnallisuuksia ja latauksia liian pitkään.

Testaussuunnitelmassa tutkitaan myös samalla, että mitkä vaiheet voidaan automatisoida käyttämällä esimerkiksi Robot Frameworkia. Mahdollisia automatisoitavia vaiheita voisivat olla esimerkiksi kuvien ja tekstien asemoinnin tarkistaminen. Tähän voidaan käyttää Robot Framework:ia.

5.1 **Testausympäristö**

Testausympäristönä toimii Fafa’s dot-sovellus. Kyseessä on siis Acrelecin luoma Direct Order Taker-sovellus, johon Fafa’s-ravintolan kioskitilauskehys on luotu. Testaaminen tapahtuu suoraan tässä ympäristössä ja testaaminen tapahtuu simuloiden kuluttajan käyttäytymistä tilaustilanteessa. Testaus suoritetaan ensisijaisesti Windows-käyttöjärjestelmässä.

5.2 **Testauksen organisointi ja raportointi**

Testitapausten tuloksista raportoidaan testausdokumentteihin, jotka löytyvät tämän opinnäytetyön liitteinä. Testeistä kerrotaan testaajan nimi, suorituspäivämäärä, testattava asia, kuvaus testistä, odotettu tulos sekä havaitut virheet. Löydetyt virheet pyritään korjaamaan mahdollisuuksien mukaan. Mikäli jokin virhe havaitaan vasta testauksen loppuvaiheessa ja sen korjaaminen olisi mahdotonta käytettävissä olevan ajan puitteissa, virhe jätetään korjaamatta ja dokumentoidaan liitteistä löytyvään virheraporttiin.

5.3 **Testausstrategia ja integrointisuunnitelma**

Testaaminen tulee olemaan toiminnallista- ja käytettävyystestausta. Testaus toteutetaan loppukäyttäjän, eli asiakkaan näkökulmasta, ja suoritetaan käyttöliittymää käyttäen. Toiminnallisella testauksella varmistetaan, että ohjelmisto toimii toivotulla tavalla toiminnallisuutensa puolesta.

Käytettävyystestaamisella testataan, toimiiko ohjelmisto hyvin ja onko se helppokäyttöinen. Sillä myös tutkitaan, onko värit, kontrastit ja siirtymät hyväksyttyjä, ja joutuuko asiakas esimerkiksi odottamaan toiminnallisuuksia liian pitkään. Testaamisen aikana testaaja suorittaa järjestelmätestit testaussuunnitelmassa määritetyssä järjestyksessä.

5.4 **Ominaisuudet, joita ei testata**

Järjestelmän tietoturvaa ja kuormituksen vaikutuksia ei erikseen testata. Ohjelmiston ohjelmointipuolta ei myöskään erikseen testata, sillä testaaminen keskittyy vain ohjelmiston toimintojen testaamiseen.

5.5 **Testien automatisoiminen**

Jokainen testitapaus on mahdollista myös automatisoida käyttämällä Robot Frameworkia. Mikäli halutaan automatisoida vain jotkin tietyt vaiheet, voidaan automatisointia käyttää esimerkiksi kuvien ja tekstien asemoinnin tarkistamiseen. Robot Framework voi tarkistaa, ovatko kuvat esimerkiksi samalla etäisyydellä sivun alareunasta. Testejä ei kuitenkaan lähdetä automatisoimaan vielä tässä vaiheessa. Pääpaino on testaussuunnitelman luomisessa, sillä ajallisesti resurssit eivät riitä testien automatisointiin.

6 TESTAUSSUUNNITELMA

Seuraavassa kuvataan ohjelmistolle suunnitellut ja suoritettavat testitapaukset. Testien tuloksista raportoidaan liitteistä löytyviin Test Case Template:ihin, jotka on laadittu testitapauksille valmiiksi. Testitapaukset ovat ohjelmiston toimintoja, jotka testataan asiakkaan näkökulmasta.

6.1 TESTITAPAUS 1: Aloitusnäyttö

Kuva 4. Fafa's tilausjärjestelmän aloitusnäyttö (Fafa's DOT)

Avataan tilausjärjestelmän aloitusnäyttö (Kuva 4.), josta valitaan, syödäänkö ravintolassa vai otetaanko tilaus mukaan.

1. Tarkistetaan ”Syötkö täällä?” ja ”Otatko mukaan?” -painikkeiden toimivuus
2. Tarkistetaan kielivalintapainikkeiden toimivuus
3. Klikataan näytön puolittavaa, eli ns. Pyörätuoli-painiketta, ja tarkistetaan, että se toimii

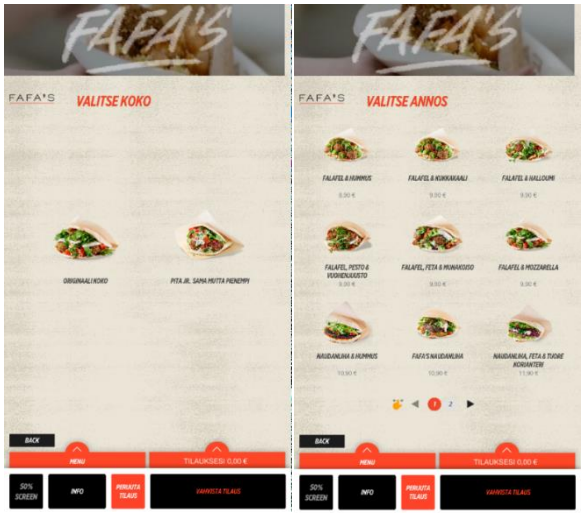
6.2 **TESTITAPAUS 2: Tilausnäkymä**

Kuva 5. Tilausnäkymä (Fafa's DOT)

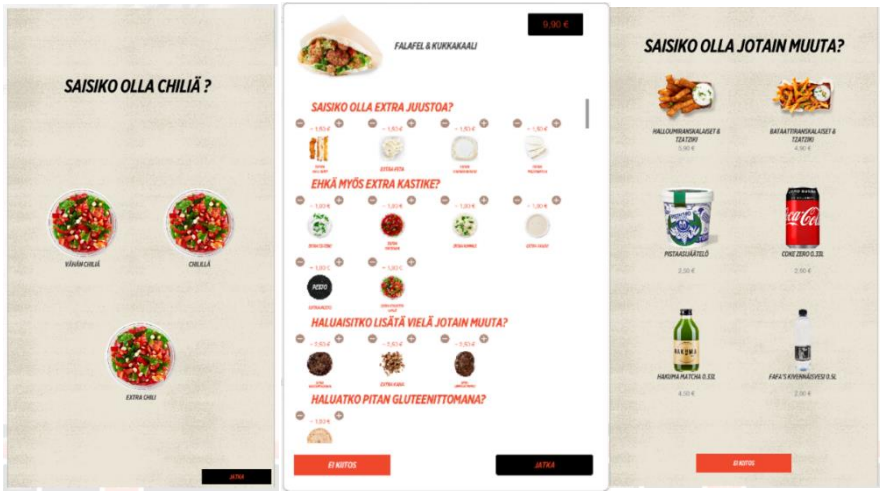
Valitaan aloitusnäkymästä (Kuva 4.) ”Syötkö täällä?”-painike ja siirrytään tilausnäkymään (Kuva 5).

1. Tarkistetaan silmämääräisesti, että tekstit ja kuvat eivät ole päällekkäin, kuvat ovat selkeitä, teksteistä saa selvää ja tekstit ovat suunnilleen samoilla linjoilla toisiinsa nähden
2. Tarkistetaan, että ”50% screen” -painike toimii ja kuvasuhde saadaan muuttumaan
3. Painetaan ”Info”-painiketta, jotta saadaan allergeeniluettelo auki
4. ”Peruuta tilaus”-painikkeesta päästään takaisin aloitusnäytölle

6.3 TESTITAPAUS 3: Ruoan tilaaminen



Kuva 6. Koon ja annoksen valinta (Fafa's DOT)

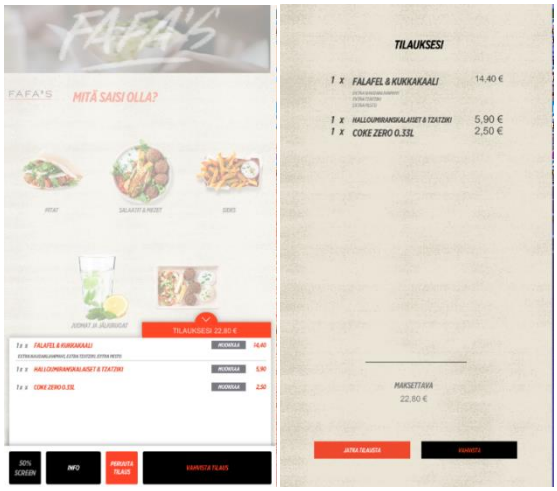


Kuva 7. Lisävalinnat tilaukselle (Fafa's DOT)

Tehdään testitilaus järjestelmässä ja tarkistetaan samalla, että kaikki tarvittavat toiminnot toimivat.

1. Valitaan tilausnäkymästä ”Pita” (Kuva 5). Tämän jälkeen järjestelmän tulee kysyä pitan kokoa. Valitaan ”Original koko” (Kuva 6)
2. ”Valitse annos”-näkymästä valitaan annos (Kuva 6)
3. ”Saisiko olla chiliä?”-valintaikkuna aukeaa (Kuva 7). Valitaan jokin vaihtoehto tai painetaan ”Jatka”
4. Lisävalintaikkuna aterialle aukeaa (Kuva 7). Tarkistetaan, että tekstit ja kuvat eivät ole päällekkäin ja että valintapainikkeet ovat tarpeeksi selkeät. Lisävalintojen jälkeen eteenpäin pääsee painamalla ”Jatka”-painiketta. Vaihtoehtoisesti painetaan ”Ei kiitos”-painiketta.
5. Lisämyynti-ikkuna avautuu tämän jälkeen (Kuva 7). Lisämyynti-ikkuna ponnahtaa käyttäjälle näkyviin niin monta kertaa tuotteen valinnan jälkeen, kunnes käyttäjä painaa ”Ei kiitos”-painiketta

6.4 **TESTITAPAUS 4: Tilauksen viimeistely**



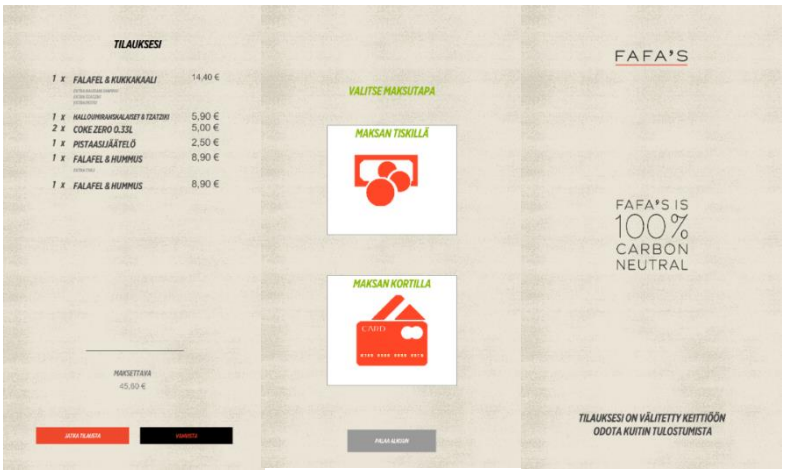
Kuva 8. ”Tilauksesi”-painike ja tilauksen yhteenveto (Fafa’s D)

Lisämyynti-ikkunan jälkeen järjestelmä palaa etusivulle. Tilaukseen voi lisätä nyt lisää tuotteita, tarkastella tilaustaan, peruuttaa tilauksensa tai siirtyä tilaamaan ja maksamaan.

1. Tarkistetaan ”Tilauksesi”-painikkeen toimivuus. Yhteenveto tehdystä tilauksesta tulisi tällöin painikkeen painamisen jälkeen näkyä näytöllä (Kuva 8).
2. Tarkistetaan ”Vahvista tilaus”-painikkeen toimivuus. Painike vie kassalle ja tilauksen tulee olla eriteltynä selkeästi ja kaikki tehdyt lisävalinnat näkyvät (Kuva 8).
3. Mikäli halutaan tehdä lisää ostoksia, klikataan vahvistussivulta ”Jatka tilausta”-painiketta.

6.5 TESTITAPAUS 5: Tilauksen vahvistaminen ja maksaminen

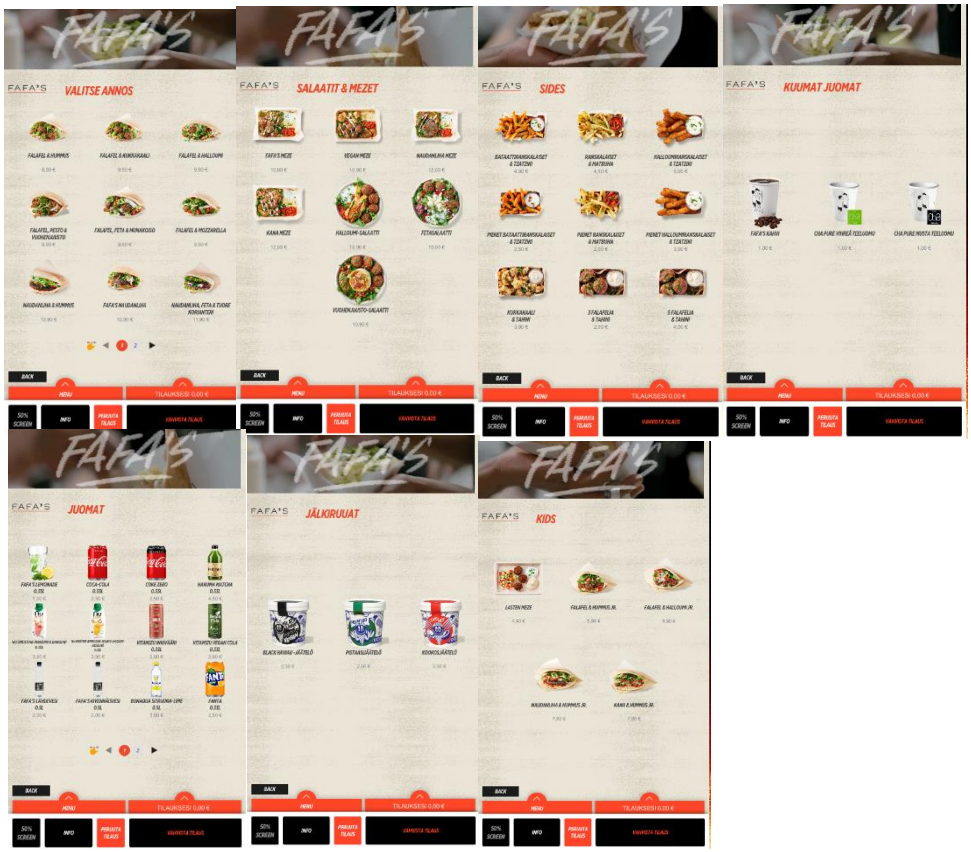
Kuva 9. Tilauksen yhteenveto, maksutavan valinta ja tilauksen vahvistaminen (Fafa’s DOT)



Kun tilaus on tehty kokonaisuudessaan, vahvistetaan tilaus. Tilausvahvistuksessa tulee tuotteet olla eriteltynä selkeästi toisistaan (Kuva 9). Mikäli tilauksessa on esimerkiksi kaksi samaa tuotetta, mutta toisessa on vaikka extra chiliä, on nämä tuotteet eritelty toisistaan tilausvahvistuksessa.

1. Klikataan ”Vahvista” tilausvahvistuksesta
2. Valitaan maksutapa (Kuva 9)
3. Maksutavan valinnan jälkeen, asiakkaan tulee saada kuittaus tilauksen vastaanottamisesta (Kuva 9)

6.6 TESTITAPAUS 6: Tuotteiden hintojen silmämääräinen tarkistus

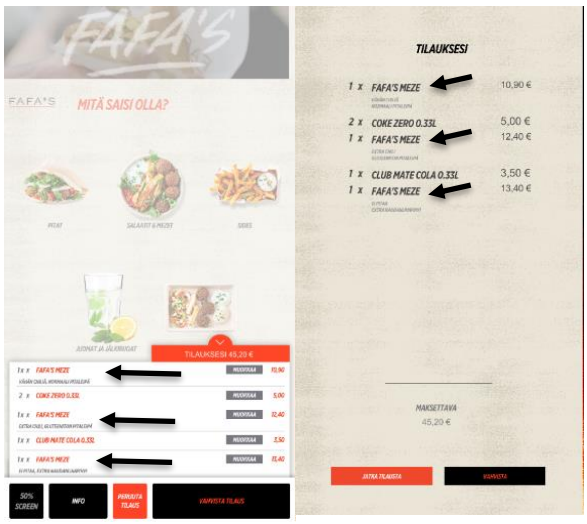


Kuva 10. Tuotteet kategorioittain ja niiden hinnat (Fafa's DOT)

Käydään läpi jokainen tuoteryhmä (Kuva 10) ja tarkistetaan silmämääräisesti, että jokaisella tuotteella on jokin hinta ja, että hinta näyttää oikealta ja tuotteeseen sopivalta.

1. Klikataan etusivulta ”Pita” ja ”Originaali koko”. Valikoima avautuu. Tarkistetaan hinnat.
2. Klikataan etusivulta ”Salaatit & mezet”. Kun valikoima avautuu, tarkistetaan, että hinnat ovat oikein.
3. Klikataan etusivulta ”Sides”. Tarkistetaan hinnat, kun valikoima avautuu.
4. Klikataan etusivulta ”Juomat ja jälkiruoat”. Käydään läpi jokainen tuotekategoria ja tarkistetaan hinnat kategorioittain:
    a. Juomat
    b. Kuumat juomat
    c. Jälkiruoat
5. Klikataan etusivulta ”Lasten ateriat” ja tarkistetaan hinnat.

6.7 TESTITAPAUS 7: Tuotteiden tilaaminen eri kombinaatioilla

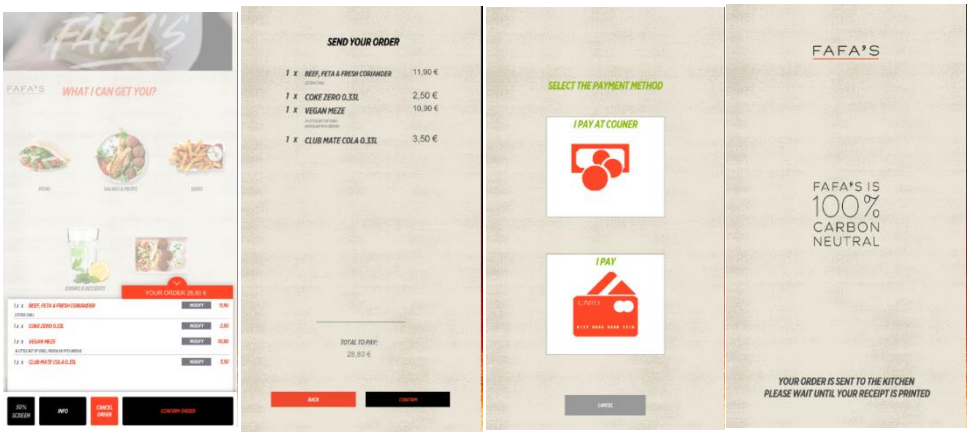


Kuva 11. Omavalintainen sattumanvarainen tilaus, ostoskori ja tilauksen yhteenveto (Fafa's DOT)

Tehdään omavalintainen tilaus, jossa on monta samaa ateriaa eri kombinaatioin (esim. kaksi samanlaista ateriaa, mutta toiseen vähän chiliä, ja toiseen ei chiliä olenkaan). Tarkistetaan tämän jälkeen, että ateriat jaottuvat ostoskoriin oikein ja tulevat selkeästi esiin myös tilauksen yhteenvedossa (Kuva 11).

1. Klikataan etusivulta ”Syötkö täällä?”.
2. Tehdään omavalintainen tilaus, jossa on sama ateria tilattuna moneen kertaan, mutta eri lisävalinnoin.
3. Tarkistetaan ostoskorin toimivuus ja että ateriat on jaoteltu sinne oikein.
4. Klikataan ”Vahvista tilaus” ja tarkistetaan tilauksen yhteenveto, jotta se vastaa ostoskoria.

6.8 TESTITAPAUS 8: Tilaaminen englanniksi ja ruotsiksi

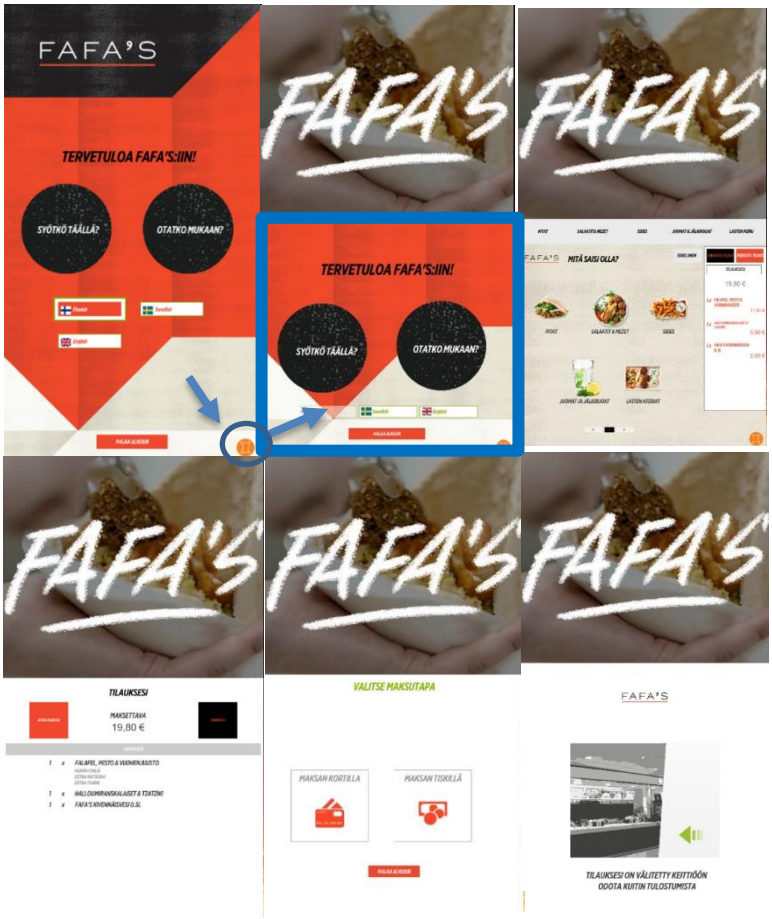


Kuva 12. Englanninkielinen tilaus (Fafa's DOT)

Tehdään omavalintainen englannin- tai ruotsinkielinen tilaus ja tarkistetaan samalla, että kieli pysyy koko ajan valittuna.

1. Valitaan etusivulta kieleksi ”English” tai ”Swedish” (Kuvassa 12 valittu englanti.)
2. Tehdään omavalintainen tilaus alusta loppuun.
3. Tarkistetaan koko ajan silmämääräisesti, että kieli pysyy valittuna kielenä.

6.9 TESTITAPAUS 9: Tilaaminen pyörätuolitoiminnolla

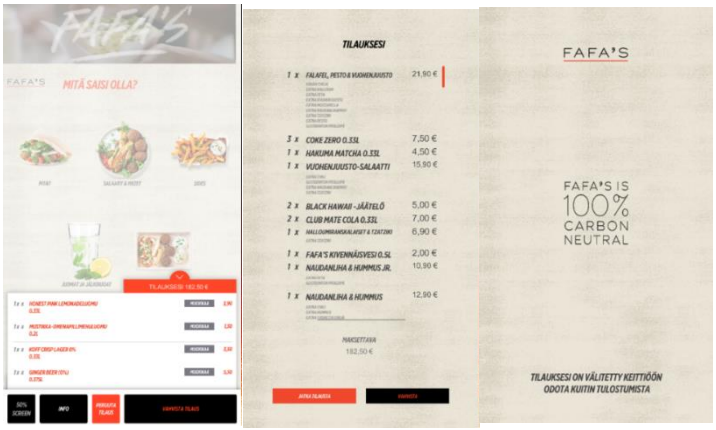


Kuva 13. Tilausvaiheet alusta loppuun pyörätuolitoiminnolla (Fafa's DOT)

Valitaan etusivulta pyörätuolitoiminto, jolloin näytön kuvasuhde muuttuu. Toiminnon avulla mm. pyörätuolia käyttävät ihmiset, pystyvät tilaamaan ateriansa helposti (Kuva 13).

1. Klikataan etusivun oikealla alakulmassa olevaa pyörätuolipainiketta (Kuva 13).
2. Sivuston kuvasuhde muuttuu. Valitse ”Syötkö täällä?”
3. Tehdään omavalintainen tilaus ja tarkistetaan, että ohjelmisto pysyy koko ajan pyörätuolitoiminnossa.
4. Vahvistetaan tilaus ja tarkistetaan toiminto pyörätuolitilassa.
5. Valitaan maksutapa ja tarkistetaan, että tilausvahvistus tulee näytölle myös pyörätuolitilassa.

6.10 **TESTITAPAUS 10: Euro- ja kappalemääräisesti suuret tilaukset**



Kuva 14. Euro- ja kappalemääräisesti suuri tilaus (Fafa's DOT)

Testataan järjestelmää tekemällä normaalia suurempi tilaus, johon valitaan mahdollisimman paljon tuotteita (Kuva 14).

1. Tehdään omavalintainen, mahdollisimman suuri tilaus, jossa on myös paljon samoja tuotteita ja aterioita pienillä eroilla (esim. samoja aterioita, mutta valitaan niihin eri määrä chiliä).
2. Tehdään tilauksesta kappalemääräisesti mahdollisimman suuri ja tarkistetaan, että ohjelmisto pysyy euromääräisesti mukana tilauksessa.
3. Vahvistetaan tilaus ja tarkistetaan yhteenvedosta, että ateriat on jaoteltu oikein.
4. Vahvistetaan tilauksen yhteenveto.

7 YHTEENVETO JA JATKOKEHITYS

Acrelec Finland Oy:ltä puuttui selkeä ja johdonmukainen testaussuunnitelma DOT-kioskitilauskehykselleen (DOT, Digital Order Taker). Yritys toivoi, että testaukseen saataisiin johdonmukainen, step-by-step testaussuunnitelma, jotta sovellus tulisi jatkossa aina testattua samalla tavalla, kun ohjelmistoon on esimerkiksi tehty päivityksiä. Pohjana testaussuunnitelmalle käytettiin Fafa's DOT-sovellusta, eli Fafa's-ravintolalle luotua kioskitilauskehystä. Testaaminen oli pitkälti toiminnallista- ja käytettävyystestaamista, sillä testaus tehtiin loppukäyttäjän, asiakkaan, näkökulmasta. Testaus myös suoritettiin Fafa's DOT-käyttöliittymää käyttäen.

Fafa's DOT-sovelluksen pohjalta luotiin kymmenen erilaista testitapausta, joissa käytiin vaihe vaiheelta läpi testattavat asiat ja määriteltiin mikä testauksen lopputuloksen tulisi olla. Testitapauksille luotiin myös erilliset Test Case Template:it, jotka löytyvät opinnäytetyön liitteistä. Test Case Template:eihin määriteltiin tarkasti, mikä minkäkin testiaskeleen läpäisykriteeri on. Näitä yritys voi käyttää jatkossakin testaamisessa.

Testaussuunnitelmassa oli myös tarkoitus tutkia, voidaanko jotkin testitapaukset mahdollisesti automatisoida. Loppujen lopuksi, jokainen testitapaus on mahdollista tehdä myös automaattisesti käyttämällä esimerkiksi Robot Framework:ia. Tähän opinnäytetyöhön ei automatisointia kuitenkaan otettu mukaan, koska se ei ollut ajallisesti mahdollista. Automatisointia voitaisiin myös käyttää esimerkiksi sovelluksessa käytettyjen kuvien testaamiseen ja siihen, ovatko kuvat ja tekstit samalla etäisyydellä toisistaan tai sivuston alareunasta. Koska testitapausten automatisointi ei opinnäytetyöhön ajallisesti mahtunut mukaan, tulisi automatisoinnista tehdä kokonaan oma tutkimuksensa ja suunnitelmansa.

Testaussuunnitelmaa on myös mahdollista laajentaa maksamiseen liittyvillä testeillä ja niiden mahdollisella automatisoinnilla. Näitä mahdollisia testitapauksia on esimerkiksi korttimaksun tekeminen ja kuitin tulostus. Tilauksen läpiviennin kannalta on tärkeää, että myös maksaminen onnistuu ongelmitta ja ohjelmisto olisi tärkeää testata myös erilaisissa maksamiseen liittyvissä tilanteissa. Olisi myös tärkeää testata asiakkaan tilauksen siirtyminen eteenpäin keittiölle ja muut niin sanotut POS (Point of Sale) integraatiot.

Testaussuunnitelmaa on mahdollista myös jatkokehittää luomalla sen esimerkiksi täysin sähköiseen muotoon. Testaussuunnitelman voisi koodata HTML-kielellä ja tehdä testaussuunnitelmasta oman sivustonsa. Testaussivuston kautta olisi myös mahdollista raportoida mahdollisista virheistä, joita testaamisen aikana ilmenisi. Testaussivuston kautta testaaminen ja raportointi voisi olla mielekkäämpää kuin Test Case Template:ien käyttö.

LÄHTEET

Acrelec. (n.d.). About. Haettu 8.7.2020 osoitteesta https://acrelec.com/about/

Acrelec Finland. (n.d.). Haettu 15.6.2020 osoitteesta https://acrelec.com/acrelec-finland/

Acrelec Kiosk. (n.d.). Haettu 8.7.2020 osoitteesta https://acrelec.com/kiosk/

Katara, M. (2011). Ohjelmistojen testaus, hyväksymistestaus. Tampereen Teknillinen Yliopisto. Haettu 29.7.2020 osoitteesta http://www.cs.tut.fi/~tie21201/s2011/luennot/OHJ-3060_2011_110-170.pdf

Katara, M., Vuori M., Jääskeläinen A. (2015) Ohjelmistojen testaus. Tampereen Teknillinen Yliopisto. Haettu 14.8.2020 osoitteesta http://www.cs.tut.fi/~tie21201/s2015/luennot/TIE-21204_2015.pdf

Kudjoi, A. (2018). *Käyttöliittymätestien kehittäminen Robot Framework-sovelluskehyksellä.* Opinnäytetyö. Tietojenkäsittelyn tradenomi. Laurea-ammattikorkeakoulu. Haettu 6.7.2020 osoitteesta https://www.theseus.fi/bitstream/handle/10024/157485/Kudjoi_Aleksi.pdf?sequence=1

Lyytinen, P. (2019). Mitä ohjelmistotestaus on ja mihin se pystyy? Blogijulkaisu 31.10.2019. Haettu 4.8.2020 osoitteesta https://www.valagroup.com/fi/2019/10/mita-ohjelmistotestaus-on-ja-mihin-se-pystyy/

Pollari, J. (2014). *Ohjelmistotestaus.* Opinnäytetyö. Tietotekniikan koulutusohjelma. Centria Ammattikorkeakoulu. Haettu 6.8.2020 osoitteesta https://www.theseus.fi/bitstream/handle/10024/85400/Pollari_Jukka.pdf?sequence=1&isAllowed=y

Robot Framework. (n.d.). Haettu 8.7.2020 osoitteesta https://robotframework.org

Salminen, L. (2020). Testausprosessit-moduulin verkkoaineisto, Luento 1, Moodle. Hämeen ammattikorkeakoulu. Haettu 15.6.2020 osoitteesta https://learn.hamk.fi

Salminen, L. (2020). Testausprosessit-moduulin verkkoaineisto, Testaussuunnitelma, Moodle. Hämeen ammattikorkeakoulu. Haettu 16.6.2020 osoitteesta https://learn.hamk.fi

Salminen, L. (2020). Testausprosessit-moduulin verkkoaineistu, Automatisoitu testaus ja Selenium Web Driver, Moodle. Hämeen ammattikorkeakoulu. Haettu 23.6.2020 osoitteesta https://learn.hamk.fi

Salminen, L. (2020). Testausprosessit-moduulin verkkoaineisto, Robot Framework, Moodle. Hämeen ammattikorkeakoulu. Haettu 23.6.2020 osoitteesta https://learn.hamk.fi

Smart education. (n.d.). Testauksen tasot. Haettu 29.7.2020 osoiteesta http://smar-teducation.jyu.fi/projektit/systech/Periaatteet/suunnittelun-periaatteet/testaus/tes-tauksen-tasot

Suomalainen, R. (2017) *Regressiotestauksen suunnittelu ja toteutus.* Opinnäytetyö. Tie-tojenkäsittelyn koulutusohjelma. Haaga-Helia Ammattikorkekaoulu. Haettu 17.8.2020 osoitteesta https://core.ac.uk/download/pdf/161417573.pdf

Testimate. (n.d.). Hyväksymistestaus. Haettu 29.7.2020 osoitteesta https://www.testi-mate.fi/palvelut/testaus-ja-laadunvarmistus/hyvaksymistestaus/

Valagroup. (n.d.). Ohjelmistotestaus ja laadunvarmistus. Haettu 10.8.2020 osoitteesta https://www.valagroup.com/fi/palvelut/ohjelmistotestaus-ja-laadunvarmistus/

Valagroup. (n.d.) Testiautomaatio-opas. Haettu 10.8.2020 osoitteesta https://www.va-lagroup.com/wp-content/uploads/pdf/Testiautomaatio-opas-2020.pdf?gclid=CjwKCAjw4MP5BRBtEiwASfwAL-s9zhzFgA8Fpsg4PeIocVYtstltXEn1vYdmNw4omSrzM5m_8aYLaxoCIA0QAvD_BwE

LIITE 1

TESTITAPAUKSET

Project Name: Fafa's kioskiohjelmiston testaussuunnitelma

Test Case Template

Test Case ID: 1
Test Designed by: Milka Ukkonen
Test Priority (Low/Medium/High): High
Test Designed date: 31.7.2020
Module Name:
Test Executed by:
Test Title: Aloitusnäyttö
Test Execution date:
Description: Avataan tilausjärjestelmän aloitusnäyttö, josta valitaan, syödäänkö ravintolassa vai otetaanko tilaus mukaan. Testataan myös pyörätuolitoiminto.

Pre-conditions:
Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Klikkaa ”Syötkö täällä?”		Järjestelmä siirtyy tilaussivulle			
2	Klikkaa ”Otatko mukaan?”		Järjestelmä siirtyy tilaussivulle			
3	Klikkaa näytön puolittavaa painiketta		Näyttö puolittuu ja siirtyy alemmas			
4						

Post-conditions:

Project Name: Fafa's kioskiohjelmiston testaussuunnitelma

Test Case Template

Test Case ID: 2 **Test Designed by: Milka Ukkonen**

Test Priority (Low/Medium/High): High **Test Designed date: 31.7.2020**

Module Name: **Test Executed by:**

Test Title: Tilausnäkymä **Test Execution date:**

Description: Valitaan aloitusnäkymästä ”Syötkö täällä?”-painike ja siirrytään tilausnäkymään ja testataan sen toimivuus.

Pre-conditions:
Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Tarkista silmämääräisesti, että tekstit ja kuvat eivät ole päällekkäin					
2	Tarkista, että tekstit ja kuvat ovat silmämääräisesti samoilla linjoilla					
3	Klikkaa ”50% screen”-painiketta		Näyttö puolittuu ja siirtyy alemmas			
4	Klikkaa ”Info”-painiketta		Allergeeniluettelo avautuu			
5	Klikkaa ”Peruuta tilaus”		Palataan aloitussivulle			

Post-conditions:

Project Name: Fafa's kioskiohjelmiston testaussuunnitelma

Test Case Template

Test Case ID: 3 **Test Designed by: Milka Ukkonen**

Test Priority (Low/Medium/High): High **Test Designed date: 31.7.2020**

Module Name: **Test Executed by:**

Test Title: Ruoan tilaaminen **Test Execution date:**

Description: Tehdään testitilaus ja tarkistetaan, että kaikki oleelliset toiminnot toimivat

Pre-conditions:
Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Valitaan tilausnäkymästä ”Pita”		Järjestelmä kysyy pitan kokoa			
2	Valitaan ”original koko”		”Valitse annos” näkymä avautuu			
3	Valitse jokin annos		”Saisiko olla chiliä?” näkymä avautuu			
4	Valitse jokin vaihtoehto tai paina ”jatka”		Lisävalintaikkuna aukeaa			

Post-conditions:

Project Name: Fafa's kioskiohjelmiston testaussuunnitelma

Test Case Template

Test Case ID: 4
Test Designed by: Milka Ukkonen
Test Priority (Low/Medium/High): High
Test Designed date: 31.7.2020
Module Name:
Test Executed by:
Test Title: Tilauksen viimeistely
Test Execution date:
Description: Lisämyynti-ikkunan jälkeen, järjestelmä palaa etusivulle. Tilaukseen voi nyt lisätä tuotteita, tarkastella tilaustaan, peruuttaa tilauksensa tai siirtyä tilaamaan ja maksamaan.

Pre-conditions:
Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Paina ”Tilauksesi”-painiketta		Yhteenveto tehdystä tilauksesta tulisi näkyä näytöllä			
2	Paina ”Vahvista tilaus”		Järjestelmän tulisi siirtyä kassalle ja tilauksen tulee olla eriteltynä selkeästi ja tehtyjen lisävalintojen näkyä			
3	Klikkaa ”Jatka tilausta”		Järjestelmä palaa etusivulle ja ostoksia voi jatkaa			
4						

Post-conditions:

Project Name: Fafa's kioskiohjelmiston testaussuunnitelma

Test Case Template

Test Case ID: 5

Test Priority (Low/Medium/High): High

Module Name:

Test Title: Tilauksen vahvistgaminen ja maksaminen

Description: Kun tilaus on tehty kokonaisuudessaan, vahvistetaan tilaus. Tilausvahvistuksessa tulee olla tuotteet eriteltynä toisistaan. Mikäli tilauksessa on esimerkiksi kaksi samaa tuotetta, mutta toisessa extra chiliä, on nämä tuotteet eritelty toisistaan tilausvahvistuksessa.

Test Designed by: Milka Ukkonen

Test Designed date: 31.7.2020

Test Executed by:

Test Execution date:

Pre-conditions:

Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Klikkaa ”Vahvista tilaus”		Tilausvahvistuksen tulisi näkyä näytöllä ja tilauskensa voi tarkistaa vielä tässä vaiheessa.			
2	Klikkaa ”Vahvista”		Maksutapavalinnan tulisi tulla näkyviin			
3	Valitaan maksutapa		Kuittaus tilauksen vastaanottamisesta tulee tulla näkyviin.			
4						

Post-conditions:

Project Name: Fafa's kioskiohjelmiston testaussuunnitelma

Test Case Template

Test Case ID: 6

Test Priority (Low/Medium/High): High

Module Name:

Test Title: Tuotteiden hintojen silmämääräinen tarkistus

Description: Käydään läpi jokainen tuoteryhmä ja tarkistetaan silmämääräisesti, että jokaisella tuotteella on jokin hinta ja, että hinta näyttää oikealta ja tuotteeseen sopivalta.

Test Designed by: Milka Ukkonen

Test Designed date: 4.9.2020

Test Executed by:

Test Execution date:

Pre-conditions:
Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Klikkaa etusivulta ”Pita” ja ”Originaali” koko		Pita-valikoima ilmestyy ruudulle			
2	Tarkista hinnat		Jokaisella tuotteella on hinta			
3	Klikkaa etusivulta ”Salaatit ja mezet”		Salaatit ja mezet-valikoima ilmestyy ruudulle			
4	Tarkista hinnat		Jokaisella tuotteella on hinta			
5	Klikkaa etusivulta ”Sides”		Sides-valikoima ilmestyy ruudulle			
6	Klikkaa etusivulta ”Juomat ja jälkiruoat”		Kategoriat: Juomat, kuumat juomat ja jälkiruoat ilmestyy ruudulle			
7	Klikkaa ”Juomat”		Juomat-valikoima ilmestyy			
8	Tarkista hinnat		Jokaisella tuotteella on hinta			
9	Klikkaa ”Kuumat juomat”		Kuumat juomat-valikoima ilmestyy			
10	Tarkista hinnat		Jokaisella tuotteella on hinta			
11	Klikkaa ”jälkiruoat”		Jälkiruoat-valikoima ilmestyy			
12	Tarkista hinnat		Jokaisella tuotteella on hinta			
13	Klikkaa etusivulta ”Lasten ateriat”		Lasten ateriat-valikoima ilmestyy			
14	Tarkista hinnat		Jokaisella tuotteella on hinta			

Post-conditions:

Project Name: Fafa's kioskiohjelmiston testaussuunnitelma

Test Case Template

Test Case ID: 7
Test Designed by: Milka Ukkonen
Test Priority (Low/Medium/High): High
Test Designed date: 4.9.2020
Module Name:
Test Executed by:
Test Title: Tuotteiden tilaaminen eri kombinaatioilla ja ostoskorin tarkistus (ns. pistokoe)
Test Execution date:
Description: Tehdään omavalintainen tilaus, jossa on monta samaa tuotetta eri kombinaatioin. Tarkistetaan, että tuotteet näkyvät oikein ostoskorissa.

Pre-conditions:
Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Klikkaa etusivulta ”Syötkö täällä?”		Tilaussivu aukeaa			
2	Tee omavalintainen tilaus, jossa on ainakin kaksi samaa tuotetta, mutta toiseen valitaan esim. vähän chiliä ja toiseen ei ollenkaan		Tilaukset siirtyvät ostoskoriin			
3	Klikkaa ostoskori auki		Ostoskori aukeaa			
4	Tarkista, että tuotteet on jaoteltu oikein ostoskoriin		Ostoskorissa on jaoteltu tuotteet omille riveilleen. Ostoskorista pystyy näkemään, missä tuotteessa on esim. vähän chiliä.			
5	Klikkaa ”Vahvista tilaus”		Tilauksen yhteenveto aukeaa			
6	Tarkista, että tuotteet näkyvät oikein tilausvahvistuksessa		Tuotteet on jaoteltu omille riveilleen, myös ne tuotteet, jotka eroavat toisistaan vain esim. chilin määrällä.			

Post-conditions:

Project Name: Fafa's kioskiohjelmiston testaussuunnitelma

Test Case Template

Test Case ID: 8
Test Designed by: Milka Ukkonen
Test Priority (Low/Medium/High): High
Test Designed date: 4.9.2020
Module Name:
Test Executed by:
Test Title: Tuotteiden tilaaminen englanniksi ja ruotsksi
Test Execution date:
Description: Tehdään tilaus alusta loppuun (eli maksamiseen asti) englannin tai ruotsin kielellä ja tarkistetaan, että kielivalinta toimii.

Pre-conditions:
Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Klikkaa etusivulta kieleksi "englanti"		Tilaussivusto avautuu englannin kielellä.			
2	Tee vapaavalintainen testitilaus		Kieli pysyy englantina			
3	Tarkista silmämääräisesti, että kieli pysyy englantina koko tilauksen ajan		Kielivalinta pysyy englantina			
4	Vahvista tilaus		Tilauksen yhteenveto on englanninkielinen.			

Post-conditions:

Project Name: Fafa's kioskiohjelmiston testaussuunnitelma

Test Case Template

Test Case ID: 9
Test Designed by: Milka Ukkonen
Test Priority (Low/Medium/High): High
Test Designed date: 4.9.2020
Module Name:
Test Executed by:
Test Title: Tilaaminen wheelchair-toiminnolla
Test Execution date:
Description: Testataan järjestelmän wheelchair-toimintoa, joka mahdollistaa esim. pyörätuolia käyttävän henkilön tekemään tilauksen itsenäisesti.

Pre-conditions:
Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Klikkaa etusivulta "wheelchair"-painiketta		Näytön kuvasuhde muuttuu ja valinnat siirtyvät alemmas näytöllä			
2	Valitse "Syötkö täällä?"		Tilaussivusto aukeaa. Kuvasuhde pysyy samanlaisena.			
3	Tee vapaavalintainen tilaus ja tarkista, että sivusto pysyy wheelchair-toiminnossa		Sivusto pysyy koko tilauksen tekoajan wheelchair-toiminnossa			
4	Vahvista tilaus		Tilauksen yhteenveto näkyy wheelchair-toiminnossa			
5	Maksa tilaus		Maksutavat tulevat näkyviin oikeassa kuvasuhteessa			
6	Vahvista tilaus		Tilausvahvistus ilmestyy wheelchair-toiminnossa.			

Post-conditions:

Project Name: Fafa's kioskiohjelmiston testaussuunnitelma

Test Case Template

Test Case ID: 10

Test Priority (Low/Medium/High): High

Module Name:

Test Title: Euro- ja kappalemääräisesti iso tilaus

Description: Testataan ohjelmiston toimivuutta, kun ohjelmiston kautta tehdään kappalemääräisesti ja euromääräisesti iso tilaus. Tarkistetaan, että ostokori ja tilausyhteenveto pysyvät tilauksen mukana.

Test Designed by: Milka Ukkonen

Test Designed date: 4.9.2020

Test Executed by:

Test Execution date:

Pre-conditions:

Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Klikkaa etusivulta ”Syötkö täällä?"		Tilausetusivu aukeaa			
2	Valitse ostoskoriin paljon tuotteita, myös samoja tuotteita eri kombinaatioilla (esim. eri chili määrä)		Tuotteet siirtyvät ostoskoriin sitä mukaan. kun niitä valitaan tilattavaksi			
3	Tarkista, että tuotteet jaottuvat oikein ostoskorissa (esim. samat tuotteet, joissa on eri määrä chiliä, ovat omilla riveillään)		Tuotteet jaottuvat oikein ostoskoriin.			
4	Klikkaa ”Vahvista tilaus”		Tilauksen yhteenveto ilmestyy			
5	Tarkista, että tilauksen yhteenvetoon on tuotteet jaoteltu oikein.		Tuotteet jaottuvat oikein tilauksen yhteenvedolla.			

Post-conditions:

LIITE 2

VIRHERAPORTTI

Bug Report Template

ID number	
Name	
Reporter	
Submit Date	
Summary	
URL	
Screenshot	
Platform	
Operating System	
Browser	
Severity	
Assigned to	/
Priority	

Description

Steps to reproduce

Expected result

Actual result

Notes