Sirja Rönkä

PIISKA-PROJEKTINHALLINTAJÄRJESTELMÄN TESTAUSSUUNNITELMA JA TESTAUS

Opinnäytetyö

Insinööri (AMK)

Tieto- ja viestintätekniikan koulutus

Tekijä	Tutkintonimike	Aika
Sirja Rönkä	Insinööri (AMK)	toukokuu 2021

Opinnäytetyön nimi

Piiska-projektinhallintajärjestelmän testaussuunnitelma ja testaus

28 sivua, liitteitä 20 sivua

Toimeksiantaja

Kaakkois-Suomen ammattikorkeakoulu

Ohjaaja

Marko Oras

Tiivistelmä

Tämän opinnäytetyön tavoitteena oli toteuttaa testaussuunnitelma Piiska-projektinhallintajärjestelmälle, ja suorittaa manuaalinen testaus suunnitelman mukaan. Pääpainona oli järjestelmän käytettävyyden testaus ja arviointi.

Työ aloitettiin tutustumalla testauksen teoriaan ja eri testausmenetelmiin. Tästä edettiin testaussuunnitelman tekoon, johon sisältyi Excelissä toteutetut testitapaukset. Testitapaukset toimivat samalla testauksen raportointipohjana, johon tulokset kirjattiin. Itse testaus suoritettiin kahdella eri käyttöjärjestelmällä ja kahdella eri selaimella.

Testauksen tuloksena järjestelmässä havaittiin erilaisia ongelmia ja virheitä. Yhtään kriittistä järjestelmää rikkovaa virhettä ei löytynyt, mutta erityisesti mobiiliversion käytettävyydessä oli puutteita. Lisäksi havaittiin muita pieniä käyttäjää haittaavia virheitä ja ongelmia.

Opinnäytetyön tuloksia voidaan käyttää järjestelmän jatkokehityksessä, ja testaus voidaan toistaa tulevaisuudessa, kun järjestelmää päivitetään.

Asiasanat

testaus, ohjelmistot, käytettävyys

Author	Degree	Time
Sirja Rönkä	Bachelor of Engineering	May 2021

Thesis title

Piiska project management system's test plan and testing

28 pages, 20 pages of appendices

Commissioned by

South-Eastern Finland University of Applied Sciences

Supervisor

Marko Oras

Abstract

The objective of this thesis was to make a test plan for Piiska project management system and perform the test manually according to the plan. The main focus was to test and evaluate the usability of the system.

The work was begun by becoming acquainted with the theory of testing and with the different testing methods. After this, the test plan was made, which included the test cases that were carried out in Excel. The test cases also worked as the test report files. The testing itself was performed with two different operating systems and with two different browsers.

The result of the testing showed some bugs and faults in the system. No critical level bugs were found, but the mobile version especially had many usability issues. Furthermore, other small bugs and problems which disturb the user were detected.

The results of this thesis can be used in the future development of the Piiska system. The test plan can also be used again when the system is upgraded.

Keywords

testing, computer programs, usability

SISÄLLYS

1 JOHDANTO 6

2 TUTKIMUKSEN TAVOITTEET JA MENETELMÄT 7

3 TESTAUKSEN TEORIA LYHYESTI 8

3.1 Perusteet 8

3.2 Testausmenetelmät 9

3.2.1 Mustalaatikkotestaus 10

3.2.2 Lasilaatikkotestaus 10

3.2.3 Hyväksymistestaus 11

3.2.4 Regressiotestaus 12

3.2.5 Savutesti 12

3.2.6 Käytettävyystestaus 13

3.2.7 Kuormitustestaus 13

3.2.8 Järjestelmätestaus 14

3.3 Raportointi 14

4 TESTAUSSUUNNITELMA 15

4.1 Testauksen vaiheet 15

4.2 Testaamatta jäävät ominaisuudet 15

4.3 Testausympäristö ja raportointi 16

4.4 Testitapaukset 16

5 PIISKAN TESTAUS JA TULOKSET 17

5.1 Yleinen toimivuus tietokoneella 17

5.2 Yleinen toimivuus mobiililla 19

5.3 Käytettävyys tietokoneella 20

5.4 Käytettävyys mobiililla 23

6 TULOKSET 24

7 YHTEENVETO 25

LÄHTEET 27

LIITTEET

Liite 1. Testitapaus kirjautumissivu
Liite 2. Testitapaus kirjautuminen
Liite 3. Testitapaus ohjekirja
Liite 4. Testitapaus palaute
Liite 5. Testitapaus sivut
Liite 6. Testitapaus projektin luonti
Liite 7. Testitapaus projektin muokkaus
Liite 8. Testitapaus työtuntien kirjaus
Liite 9. Testitapaus projektiin liittyminen
Liite 10. Testitapaus käyttäjä
Liite 11. Testitapaus tehtävälista
Liite 12. Testitapaus kaaviosivu
Liite 13. Testitapaus tuntien lähetys
Liite 14. Testitapaus projektin lopetus
Liite 15. Testitapaus projektisivun käytettävyys
Liite 16. Testitapaus lokisivun käytettävyys
Liite 17. Testitapaus lokihistoriasivun käytettävyys
Liite 18. Testitapaus kaaviosivun käytettävyys
Liite 19. Testitapaus tuntikirjauksen rajat
Liite 20. Testitapaus lokin muokkaus

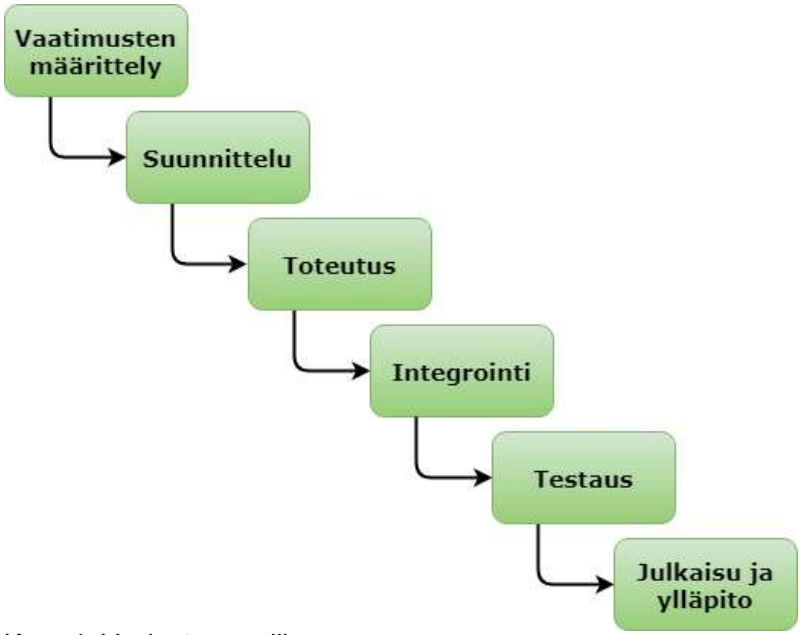
1 JOHDANTO

Ohjelmien muuttuessa yhä monimutkaisemmiksi on testauksesta tullut aiempaa tärkeämpää. Tällöin tarvitaan testaukseen erikoistuneita tiimin jäseniä. Testaaminen voi antaa uusia näkökulmia ja auttaa tekemään päätöksiä projektin ominaisuuksista ja suunnasta. Testaamisen päätarkoitus on kuitenkin virheiden löytäminen ohjelmasta, jotta ne päästään korjaamaan ennen julkaisua. (Levy & Novak 2010.)

Testauksen tulisi kuulua koko projektiin sen alusta loppuun saakka, eikä pelkästään lopuksi, kun kaikki on jo melkein valmista. Tässä vaiheessa suurempien muutosten tekeminen on työläämpää tai miltei mahdotonta, vaikka testaus osoittaisi niiden olevan tarpeellisia. Testaus aloitetaan yleensä ohjelman erillisitä komponenteista (yksikkötestaus), ja kehitysprosessin edetessä siirrytään koko ohjelman testaamiseen kokonaisuutena (järjestelmätestaus). Testaaminen ei kuitenkaan ole täysin lineaarista, vaan uusia ominaisuuksia ja korjauksia tehtäessä käytetään eri testausmenetelmiä. Tämän vuoksi esimerkiksi yksikkötestaus on yleistä myös ohjelman kehityksen loppuvaiheessa. (Homès 2012; Kasurinen 2013; Lyytinen 2019.)

Vesiputousmallissa (kuva 1) ohjelmiston kehittäminen nähdään eteenpäin kulkevana prosessina, jossa edelliseen vaiheeseen palaaminen on hankalaa, kallista tai jopa turhaa. Malli kuvaakin ohjelmistoprosessille tyypillisen kehityskaaren huonosti, ja sitä käytetään yleensä esimerkkinä huonosta kehitysmallista. Malli kuitenkin korostaa aikaisen suunnittelun ja hyvän dokumentoinnin tärkeyttä, mitä pidetään sen vahvoina puolina. (Kasurinen 2013.)

Tässä työssä käsitellään testauksen teoriaa lyhyesti, sekä Xamkin peliohjelmoinnin opiskelijoiden käytössä olevan Piiska-projektihallintajärjestelmän testaussuunnitelman tekoa ja itse testauksen suorittamista. Tuloksia voidaan hyödyntää järjestelmän kehitykseen ja käytettävyyden parantamiseen.



Kuva 1. Vesiputousmalli

2 TUTKIMUKSEN TAVOITTEET JA MENETELMÄT

Tämä opinnäytetyö on toiminnallinen opinnäytetyö. Se koostuu kirjallisesta teoria-, suunnittelu- sekä raporttiosasta ja kohteena olevan ohjelmiston manuaalisesta testaamisesta. Päätavoitteina on tehdä testaussuunnitelma Piiska-projektinhallintajärjestelmälle sekä suorittaa itse testaus. Lisäksi tavoitteena on saada tietoa, jota toimeksiantajan on tarkoitus hyödyntää Piiska-Projektihallintajärjestelmän kehittämiseen.

Keskeisenä tutkimusmenetelmänä tässä opinnäytetyössä on pääosin kvalitatiivinen eli laadullinen tutkimus. Tämä opinnäytetyö pyrkii vastaamaan seuraaviin tutkimuskysymyksiin:

- Miten web-sovellus testataan?
- Onko testattavan järjestelmän käytettävyys hyvä?
- Toimiiko järjestelmä virheittä?
- Mitä kehittämistarpeita järjestelmässä on?

Opinnäytetyön tavoite testauksen suunnittelusta ja suorittamisesta on sen eettisenä pohjana; työn tuloksena syntyy uutta tarpeellista tietoa. Näin ollen se täyttää hyödyllisyyden ja tarpeellisuuden määritteen. Luotettavuutta on pyritty lisäämään mahdollisimman ajantasaisilla kirjallisilla- sekä digitaalisilla lähteillä.

3 TESTAUKSEN TEORIA LYHYESTI

Testaus on aina päämäärällistä: se on sarja tehtäviä, joiden tarkoituksena on löytää ja tunnistaa ongelmia pelistä tai ohjelmasta ja arvioida samalla sen laatua (Homès 2012).

Testauksen voi jakaa sen suorittamistyylin mukaan kahteen osaan, joista ensimmäinen on manuaalitestaus. Manuaalitestauksessa testaaja itse testaa ohjelmaa ja sen ominaisuuksia toistaen testit tarvittaessa useaankin kertaan. Nykyaikaiset ohjelmat ovat usein niin monimutkaisia rakenteeltaan, että ihmisen tekemä manuaalinen testaus on välttämätöntä.

Toinen testauksen suorittamisen muoto on automaatiotestaus. Tässä muodossa jokin valmis ohjelma tai testaajan kirjoittama testiohjelma testaa ohjelmaa useita satoja, ja jopa tuhansia kertoja. Tämän tyylinen testaus sopii hyvin esimerkiksi koodia testatessa, jolloin manuaalinen arvojen syöttö veisi pitkän ajan. (Levy & Novak 2010.)

3.1 Perusteet

Testaajan tavoitteena on löytää ohjelmasta virheet, löytää polku, jolla ne saa toistettua, sekä raportoiden niistä kattavasti ja selkeästi. Toissijaisena tavoitteena on varmistaa, että virheet on korjattu. (Levy & Novak 2010; Kasurinen 2013.)

Testiryhmän johtaja aloittaa uuden projektin testaamisen laatimalla testisuunnitelmaan. Testisuunnitelma toimii testaustiimin ohjekirjana. Se sisältää ohjelman kuvauksen, aikataulun, tarkastuslistan kaikista testitapauksista, testien päämäärät sekä mahdollisen luettelon testattavista ja ei-testattavista asioista. (Kasurinen 2013; Testbytes 2018.)

Testitapaus on yksittäinen testi, jonka yksi tai useampi testaaja suorittaa. Jokaisella tapauksella on oma yksilöllinen objektiivinsa. Jokaista yksittäistä vaihetta testitapauksen sisällä kutsutaan testiaskeleeksi (test step). (Software Testing Fundamentals 2020; Schultz & Bryant 2012.)

Tarkastuslistat ovat esimerkiksi Excel-taulukoita, jotka sisältävät listan testitapauksista. Yleensä jokaisella testaajalla on oma taulukkonsa, joka sisältää erilaisia testauksen kohteita. Testitulokset kuvaavat jonkun tietyn toiminnon testaamista, kuten esimerkiksi tallentamisen ja tallennuksen latauksen toimivuutta. (Katara, Vuori, ym. 2015.)

Virheitä on monia eri tyylisiä, kuten esimerkiksi visuaalinen-, lataus- ja suorituskykybugi. Virheillä on myös erilaisia vakavuusasteita, jotka määritellään usein tarkemmin testaussuunnitelmassa (Levy & Novak 2010, 50):

- Vähän haittaavat (Low Priority) virheet eivät ole merkittäviä. Näitä voivat olla esimerkiksi hyvin pieni graafinen virhe tai jokin muu ongelma, joka ei haittaa ohjelman käyttämistä tai vaikuta sen mielekkyyteen. Myös muun muassa kirjoitusvirheet kuuluvat näihin. Jos aikataulu on tiukka, saatetaan ohjelmaan jättää näitä, vaikka ne olisikin raportoitu.

- Keskiverto-virheet (Medium Priority) toistuvat kohtalaisen usein ja ne pitäisi korjata ennen julkaisua. Nämä eivät yleensä vaikuta käyttäjän etenemiseen mutta voivat muuten haitata kokemusta.

- Suuret virheet (High Priority) pitäisi ehdottomasti korjata koska ne haittaavat käyttämistä merkittävästi. Tällaisia voivat olla esimerkiksi se, että yksi osa ohjelmasta ei toimi lainkaan.

- Kriittiset virheet (Critical) ovat erikoistapauksia ja ne vaativat koko työtiimin välittömän huomion. Näitä voivat olla esimerkiksi datan korruptoituminen. Tämän tyyliset virheet pitäisi korjata mahdollisimman nopeasti.

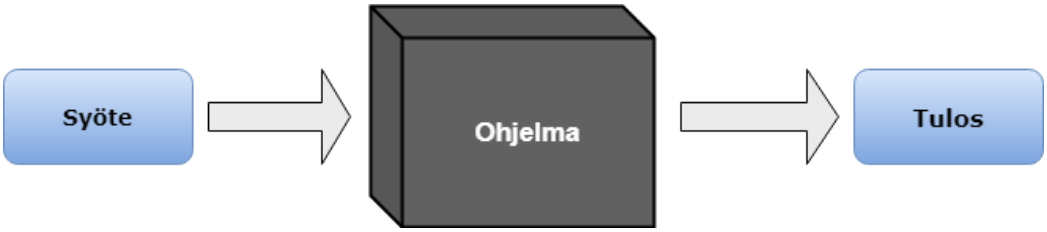
3.2 Testausmenetelmät

Staattinen ja dynaaminen testaaminen termeinä kuvaavat, kuinka testaus tapahtuu ja millaista se pohjimmiltaan on. Staattinen testaaminen tarkoittaa sitä, että itse järjestelmää ei käytetä vaan sitä arvioidaan esimerkiksi

koodianalysaattorilla. Dynaamisessa testauksessa puolestaan järjestelmää käytetään ja sen reaktioita seurataan. Staattinen testaaminen korostuu usein jo suunnitteluvaiheessa, kun taas dynaaminen ohjelman kehityksen loppupuolella. Staattisella testauksella kiinni saadut ongelmat ovat paljon halvempi korjata, kuin jos samat virheet huomattaisiin vasta dynaamisessa testauksessa. Useimmat testausmenetelmät kuuluvat dynaamiseen testaukseen. (Kasurinen 2013, 65.)

3.2.1 Mustalaatikkotestaus

Perinteisin testausmuoto, jossa ohjelmalle annetaan syötteitä ja tarkastellaan lopputulosta välittämättä siitä, mitä ohjelman sisällä tapahtuu (kuva 2). Yleensä testattavat syötteet ym. määritellään etukäteen testitapauksissa. Menetelmää voidaan käyttää testauksen kaikissa vaiheissa, jos ohjelmasta on olemassa käynnistyvä versio.



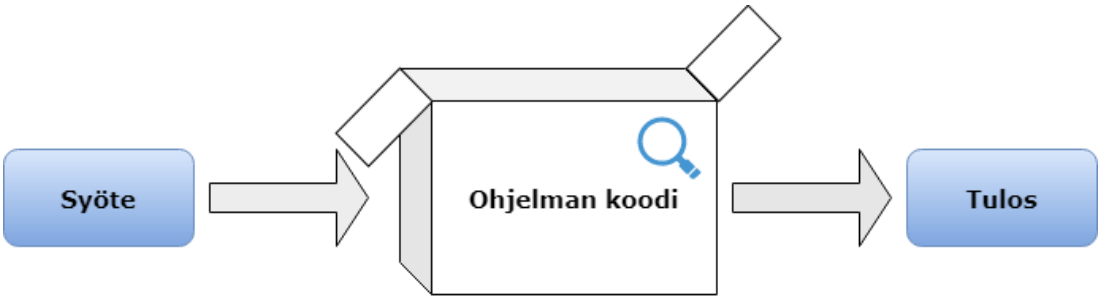
Kuva 2. Mustalaatikkotestaus

Mustalaatikkotestaus tehdään usein tiimin ulkopuolelta, ja sen tekee yleensä joku sellainen, joka ei tiedä kyseisestä koodista mitään. Tällainen testaus mahdollistaa testaajan asettumisen loppukäyttäjän rooliin parhaiten, ja on tehokkain keino löytää bugeja. Testaajan tehtävänä on vain varmistaa, että syötteet ovat oikein ja lopputulos vastaa odotettua. Mustalaatikkotestauksen yksinkertainen luonne johtaa myös usein siihen, että tällaiset testitapaukset ovat automatisoituja. (Kasurinen 2013, 65–66; Schultz & Bryant 2012, 126–127; Lee 2017; Moore 2015.)

3.2.2 Lasilaatikkotestaus

Lasilaatikkotestaus on käytännössä hienompi versio Mustalaatikkotestauksesta, jossa syötteiden ja tuloksen lisäksi kiinnitetään myös huomiota järjestelmän sisäiseen toimintaan testauksen aikana.

Menetelmä antaa testaajalle mahdollisuuden päästä näkemään koodi suoraan (kuva 3). Esimerkiksi ohjelmoijien itsensä tekemä testi koodille uuden lisäyksen jälkeen kuuluu tähän kategoriaan.



Kuva 3. Lasilaatikkotestaus

Lasilaatikkotestit ovat mustalaatikkotestejä tarkempia ja niitä tehdessä testaaja pystyy jäljittämään virheen suoraan kooditasolla. Lasilaatikkotestaus vaatii siis testaajalta testattavan järjestelmän hyvää tuntemusta, sekä ymmärrystä ohjelmoinnista. Menetelmällä voidaan varmistaa, että koodi oikeasti toimii halutulla tavalla, eikä oikea lopputulos ole vain sattumaa. Lasilaatikkotestit eivät kuitenkaan kerro esimerkiksi puutteista ominaisuuksissa, joten ne yksin eivät riitä laadun testausmenetelmänä. (Kasurinen 2013, 67–68; Schultz & Bryant 2012, 128–129; Lee 2017.)

3.2.3 Hyväksymistestaus

Hyväksymistestaus on perinteisen testausmallin viimeinen osa, jossa varmistetaan, että ohjelma on määritysten mukainen ja laadukas. Täydellinen ohjelman ei tarvitse olla, mutta sen pitää täyttää etukäteen sille määritellyt vaatimukset.

Tämä testaus suoritetaan normaalisti kohdeympäristössä, erona aikaisempiin vaiheisiin, joissa testaus suoritetaan yleensä erillisessä testausympäristössä. Hyväksymistestauksen jälkeen ohjelma tavallisesti siirtyy käytettäväksi ja sen tilanneelle asiakkaalle. (Kasurinen 2013, 57; Pesonen s.a.)

3.2.4 Regressiotestaus

Regressiotestauksessa testataan päivitettyä versiota ohjelmistosta. Vanhoja virheitä yritetään toistaa, ja varmistetaan että ne on saatu korjattua. Samalla testataan, että tehdyt korjaukset eivät ole hajottaneet mitään muuta osaa ohjelmasta. Regressiotestauksen nimi tulee siitä, että siinä tarkistetaan, onko koodi korjauksen myötä regressoitunut eli mennyt taaksepäin kehityksessä. Yksinkertaisten virheiden, kuten kirjoitusvirheiden, korjauksen tarkastus on melko yksinkertaista. Vakavammat virheet, kuten ohjelman kaatuminen, voivat vaatia laajempaa regressiotestausta, sillä yhden testin läpäisy ei tarkoita, että tallaiset virheet on kokonaan korjattu. Vakavat virheet voidaan pitää testiraportissa avoinna parin-kolmen projektiversion ajan, jolloin voidaan paremmin todeta ne varmasti korjatuiksi. (Schultz & Bryant 2012, 139, 332; Pesonen s.a.)

3.2.5 Savutesti

Savutesti tehdään yleensä jonkin uuden lisäyksen jälkeen, ja sillä määritellään, toimiiko ohjelma niin, että se voidaan lähettää testattavaksi. Tämä on yksinkertainen testi, jossa katsotaan, että ohjelma käynnistyy ilman suurempia ongelmia. Näin vältetään turhaa edestakaista työtä, jossa testaajat saisivat version ohjelmista, jota ei pysty testaamaan. (Schultz & Bryant 2012, 138.)

Kuva 4. Savutesti

Savutestejä voidaan tehdä sekä manuaalisesti että niitä voidaan automatisoida. Jos savutesti ei mene läpi, palaa ohjelma korjattavaksi. Tämän jälkeen korjaus tarkistetaan uudella savutestillä ennen kuin ohjelma pääsee eteenpäin testattavaksi. Prosessi on havainnollistettu kuvassa 4.

3.2.6 Käytettävyystestaus

Käytettävyydelle on erilaisia määritelmiä. Esimerkiksi käytettävyyskonsultti Jakob Nielsen määrittelee sen koostuvan seuraavista osa-alueista: opittavuus, tehokkuus, muistettavuus, virheettömyys ja tyytyväisyys. Tämän testauksen käytettävyyden arviointi perustetaan pääosin edellä mainittuihin ominaisuuksiin. (Nielsen 2012.)
Käytettävyystestauksessa siis tutkitaan, kuinka hyvin suunniteltu tai jo toiminnassa olevan ohjelmiston käyttöliittymä toimii. Testataan, onko se helppokäyttöinen, intuitiivinen ja looginen ja ovatko värit, kontrastit ja siirtymät hyväksyttäviä. Testataan myös, joutuuko käyttäjä odottamaan toiminnallisuuksia tai latauksia liian kauan.

Tiivistettynä menetelmä siis testaa, onko käyttöliittymä suunniteltu ja toteutettu oikein. Käytettävyyden testaukseen on useita eri lähestymistapoja. Näitä ovat esimerkiksi käyttökokeilut kohderyhmällä ja asiantuntijan tekemät testit. Testeihin on usein myös liitettynä työkaluja, joilla voidaan esimerkiksi tallentaa käyttäjän syötteet tai seurata tämän katsetta testausta suorittaessa. Myös testaustilanteen nauhoitus on yleisesti käytetty apukeino, koska ilmeiden ja eleiden avulla saadaan tarkempaa palautetta kuin mitä testaaja saattaa haastattelussa muistaa kertoa. (Kasurinen 2013, 70–71; Pesonen s.a.)

3.2.7 Kuormitustestaus

Menetelmässä luodaan aitoa käyttötilannetta vastaava tilanne, jossa esimerkiksi virtuaalikäyttäjät simuloivat normaalia toimintaa. Testauksen kohteena voi olla muun muassa verkkokauppa. Kuormitustestaus pyrkii tarkastelemaan kolmea erilaista ongelmakohtaa: järjestelmän pullonkauloja, maksimikapasiteettia ja selviytymistä normaaleista käyttöolosuhteista. Mikäli järjestelmä ei suoriudu odotetusti, on löydetty virhe, joka vaatii korjaamista.

Laajennettu muoto tästä on rasitustestaus, joka pyrkii löytämään järjestelmän normaalin toiminnan ehdottoman ylärajan. Tässä järjestelmää rasitetaan suunniteltua suuremmalla käyttäjämäärällä tai muilla toiminnoilla ja syötteillä. Näin voidaan esimerkiksi havaita yhtäaikaisten kutsujen aiheuttama lukkiutuminen. (Kasurinen 2013, 71–72.)

3.2.8 Järjestelmätestaus

Järjestelmätestaus itsessään ei käytännössä ole testaustapa, vaan se on yleisnimitys kaikelle kokonaisen järjestelmän testaamiselle, ja se on perinteisen testausmallin kolmas vaihe. Tässä vaiheessa järjestelmä on koottu yhdeksi kokonaisuudeksi, jota voidaan myös testata kokonaisuutena yksittäisten osien sijaan. Testauksen tavoitteena on varmistaa, että järjestelmä toimii kokonaisuutena. Tässä vaiheessa testaus suoritetaan vielä testiympäristössä, kun myöhemmin se tapahtuu käyttöympäristössä.

Teoriassa tässä vaiheessa suoritetaan musta- ja lasilaatikkotestausta, mutta käytännössä testaustavat riippuvat projektista. Testauksen tavoitteena on vielä tässä vaiheessa aktiivisesti löytää puutteita ja virheitä, ja ohjelmaan tehdään yleensä testauksen myötä muutoksia. Myöhemmin tulevassa hyväksymistestaus-vaiheessa painopiste siirtyy vastaavasti toiminnallisuuden testaukseen, eikä suuria muutoksia ohjelmistoon enää tehdä. Tämä kaava ei kuitenkaan välttämättä toimi joka projektissa samalla tavalla, ja testausprosessi muokkautuu aina tarpeeseen. (Kasurinen 2013, 56–57.)

3.3 Raportointi

Testaajan yksi tärkeimmistä taidoista on osata kirjoittaa hyvin. Testauksen yhteydessä kirjoitetaan aina raportti, josta löydetyt virheet käyvät ilmi. Virheiden kuvausten tulisi olla tarkkoja ja helposti luettavia. Laajasta sanavarastosta on tässä apua. Ohjelmoijat ja suunnittelijat käyttävät testiraporttia apunaan virheiden korjaamisessa ja uusien ominaisuuksien teossa, joten raporttia kirjoittaessa on hyvä muistaa, että myös muut lukevat sen. (Kinsbruner 2020.)

Ohjelmaa testatessa on tärkeää tehdä muistiinpanoja raporttia varten. Heti virheen löydyttyä kirjoitetaan ylös, mitä sillä hetkellä tehtiin. Seuraavana kone tai ohjelma käynnistetään uudelleen ja yritetään toistaa virhe muistiinpanojen perusteella. Jos virheen saa toistettua, voi siitä kirjoittaa raportin. Jos virhe ei heti toistu, sitä testataan useampaan kertaan. Lopputuloksen ollessa hyväksytty, eli virhettä ei pysty toistamaan, voi siitä jättää itselleen muistiinpanon mutta virallista bugiraporttia ei tehdä. Muistiinpanoja on hyvä

tehdä myös yleisellä tasolla esimerkiksi kokouksissa, tai testauksen aikana heränneiden kysymysten muodossa. Nämä auttavat sekä testaajaa itseään että muuta tiimiä. (Moolya Testing 2020; Levy & Novak 2010.)

Yksi tapa testauksen raportoinnille ovat myös näyttökuvat ja testauksen videointi. Nämä auttavat havainnollistamaan ongelmakohtia, ja etenkin videolta voidaan helpommin määritellä, mikä ongelma on katsomalla se useampaan kertaan. Tämä toimii hyvin esimerkiksi visuaalisten virheiden kanssa. (Levy & Novak 2010, 126–127.)

4 TESTAUSSUUNNITELMA

Testauksen kohteena on Xamkin peliohjelmoinnin käytössä oleva Piiska-projektinhallintajärjestelmä. Järjestelmän päätarkoituksena on, että opiskelija pystyy tallentamaan sinne tiedot omista meneillään olevista projekteistaan, ja niiden tuntikirjanpidot. Tietojen on myös tarkoitus säilyä järjestelmässä projektin valistumisen jälkeen. Testauksen tavoitteena on testata järjestelmän toimivuutta ja käytettävyyttä. Järjestelmälle suoritetaan järjestelmätestaus, ja päämenetelminä ovat hyväksymis- ja käytettävyystestaus. Testaus suoritetaan loppukäyttäjän näkökulmasta ja tehdään itse järjestelmää käyttäen.

4.1 Testauksen vaiheet

Normaalisti järjestelmälle suoritetaan useita eri testauksia eri kehitysvaiheissa. Piiska on jo käytössä oleva järjestelmä, joten sille suoritetaan tässä tapauksessa vain järjestelmätestaus.

Uusia ominaisuuksia tai päivityksiä ei ole heti tiedossa, joten esimerkiksi regressiotestaus ei ole tässä vaiheessa tarpeen. Testaus suoritetaan suunniteltujen testiaskeleiden mukaan.

4.2 Testaamatta jäävät ominaisuudet

Järjestelmän kuormituskykyä ei tässä testauksessa tehdä, koska testaus tehdään manuaalisesti yhden testaajan toimesta. Myöskään koodia ei testata

tai tarkasteta erikseen, vaan testaus suoritetaan mustalaatikko-menetelmän mukaisesti.

Testauksen yhteydessä saattaa myös tulla eteen joitain ominaisuuksia, jotka eivät ole olleet ennakkoon tiedossa. Niitä ei lisätä jälkikäteen enää testitapauksiin, eikä myöskään testata tämän testauksen yhteydessä.

4.3 Testausympäristö ja raportointi

Testausympäristönä toimivat Google Chrome- ja Mozilla Firefox -selaimet, ja niitä käytetään tietokoneella Windows 10 ja kännykällä Android 9 -käyttöjärjestelmällä. Piiskan suositeltu selain on Google Chrome.

Raportointiin käytetään liitteenä löytyvää valmiiksi määriteltyä testitapauspohjaa, joka sisältää seuraavat tiedot:

1. Testitapaus ID
2. Testitapauksen kuvaus
3. Odotettu tulos
4. Testiaskeleet eli test steps
5. Päiväys
6. Järjestelmäversio
7. Testaaja
8. Käyttöjärjestelmä
9. Selain
10. Pass/Fail
11. Testauksen tulos.

4.4 Testitapaukset

Testitapaus on yksi tapahtuma, jolla tarkistetaan jonkin ohjelman ominaisuuden toimivuus. Jokaisella testitapauksella on oma yksilöivä ID sekä kuvaus suoritettavasta testistä. Testitapauksessa on myös määritelty tarkkaan askeleet, joiden mukaan testaus suoritetaan, sekä kirjattu mikä on testin läpäisykriteeri, jolla se saa testauksesta merkinnän pass eli hyväksytty. Virheen löydyttäessä kirjataan fail eli hylätty ja tarkka selostus virheestä.

Useista testitapauksista muodostuvaa kokonaisuutta kutsutaan testijoukoksi. (Kasurinen 2013, 118–119; Schultz & Bryant 2012, 107–108; Software Testing Fundamentals 2020.)

Tässä tehtävien testitapausten yleisenä ennakkoehtona on, että testaajalla on järjestelmään toimivat tunnukset. Lisäksi kirjautumistestausta lukuun ottamatta ennen testien suorittamista kirjaudutaan aina järjestelmään. Kaikki tämän testauksen testitapaukset ja -raportit löytyvät lopusta liitteinä.

5 PIISKAN TESTAUS JA TULOKSET

Testaussuunnitelman jälkeen siirryttiin itse testaukseen ja se tehtiin testitapauksissa löytyvien testiaskeleiden mukaan. Testien suorittaminen aloitettiin Windows 10 käyttöjärjestelmällä käyttäen Google Chrome (versio 90.0.4430.212) selainta. Pääsääntöisesti testeissä edettiin ensimmäisestä (liite 1) viimeiseen (liite 20). Käytettävyyttä kuitenkin testattiin käytännössä koko testauksen ajan, ja näihin testikohtiin (liitteet 15, 16, 17 ja 18) kirjoitettiin kommentteja aina kun oli tarpeellista.

Ensimmäisen testiosuuden jälkeen tehtiin samat testit Mozilla Firefox -selaimella (versio 88.0.1), ja tämän jälkeen ne vielä toistettiin molemmilla selaimilla käyttäen mobiililaitetta, jossa on Android 9 -käyttöjärjestelmä. Yhteensä testaus suoritettiin siis neljä kertaa. Mobiilissa Google Chromen versio oli 90.0.4430.210 ja Mozilla Firefoxin 88.1.3.

Mobiililaitteella testaus tehtiin pystynäkymässä, koska alustava testaus osoitti, että järjestelmä toimi siten parhaiten. Vaakasuunnassa järjestelmä oli käytännössä käyttökelvoton. Pystynäkymän käyttö on lisäksi yleistä ja usein oletettu käyttömuoto.

5.1 Yleinen toimivuus tietokoneella

Testauksessa testattiin yleisimmät järjestelmään liittyvät toiminnot. Tietokoneella ei havaittu erityisen suuria virheitä tai ongelmia toimintojen suhteen, vaan havaitut viat olivat enemmän kosmeettisia tai menivät

käytettävyyden puolella. Suurin osa testeistä (liitteet 2, 3, 5, 9, 10, 12, 13, 14 ja 20) meni tietokoneella läpi ilman erityisiä kommentteja.

Havaitut suurimmat ongelmat olivat liitteiden 7, 11 ja 19 testeissä. Liitteessä 7 testattiin projektin tietojen muokkausta. Muokkauksen jälkeen tiedot eivät päivittyneet mistä syystä testi sai hylätyn tuloksen (fail). Järjestelmä testattiin normaalina käyttäjän, joten on mahdollista, että muutokset pitää hyväksyä opettajan toimesta ennen niiden näkymistä. Tämä ei kuitenkaan käynyt ilmi järjestelmästä tai ohjeesta.

Liitteen 11 testissä testattiin ”To Do”- eli tehtävälista -ominaisuutta, jossa itselleen voi kirjoittaa ylös tulevat tehtävät. Listaa ei kuitenkaan saatu näkyviin, eikä ohjeesta käynyt ilmi millä ehdoilla lista ilmestyy. Tästä syystä testin tulos oli hylätty eli fail.

Liitteen 19 testi koski tuntikirjaukseen asetettuja rajoja. Yhdelle päivälle pystyy laittamaan vain yhden merkinnän samalla kirjauskerralla, mutta järjestelmä ei tarkista eri merkintökertoja päällekkäisyyksien varalta. Tästä johtuen samalle päivälle voi merkitä yli 24 tuntia työtä, ja myös identtisiä tuntimerkintöjä voi tehdä useamman. Tämä voi johtaa potentiaaliseen virheeseen huolimattomuudesta johtuen, jolloin käyttäjän voi olla vaikea muistaa myöhemmin mille päivälle tunnit kuuluvat, ja onko ne esim. merkinnyt vain kahteen kertaan. Järjestelmä hyväksyy myös negatiiviset tunnit. Esimerkiksi yhden tunnin työskentely ja 90 minuutin tauko hyväksytään ilman huomautusta järjestelmältä ja merkataan lokiin muotoon 0 tuntia. Tällainen syöte voi olla pelkkä virhe käyttäjän puolelta, jolloin järjestelmän olettaisi huomauttavan siitä.

Pienempiä kosmeettisia ym. käyttöä haittaamattomia vikoja oli liitteiden 4, 6 ja 8 testeissä. Liitteen 4 testissä testattiin palautteen antamista, ja siinä havaittiin, että palautesivulta ei ole suoraa painiketta takaisin etusivulle tai järjestelmään, vaan paluun joutuu tekemään selaimen sivunvaihtonuolilla. Lisäksi tyhjästä palautekentästä ei tullut ilmoitusta, eli tyhjän palautteen lähetys oli mahdollista.

Liitteen 6 testissä testattiin projektin luominen, ja sen yhteydessä havaittiin, että järjestelmän kello ei seuraa Suomen GMT+2 -aikaa, vaan näyttää olevan asetettu GMT -aikaan. Tästä johtuen projektin luontipäiväksi saattaa tulla väärä päivämäärä, jos se luodaan pian puolenyön jälkeen. Tämä ongelma liittyy myös liitteen 8 testiin, jossa kirjattiin tunteja. Järjestelmän eroavasta aikavyöhykkeestä johtuen tuntien kirjaus samalle päivälle onnistuu vasta kolme tuntia suomen puolta yötä jäljessä (tai talviajassa kaksi tuntia jäljessä). Ongelma ei sinänsä ole suuri, jos järjestelmän käyttö rajoittuu päiväsaikaan, mutta opiskelijat saattavat todennäköisesti luoda projekteja ja kirjata tunteja mihin vuorokaudenaikaan tahansa.

5.2 Yleinen toimivuus mobiililla

Kännykällä testattaessa ongelmia ja virheitä oli enemmän. Iso osa niistä liittyi kuvakkeiden kokoon tai sivun asetteluun, joka vaikeutti järjestelmän käyttöä. Sisään- ja uloskirjautumisen testauksessa (liite 2) kävi ilmi, että kännykällä taaksepäin-nappi päästi käyttäjän takaisin järjestelmään uloskirjautumisen jälkeen. Tietokoneella järjestelmä vaati kirjautumisen uudelleen eikä päästänyt ”takaisin”-nuolella käyttäjää enää sisälle.

Liitteen 3 testissä huomattiin, että ohjekirjaa ei ollut saatavilla mobiilinäkymässä. Liitteessä 5 olevassa testissä testattiin sivujen yleinen avautuminen, jolloin havaittiin että ”Charts”- eli kaaviot-sivun asettelu on pahasti päällekkäin mobiililla. Lisäksi tehtiin huomio, että mobiilivalikossa sivujärjestys on eri kuin tietokoneella. Tähän palataan myöhemmin lisää ”käytettävyys”-osiossa.

Liitteessä 6 oleva testi keskittyi projektin luontiin, ja Mozilla Firefox -selaimella havaittiin, että projektin tilaajan, ryhmäläisten ja merkintöjen käyttämät pudotusvalikot eivät näkyneet, joten näitä tietoja ei voinut projektiin laittaa.

Liitteen 12 testi koski kaaviot-sivua, ja siinä keskityttiin testaamaan kaavioiden näkyminen ja suodatustoiminnot. Kuten aikaisemmin mainittiin, oli sivun asettelu pahasti päällekkäin mobiililla. Sivun asettelu oli tietokoneen mukaisesti vaakasuuntaan, vaikka mobiililla sen olisi parempi olla pystynäkymään tehty. Päällekkäisyydestä johtuen osa suodattimista oli

kaavionäkymän alla ja näin ollen mahdotonta käyttää. Kaaviota pystyi kuitenkin tekemään, mutta toimivuus on parempi tietokoneella. Kaaviot-sivu olisi hyvä olla saatavilla vain tietokoneella, ja piilottaa se mobiililla.

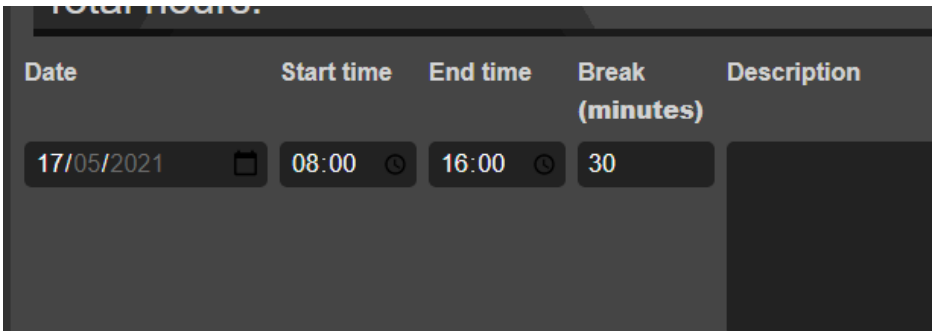
5.3 Käytettävyys tietokoneella

Käytettävyyttä testattiin pääasiassa liitteiden 1, 15, 16, 17 ja 18 testeissä. Liitteen 1 testissä arvioitiin etusivua ja kirjautumista. Sivun asettelu voisi olla parempi, esimerkiksi sivun tiedot ovat venyneet pituussuunnassa niin että sivua joutuu rullaamaan, vaikka kaikki tiedot mahtuisivat näytölle. Lisäksi palautenappi on piilotettu alareunukseen (kuva 5), josta se on vaikea huomata. Virheellisistä kirjautumistiedoista ei tule mitään palautetta, jolloin on vaikea tietää, missä vika on. Oletettavasti tämä on turvallisuussyistä, mutta se ei ole kovin käyttäjäystävällistä, koska myöskään ohjeita asiasta ei löydy.

Kuva 5. Sivun alareuna, jossa vasemmalla on palautenappi (puhekupla)

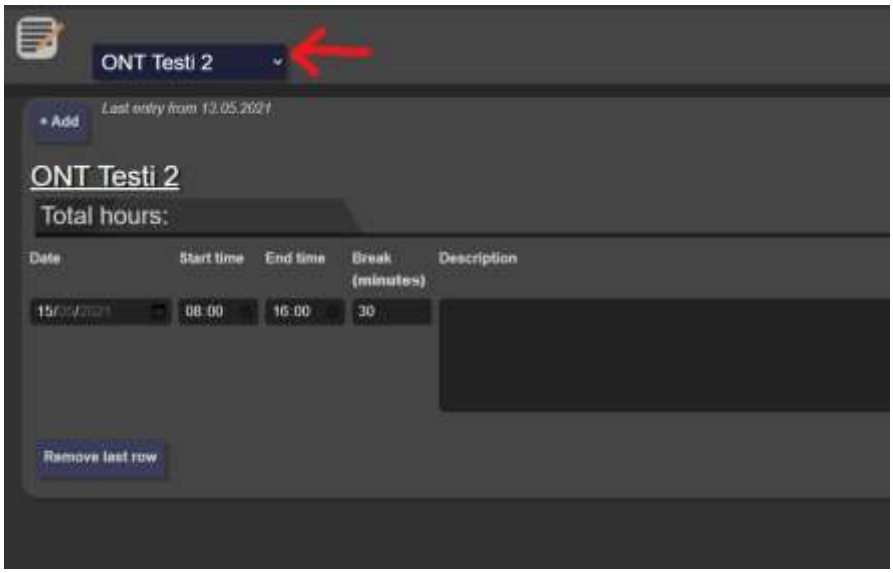
Liitteessä 15 oleva testi keskittyi ”Projects”- eli projektisivun käytettävyyteen. Sivulla ei ollut mitään isoja käytettävyyteen liittyviä ongelmia. Yleisilme oli ensinäkemältä hieman sekava ja kaikki kohdat piti lukea tarkasti läpi, mutta toiminnot oppi nopeasti.

Liite 16:n testi arvioi ”Log”- eli lokisivun käytettävyyttä. Google Chrome -selaimella päivä- ja kellonaikavalikoiden pienet kuvakkeet olivat liian tummat eivätkä näkyneet kunnolla kuin tietyssä kulmassa (kuva 6).



Kuva 6. Heikosti näkyvät kalenteri- ja kellosymbolit

Projektin, johon tuntikirjausta tehtiin, valinta oli jokseenkin huomaamaton yläreunassa (kuva 7), josta se oli helppo ohittaa ja epähuomiossa kirjata tunnit väärään projektiin. Tuntimerkinnän lisääminen voisi alkaa sillä, että projekti valitaan, nyt se on automaattinen, kun sivulle tullaan. Huomiota kiinnitettiin myös siihen, että sivun teksteissä oli käytetty jonkinasteista varjostusta mikä ei ollut silmäystävällistä (kuva 8).



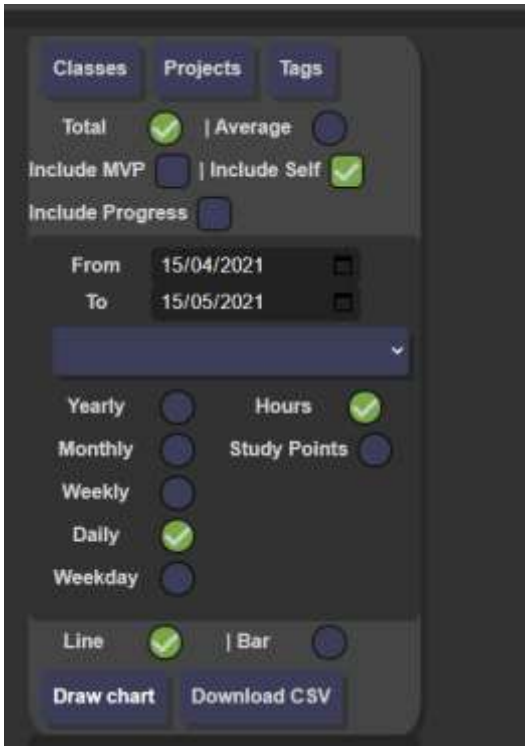
Kuva 7. Projektin valintapainike

Kuva 8. Tekstin varjostus

”Log History”- eli lokihistoria-sivun (liite 17) käytettävyystesti toi ilmi muutamia pieniä korjauksen kohteita. Lokimerkintöjen laajennustoiminto ei toiminut täydellisesti, vaan välillä hiirtä joutui liikuttamaan useaan kertaan edestakaisin, jotta haluttu merkintä saatiin laajennettua lukumuotoon. Merkintöjen olisi pitänyt avautua heti kun hiiri viedään niiden päälle. Muokkaustilanteessa pelkkä muokkauksen aktivoiva kynän kuva muutti väriään, mutta käytettävyyden kannalta olisi parempi, jos esimerkiksi lokialueen reunat olisivat myös värilliset. Näin on helpompi havaita, jos muokkaus on päällä. Tehtiin myös havainto, että muokkauksen ollessa aktiivinen projektin nimi

avaa projektinäkymän oikealle, mikä menee osittain muokkausikkunan alle Google Chrome -selaimessa.

Liite 18:n testi keskittyi ”Charts”- eli kaaviosivuun. Tämä oli kaikista sivuista monimutkaisin käytettävyyden kannalta (kuva 9), ja vaati ohjekirjan lukemista. Sivulta ei löydy minkäänlaista ohjenappia, vaikka sellaiselle olisi tarvetta. Kaavioiden lataaminen koneelle oli epäkäytännöllistä, koska kaikki session aikana luodut kaaviot ladattiin yksitellen. Ainut keino tyhjentää kaaviolista oli siirtyä toiselle sivulle ja sitten takaisin. Tämän jälkeen haluttu kaavio piti tehdä uudelleen, jos vain sen halusi ladata. Kaavioiden nimeäminen oli myös työlästä, koska nimi-ikkuna deaktivoitui ensimmäisen kirjaimen jälkeen. Olisi myös kätevää, jos nimi siirtyisi tallennettavan tiedoston oletusnimeksi. Sivun vasemmassa alareunassa on ylä- ja sivupalkkien piilotusnappi sekä nappi, joka aktivoi diaesityksen-esityksen. Näissä voisi näkyä selitykset mitä ne tekevät, kun hiiren vie niiden päälle. Sivuhuomiona mainittakoon, että kaaviosivu on ekstrana järjestelmässä, eikä ole sen ydintoimintoja.



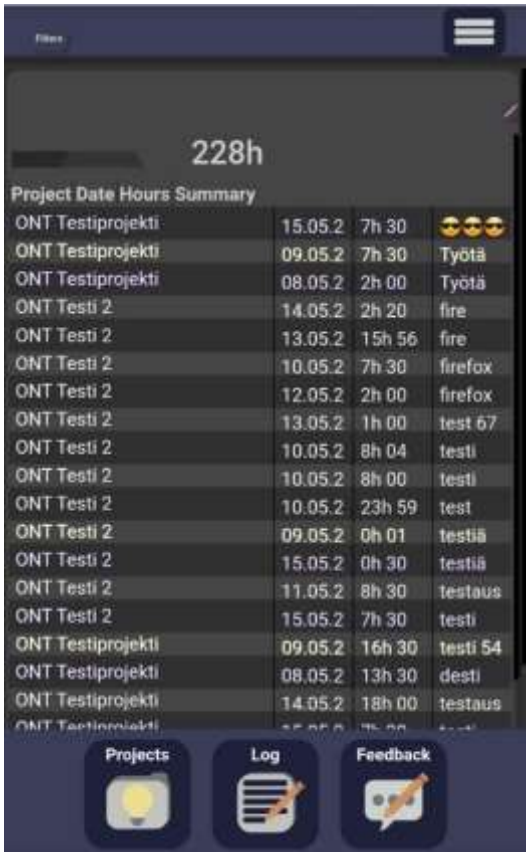
Kuva 9. Kaavioiden suodatusvalikko

5.4 Käytettävyys mobiililla

Järjestelmää ei ole tässä vaiheessa vielä optimoitu kunnolla mobiiliin, joten kännykällä testattaessa sivujen käytettävyys vaihteli suuresti. Kirjautumissivulla (liite 1) ei ilmennyt suuria vikoja, mutta Google Chrome -selaimella kirjautumiskohta oli todella pieni muuhun sivuun verrattuna, ja Mozilla Firefoxilla puolestaan ”disclaimer”-osio oli todella isolla fontilla.

Projektisivulla (liite 15) mobiiliversion asettelu oli tietokonetta selkeämpi ja helpompi täyttää, kun tiedot menivät ylhäältä alaspäin, sen sijaan että olisivat olleet osittain vierekkäin. Mozilla Firefox ei näyttänyt pudotusvalikkoja, joten osaa tiedoista ei voinut täyttää.

Lokihistoria-sivulla (liite 17) suurin ongelma käytettävyyden kannalta oli, että kirjattuja tunteja ei saanut avattua tai ne avautuivat mutta muun listan alle (kuva 10). Kirjattuja tietoja ei voi siis kunnolla tarkistaa. Listan pitäisi olla myös skaalattu isommaksi, jotta oikean merkinnän painaminen olisi helpompaa. Tällä hetkellä virhepainallusten määrä on suuri.



Kuva 10. Lokihistoriasivu. Yksi kirjaus on avattuna mutta se menee muiden alle eikä näy

Kaaviosivu (liite 18) oli optimoinnin puutteen vuoksi käytännössä käyttökelvoton (kuva 11). Sivun elementit menivät päällekkäin, jolloin osaa toiminnoista ei voinut edes käyttää. Kaaviosivulla ei myöskään ole kunnollista funktiota mobiilissa, joten sen voisi piilottaa sieltä kokonaan.

Kuva 11. Kaaviosivu mobiilinäkymässä

6 TULOKSET

Testauksessa ilmeni joitakin parantamisen kohteita, mutta mitään järjestelmää rikkovaa ei havaittu. Mobiilioptimointia ei ole vielä tehty, joten mobiilissa oli luonnollisesti hieman enemmän ongelmia. Piiskan suositeltu selain on Google Chrome, ja sillä järjestelmässä oli vähiten virheitä.

Isoin ongelma mikä järjestelmässä ilmeni, oli päiväkohtaisten tuntirajoitteiden puuttuminen. Järjestelmään voidaan siis kirjata monta päällekkäistä tuntilokia, tai yli vuorokauden määrän tunteja yhteen päivään. Käytännössä tämä voi haitata tuntien laskemista lopuksi, jos käyttäjä esimerkiksi tallentaa

epähuomiossa saman päivän työtunnit kaksi kertaa, mutta ei enää myöhemmin muista oliko kyse tästä, vai ovatko tunnit väärässä päivässä. Järjestelmä ei myöskään ilmoittanut negatiivisista tunneista, tai jos työn ja taukojen erotus oli 0 tuntia.

Toinen huomattava ongelma oli, että projekteille on editointiominaisuus, mutta tehdyt muutokset eivät näytä tallentuvan. Esimerkiksi Mozilla Firefox mobiiliselaimella projektin luontiin liittyi ongelmia osan tietojen lisäämisen kanssa, joten toimivan editointimahdollisuuden puute on kohtalainen haitta.

Mobiiliselaimissa merkittävä virhe oli, että sivuilla taaksepäin meno päästi takasin järjestelmään uloskirjautumisen jälkeen, mikä ei ole turvallisuuden kannalta hyvä asia. Järjestelmän kello olisi myös hyvä saada toimimaan suomen aikaa, jolloin projektien luontipäivät näkyisivät oikein ja tuntikirjaukset onnistuisivat ilman ongelmia vuorokaudenajasta riippumatta. ”Charts”- eli kaaviosivu oli optimoinnin puutteen vuoksi kaikista ongelmaisin, mutta koska se ei ole järjestelmän käytettävyyden kannalta merkittävä sivu, ei sitä nosteta tässä yhteenvedossa erityisesti esille.

Tietokoneen selainta käytettäessä ongelmat olivat käytettävyyden kannalta kohtalaisen pieniä ja vaativat vain hieman optimointia. Esimerkiksi lokiluettelon tiedot olisi hyvä saada aukeamaan aina kun hiiren vie tuntikirjauksen päälle, ettei käyttäjän tarvitse yrittää montaa kertaa. Myös osioiden kokoa ja sijoittelua voisi hioa. Tämä vaatisi kuitenkin laajemman testaajajoukon, jonka avulla löydettäisiin paras keskiarvo optimoinnin suhteen.

Järjestelmä pitäisi myös optimoida mobiilille, jotta sitä pystyisi käyttämään, vaikka tietokone ei ole lähistöllä. Nyt esimerkiksi jotkin toiminnot, kuten tuntien lähetys hyväksyttävästi tai projektin päätösilmoitus, eivät ole lainkaan saatavilla selaimien mobiiliversioissa. Sivut skaalautuvat myös huonosti, mikä laskee käyttäjäkokemusta, jos mobiiliversiota tarvitsee tällä hetkellä käyttää.

7 YHTEENVETO

Testaus on tärkeä osa koko ohjelmistokehitysprosessia. Testaus sisältää erilaisia vaiheita ja menetelmiä, jotka testaajan on hyvä hallita.

Testisuunnitelman teon tulisi olla ensimmäisiä asioita uudessa projektissa. Näin voidaan välttää suurempia ongelmia lopussa, ja varmistaa että ohjelma toimii vakaasti ja halutulla tavalla.

Käytettävyystestaus on tärkeä testauksen alue, jolla optimoidaan ohjelma loppukäyttäjiä varten. Monet asiat voivat vaikuttaa suunnittelupöydällä hyviltä ideoilta, mutta osoittautuvat vasta käytössä epäkelvollisiksi. Tästä syystä myös uudet lisäykset ja ominaisuudet olisi hyvä testata ennen käyttöönottoa, vaikka ne vaikuttaisivatkin sopivan ohjelmaan, joka on jo hyväksytty käyttöön.

Tässä testauksessa tuli esiin sekä järjestelmän toimintaan että sen käytettävyyteen liittyviä eritasoisia ongelmia. Eliminoimalla ongelmakohdat mahdollisimman laajasti saadaan järjestelmästä vakaampi ja käyttäjäystävällisempi. Vähänkään hankalasti käytettävä ohjelma ei houkuttele ketään sitä käyttämään, joten testaukseen ja käytettävyyteen kannattaa ohjelmistoprojekteissa panostaa.

LÄHTEET

Homès, B. 2012. Fundamentals of Software Testing. Lontoo: ISTE ltd, Hoboken: John Wiley & Sons inc.

Kasurinen, J. 2013. Ohjelmistotestauksen käsikirja. Jyväskylä: Docendo.

Katara, M., Vuori, M. & Jääskeläinen A. 2015. Ohjelmistojen testaus. Tampereen teknillinen yliopisto. PDF-dokumentti. Saatavissa: http://www.cs.tut.fi/~tie21201/s2015/luennot/TIE-21204_2015.pdf [viitattu 26.4.2021].

Kinsbruner, E. 2020. Test Reporting: What It Is & How to Make It Work for Continuous Testing. Perfecto. WWW-dokumentti. Saatavissa: https://www.perfecto.io/blog/test-reporting [viitattu 17.5.2021].

Lee, J. 2017. White-Box Testing: Pros and Cons. Segue Technologies. WWW-dokumentti. Saatavissa: https://www.seguetech.com/white-box-testing-pros-cons/ [viitattu 18.5.2021].

Levy, L. & Novak J. 2010. Game development essentials: Game QA & testing. Clifton Park: Delmar.

Lyytinen, P. 2019. Mitä ohjelmistotestaus on ja mihin se pystyy. VALA Group. WWW-dokumentti. Saatavissa: https://www.valagroup.com/fi/2019/10/mita-ohjelmistotestaus-on-ja-mihin-se-pystyy/ [viitattu 26.4.2021].

Moolya Testing. 2020. Note-taking as a Skill for Testers. Moolya. Blogi. Saatavissa: https://moolya.com/blog/life-of-testers/notetaking_as_a_skill/ [viitattu 17.5.2021].

Moore, W. 2015. What is Black Box Testing: Advantages and Disadvantages. Invensis. WWW-dokumentti. Saatavissa: https://www.invensis.net/blog/it/black-box-testing-advantages-disadvantages/ [viitattu 18.5.2021].

Nielsen, J. 2012. Usability 101: Introduction to Usability. Nielsen Norman Group. WWW-dokumentti. Saatavissa: https://www.nngroup.com/articles/usability-101-introduction-to-usability/ [viitattu 9.5.2021].

Pesonen, T. s.a. Ohjelmistotestaus ja laadunvarmistus. VALA Group. WWW-dokumentti. Saatavissa: https://www.valagroup.com/fi/palvelut/ohjelmistotestaus-ja-laadunvarmistus/ [viitattu 26.4.2021].

Schultz, C. & Bryant, R. 2012. Game testing: All in one. 2. painos. Dulles: Mercury learning and information.

Software Testing Fundamentals. 2020. Test Case. WWW-dokumentti. Saatavissa: https://softwaretestingfundamentals.com/test-case/ [viitattu 15.5.2021].

Testbytes. 2018. Game Testing Tutorial: All The Information is Here! Blogi. Saatavissa: https://www.testbytes.net/blog/game-testing-tutorial/ [viitattu 18.5.2021].

TESTITAPAUS KIRJAUTUMISSIVU

Testitapaus ID	Kuvaus	Odotettu tulos
TC_Login-page_Usability_01	Testataan kirjautumissivun toimivuus ja sen yleinen käytettävyys.	Kirjautumissivu toimii ja se on käyttäjäystävällinen.

Test steps	
1	Avataan piiska.gamelab.fi sivu
2	Tarkistetaan että kaikki elementit näkyvät sivulla kuten kuuluu
3	Arvioidaan opittavuus.
4	Arvioidaan tehokkuus.
5	Arvioidaan muistettavuus.
6	Arvioidaan virheettömyys.
7	Arvioidaan tyytyväisyys.
8	
9	
10	

Päiväys
13.5.2021

Järjestelmäversio
1.0.3

Testaaja
Sirja R.

Käyttöjärjestelmä	Selain	Pass/Fail	Testauksen tulos
Windows 10	Google Chrome	Pass	"Feedback" painike on piilotettu alareunaan. Tiedot mahtuisivat ruudulle mutta nyt sivu on venynyt niin että rullaus aktivoituu. Kirjautuminen voisi olla keskellä niin sivulle ei jäisi jättikokoista mustaa aukkoa. Sivulla voisi olla linkki/ohjeet jos kirjautuminen ei onnistu. Virheelliset kirjautumistiedot eivät anna mitään palautetta (turvallista mutta ei käyttäjäystävällistä)
Windows 10	Mozilla Firefox	Pass	Ks. yllä
Android 9	Google Chrome	Pass	”Login”-osio on pieni muuhun verrattuna
Android 9	Mozilla Firefox	Pass	”Disclaimer” on todella isolla fontilla

TESTITAPAUS KIRJAUTUMINEN

Testitapaus ID	Kuvaus	Odotettu tulos
TC_Login_02	Kirjaudutaan järjestelmään ja kirjaudutaan ulos järjestelmästä	Järjestelmään pääsee kirjautumaan sisään ja ulos.

Test steps	
1	Kirjoitetaan käyttäjätunnus ja salasana niille varattuihin kenttiin.
2	Painetaan "login"
3	Painetaan "sign out"
4	Varmistetaan että selaimen taaksepäin nuolen käyttäminen ei päästä takaisin järjestelmään.
5	
6	
7	
8	
9	
10	

Päiväys
14.5.2021

Järjestelmäversio
1.0.3

Testaaja
Sirja R.

Käyttöjärjestelmä	Selain	Pass/Fail	Testauksen tulos / Kommentit
Windows 10	Google Chrome	Pass	ok
Windows 10	Mozilla Firefox	Pass	ok
Android 9	Google Chrome	Fail	Taaksepäin meno päästää takaisin järjestelmään. Myös alareunan pikalinkit päästävät takaisin sisälle uloskirjautumisen jälkeen.
Android 9	Mozilla Firefox	Fail	Ks. yllä

TESTITAPAUS OHJEKIRJA

Testitapaus ID	Kuvaus	Odotettu tulos
TC_User-manual_03	Testataan, toimiiko user manual eli ohjekirja.	Ohjekirja aukeaa ja sen pystyy lukemaan kokonaan.

Test steps	
1	Painetaan sivun alalaidasta "UPDATED version 1.0.3 User manual available!!!" linkkiä.
2	Selataan ohje läpi.
3	
4	
5	
6	
7	
8	
9	
10	

Päiväys
12.5.2021

Järjestelmäversio
1.0.3

Testaaja
Sirja R.

Käyttöjärjestelmä	Selain	Pass/Fail	Testauksen tulos / Kommentti
Windows 10	Google Chrome	Pass	ok
Windows 10	Mozilla Firefox	Pass	ok
Android 9	Google Chrome	Fail	Ohjekirjaa ei saatavilla mobiilissa.
Android 9	Mozilla Firefox	Fail	Ohjekirjaa ei saatavilla mobiilissa.

TESTITAPAUS PALAUTE

Testitapaus ID	Kuvaus	Odotettu tulos
TC_Feedback_04	Testataan toimiiko feedback/palaute.	Palaute-linkki toimii ja palautteen saa kirjoitettua ja lähetettyä.

Test steps	
1	Painetaan sivun alalaidasta puhekuplan/kynän kuvaa.
2	Valitaan palautetyyppi.
3	Kirjoitetaan palaute.
4	Painetaan "send".
5	Palataan etusivulle.
6	
7	
8	
9	
10	

Päiväys
15.5.2021

Järjestelmäversio
1.0.3

Testaaja
Sirja R.

Käyttöjärjestelmä	Selain	Pass/Fail	Testauksen tulos / Kommentti
Windows 10	Google Chrome	Pass	Etusivulle ei ole suoraa paluupainiketta, vaan siirtyminen pitää tehdä selaimen "edellinen sivu" toiminnon kautta. Tyhjä kommenttikenttä ei anna virheilmoitusta
Windows 10	Mozilla Firefox	Pass	Ks. yllä. Tekstikentän ympärys muuttuu punaiseksi lähetyksen jälkeen mikä antaa kuvan, että jotain meni vikaan. Tyhjä kommenttikenttä ei anna virheilmoitusta
Android 9	Google Chrome	Pass	Etusivulle ei paluulinkkiä. Tyhjä kommenttikenttä ei anna virheilmoitusta
Android 9	Mozilla Firefox	Pass	Etusivulle ei paluulinkkiä. Tyhjä kommenttikenttä ei anna virheilmoitusta

TESTITAPAUS SIVUT

Testitapaus ID	Kuvaus	Odotettu tulos
TC_Pages_05	Testataan kaikkien sivujen avautuminen.	Kaikki sivut avautuvat ilman virheitä, kun niiden nimeä painetaan.

Test steps	
1	Kirjaudutaan järjestelmään.
2	Jos on olemassa projekti, järjestelmä avautuu "Log" sivulle. Tyhjä profiili avautuu "Projects" sivulle.
3	Avataan "Projects" sivu.
4	Avataan "Log" sivu.
5	Avataan "Log History" sivu.
6	Avataan "Charts" sivu.
7	
8	
9	
10	

Päiväys
11.5.2021

Järjestelmäversio
1.0.3

Testaaja
Sirja R.

Muu lisäys
Mobiilivalikon sivujärjestys eroaa tietokoneesta. Projects voisi olla ensimmäisenä ja Charts viimeisenä. Sign Out voisi olla alempana erikseen, ettei siitä paina vahingossa.

Käyttöjärjestelmä	Selain	Pass/Fail	Testauksen tulos / Kommentti
Windows 10	Google Chrome	Pass	ok
Windows 10	Mozilla Firefox	Pass	ok
Android 9	Google Chrome	Pass	Charts sivun asettelu pahasti päällekkäin, piilota header/footer ja slideshow painikkeet liian pienet.
Android 9	Mozilla Firefox	Pass	Charts sivun asettelu pahasti päällekkäin, piilota header/footer ja slideshow painikkeet liian pienet.

TESTITAPAUS PROJEKTIN LUONTI

Testitapaus ID	Kuvaus	Odotettu tulos
TC_Projects_06	Testataan projektin luominen.	Projektin saa luotua ja se tallentuu.

Test steps	
1	Siirrytään Projects-sivulle.
2	Valitaan "Create New".
3	Täytetään projektin tiedot.
4	Painetaan "Submit".
5	Tarkistetaan että projekti ilmestyy "Your Projects" osioon.
6	
7	
8	
9	
10	

Päiväys
16.5.2021

Järjestelmäversio
1.0.3

Testaaja
Sirja R.

Käyttöjärjestelmä	Selain	Pass/Fail	Testauksen tulos / Kommentti
Windows 10	Google Chrome	Pass	Päiväys ei mene Suomen kelloajan mukaan. 16.5 klo 2:30 luotu projekti näytetään 15.5 luotuna.
Windows 10	Mozilla Firefox	Pass	ks. yllä
Android 9	Google Chrome	Pass	ks. yllä
Android 9	Mozilla Firefox	Pass	"Who is this for", "Tags" ja "Project member" osioiden valintalistaa ei ilmesty. Tagejen lisäys ei onnistu tästä syystä. +ks. Yllä.

TESTITAPAUS PROJEKTIN MUOKKAUS

Testitapaus ID	Kuvaus	Odotettu tulos
TC_Projects_Edit_07	Muokataan talennettua projektia.	Projektia voi muokata ja muokkaus tallentuu.

Test steps	
1	Siirrytään Projects-sivulle.
2	Valitaan "Your Projects" osiosta tallennettu projekti.
3	Painetaan kynän kuvaa, joka aktivoi muokkauksen.
4	Lisätään/poistetaan tietoja.
5	Painetaan "Update".
6	Tarkistetaan että projektin tiedot päivittyivät.
7	
8	
9	
10	

Päiväys
15.5.2021

Järjestelmäversio
1.0.3

Testaaja
Sirja R.

Käyttöjärjestelmä	Selain	Pass/Fail	Testauksen tulos / Kommentti
Windows 10	Google Chrome	Fail	Tiedot eivät päivittyneet, tai ainakaan tallennuksesta ei tullut tietoa, jos muutokset vaativat opettajan hyväksynnän ennen näkymistä.
Windows 10	Mozilla Firefox	Fail	Ks. yllä. Avautuva muokkausikkuna menee turhaan scroll-palkeille.
Android 9	Google Chrome	Fail	Ks. yllä^^ Kynän kuva mobiilisti liian pieni.
Android 9	Mozilla Firefox	Fail	ks. yllä^^^ Kynän kuva mobiilisti liian pieni.

TESTITAPAUS TYÖTUNTIEN KIRJAUS

Testitapaus ID	Kuvaus	Odotettu tulos
TC_Log_08	Kirjataan työtunteja.	Työtunteja voi kirjata tallennettuun ja hyväksyttyyn projektiin.

Test steps	
1	Käyttäjällä pitää olla hyväksytty aktiivinen projekti.
2	Siirrytään "Log"-sivulle.
3	Valitaan projekti.
4	Annetaan päivä, aika, tauot ja kuvaus.
5	Painetaan "+Add" ja lisätään toinen tuntikirjaus vaiheen 4 tavoin.
6	Painetaan "Submit".
7	Tarkistetaan "Log History" sivulta että tunnit kirjautuivat.
8	
9	
10	

Päiväys
13.5.2021

Järjestelmäversio
1.0.3

Testaaja
Sirja R.

Muu lisäys
Järjestelmä ei anna kirjata tunteja heti vuorokauden vaihduttua. Ilmeisesti aika on GMT eikä Suomen GMT+2 (+3 kesällä).

Käyttöjärjestelmä	Selain	Pass/Fail	Testauksen tulos / Kommentti
Windows 10	Google Chrome	Pass	Kuvauksen kirjoittamisen jälkeen pitää klikata kentän ulkopuolelta, jotta ”submit” nappi aktivoituu.
Windows 10	Mozilla Firefox	Pass	ok
Android 9	Google Chrome	Pass	ok
Android 9	Mozilla Firefox	Pass	ok

TESTITAPAUS PROJEKTIIN LIITTYMINEN

Testitapaus ID	Kuvaus	Odotettu tulos
TC_Group_Projects_09	Liitytään projektiin.	Projektiin, jossa on tilaa voi liittyä. Samaan projektiin ei voi liittyä useaan kertaan.

Test steps	
1	Siirrytään Projects-sivulle.
2	Valitaan "All Projects".
3	Painetaan yhden projektin kohdalla "Apply".
4	Kirjoitetaan avautuvaan ikkunaan viesti ja painetaan "Send".
5	Yritetään liittyä samaan proketiin uudelleen, josta pitäsi tulla ilmoitus, ettei se onnistu.
6	
7	
8	
9	
10	

Päiväys
15.5.2021

Järjestelmäversio
1.0.3

Testaaja
Sirja R.

Käyttöjärjestelmä	Selain	Pass/Fail	Testauksen tulos / Kommentti
Windows 10	Google Chrome	Pass	ok
Windows 10	Mozilla Firefox	Pass	ok
Android 9	Google Chrome	Pass	"Apply" painikeet pitää erikseen scrollata näkyviin oikealta.
Android 9	Mozilla Firefox	Pass	"Apply" painikeet pitää erikseen scrollata näkyviin oikealta.

TESTITAPAUS KÄYTTÄJÄ

Testitapaus ID	Kuvaus	Odotettu tulos
TC_User_10	Testataan käyttäjä-ikkunan toimivuus.	Klikatessa käyttäjän profiili aukeaa ja näkyy oikein.

Test steps	
1	Painetaan käyttäjän nimeä.
2	Tarkastetaan että profiili aukeaa oikeilla tiedoilla ja näkyy kokonaan
3	Mobiilissa avataan oikean yläkulman pudotusvalikko
4	
5	
6	
7	
8	
9	
10	

Päiväys
15.5.2021

Järjestelmäversio
1.0.3

Testaaja
Sirja R.

Käyttöjärjestelmä	Selain	Pass/Fail	Testauksen tulos / Kommentti
Windows 10	Google Chrome	Pass	ok
Windows 10	Mozilla Firefox	Pass	ok
Android 9	Google Chrome	Pass	ok
Android 9	Mozilla Firefox	Pass	ok

TESTITAPAUS TEHTÄVÄLISTA

Testitapaus ID	Kuvaus	Odotettu tulos
TC_To_Do_11	Testataan "To Do"-listan toimivuus.	Listaan voi lisätä ja siitä voi poistaa tehtäviä.

Test steps	
1	Klikataan "Log" sivun oikeassa reunassa olevan "To Do" listan "+"Add" painiketta.
2	Kirjoitetaan tiedot.
3	
4	
5	
6	
7	
8	
9	
10	

Päiväys
15.5.2021

Järjestelmäversio
1.0.3

Testaaja
Sirja R.

Käyttöjärjestelmä	Selain	Pass/Fail	Testauksen tulos / Kommentti
Windows 10	Google Chrome	Fail	Listaa ei saanut näkyviin.
Windows 10	Mozilla Firefox	Fail	ks. yllä
Android 9	Google Chrome	Fail	ks. yllä
Android 9	Mozilla Firefox	Fail	ks. yllä

TESTITAPAUS KAAVIOSIVU

Testitapaus ID	Kuvaus	Odotettu tulos
TC_Charts_Filter_12	Testataan charts-sivun suodatustoiminnot.	Suodatuskriteerit toimivat oikein ja näyttävät halutut tiedot.

Test steps	
1	Siirrytään "Charts" sivulle.
2	Valitaan suodattimien "Projects" kohdasta projekti.
3	Valitaan kriteereiksi "Total", "Include Self" ja "Include Progress".
4	Valitaan aikaväli.
5	Valitaan vuodeksi meneillään oleva vuosi.
6	Testataan sekä "Hours" että "Study Points" käyttäen aikavälejä "Weekly", "Daily" ja "Weekday".
7	Testataan ensin viivakaaviona (Line) ja sitten palkeilla (Bar).
8	
9	
10	

Päiväys
15.5.2021

Järjestelmäversio
1.0.3

Testaaja
Sirja R.

Käyttöjärjestelmä	Selain	Pass/Fail	Testauksen tulos / Kommentti
Windows 10	Google Chrome	Pass	ok
Windows 10	Mozilla Firefox	Pass	ok
Android 9	Google Chrome	Fail	Sivun elementit skaalautuvat toistensa päälle. "Include self" valinta on taulukon alla eikä sitä voi muuttaa.
Android 9	Mozilla Firefox	Fail	Sivun elementit skaalautuvat toistensa päälle. "Include self" valinta on taulukon alla eikä sitä voi muuttaa.

TESTITAPAUS TUNTIEN LÄHETYS

Testitapaus ID	Kuvaus	Odotettu tulos
TC_Send_Log_13	Läheteään tunnit kirjattavaksi.	Lähetys onnistuu.

Test steps	
1	Klikataan käyttäjän nimen kohdalla olevaa huutomerkkiä.
2	Aukeavasta osiosta valitaan "Send report notice".
3	Vahvistus-ikkunasta valitaan "ok".
4	
5	
6	
7	
8	
9	
10	

Päiväys
15.5.2021

Järjestelmäversio
1.0.3

Testaaja
Sirja R.

Käyttöjärjestelmä	Selain	Pass/Fail	Testauksen tulos / Kommentti
Windows 10	Google Chrome	Pass	ok
Windows 10	Mozilla Firefox	Pass	ok
Android 9	Google Chrome	Fail	Painiketta ei löydy.
Android 9	Mozilla Firefox	Fail	Painiketta ei löydy.

TESTITAPAUS PROJEKTIN LOPETUS

Testitapaus ID	Kuvaus	Odotettu tulos
TC_Finish_Project_14	Merkitään projekti valmistuneeksi.	Pyynnön lähetys onnistuu.

Test steps	
1	Siirrytään "Projects" sivulle.
2	Valitaan "Your Projects" kohdasta haluttu projekti.
3	Aukeavan osion oikeasta alalaidasta valitaan "Send notice for" ja "Completion".
4	
5	
6	
7	
8	
9	
10	

Päiväys
15.5.2021

Järjestelmäversio
1.0.3

Testaaja
Sirja R.

Käyttöjärjestelmä	Selain	Pass/Fail	Testauksen tulos / Kommentti
Windows 10	Google Chrome	Pass	ok
Windows 10	Mozilla Firefox	Pass	ok
Android 9	Google Chrome	Fail	Painiketta ei löydy.
Android 9	Mozilla Firefox	Fail	Painiketta ei löydy.

TESTITAPAUS PROJEKTISIVUN KÄYTETTÄVYYS

Testitapaus ID	Kuvaus	Odotettu tulos
TC_Projets_Page_Usability_15	Testataan Projects sivun käytettävyys.	Sivu on helppo ymmärtää ja tarvittavat toiminnot löytää.

Test steps	
1	Avataan “Projects” sivu.
2	Arvioidaan opittavuus.
3	Arvioidaan tehokkuus.
4	Arvioidaan muistettavuus.
5	Arvioidaan virheettömyys.
6	Arvioidaan tyytyväisyys.
7	
8	
9	
10	

Päiväys
15.5.2021

Järjestelmäversio
1.0.3

Testaaja
Sirja R.

Käyttöjärjestelmä	Selain	Pass/Fail	Testauksen tulos / Kommentti
Windows 10	Google Chrome	Pass	Ensivaikutelma jokseenkin sekava, ei houkuttele opettelemaan käyttöä. Tutustumisen jälkeen kuitenkin helppo oppia ja käyttää.
Windows 10	Mozilla Firefox	Pass	ks yllä
Android 9	Google Chrome	Pass	Asettelu parempi mobiilissa kuin tietokoneella.
Android 9	Mozilla Firefox	Pass	Asettelu parempi mobiilissa kuin tietokoneella. Valinta-valikot eivät toimi, esim "Tags" kohtaa ei voi täydentää.

TESTITAPAUS LOKISIVUN KÄYTETTÄVYYS

Testitapaus ID	Kuvaus	Odotettu tulos
TC_Log_Page_Usability_16	Testataan Log sivun käytettävyys.	Sivu on helppo ymmärtää ja tarvittavat toiminnot löytävät.

Test steps	
1	Avataan “Log” sivu.
2	Arvioidaan opittavuus.
3	Arvioidaan tehokkuus.
4	Arvioidaan muistettavuus.
5	Arvioidaan virheettömyys.
6	Arvioidaan tyytyväisyys.
7	
8	
9	
10	

Päiväys
15.5.2021

Järjestelmäversio
1.0.3

Testaaja
Sirja R.

Käyttöjärjestelmä	Selain	Pass/Fail	Testauksen tulos / Kommentti
Windows 10	Google Chrome	Pass	Kello- ja kalenterikuvat ovat liian tummia päivämäärän, alku- ja lopuajan kohdalla. Projektin valintakohta on hieman huomaamaton sivun ylälaidassa, mahdollista laittaaa virheellisesti väärään, jolloin joutuu korjaamaan. Aikateidoissa käytetty tekstin varjostus rasittaa silmiä.
Windows 10	Mozilla Firefox	Pass	Projektin valintakohta on hieman huomaamaton sivun ylälaidassa, mahdollista laittaaa virheellisesti väärään, jolloin joutuu korjaamaan. Aikateidoissa käytetty tekstin varjostus rasittaa silmiä.
Android 9	Google Chrome	Pass	Projektin valinta tässäkin helppo ohittaa.
Android 9	Mozilla Firefox	Pass	Projektin valinta tässäkin helppo ohittaa.

TESTITAPAUS LOKIHISTORIASIVUN KÄYTETTÄVYYS

Testitapaus ID	Kuvaus	Odotettu tulos
TC_Log_History_Page_ Usability_17	Testataan ”Log History”-sivun käytettävyys.	Sivu on helppo ymmärtää ja tarvittavat toiminnot löytää.

Test steps	
1	Avataan “Log History” sivu.
2	Arvioidaan opittavuus.
3	Arvioidaan tehokkuus.
4	Arvioidaan muistettavuus.
5	Arvioidaan virheettömyys.
6	Arvioidaan tyytyväisyys.
7	

Päiväys
15.5.2021

Järjestelmäversio
1.0.3

Testaaja
Sirja R.

Käyttöjärjestelmä	Selain	Pass/Fail	Testauksen tulos / Kommentti
Windows 10	Google Chrome	Pass	Lisäyksien laajennettu näkymä ei toimi aina kun hiiren vie lisäyksen päälle, turhauttavaa jos haluamaansa kohtaa ei saa auki. Näyttää tapahtuvan, jos hiirtä liikuttaa useassa kohtaa ennen pysähtymistä. Muokkaustilanteessa kynän kuvan lisäksi esim. alueen reunat voisivat olla eri väriset, jolloin huomaa, jos on vahingossa aktivoinut muokkaustilan. Kynän kohdalla voisi näkyä tooltip "edit" kun hiiren vie sen päälle. Projektin nimeä klikkaamalla avautuu myös projektinäkymä oikealle, mutta jos muokkaus on päällä, samalla avautuu myös tuntien muokkaus. Menevät osittain päällekkäin*.
Windows 10	Mozilla Firefox	Pass	ks yllä. *Firefoxissa muokkausikkuna ja projektitiedot eivät mene päällekkäin.
Android 9	Google Chrome	Pass	Sisältökategoriat eivät skaalaudu kenttien mukaan, vaan ovat vierekkäin vasemmassa reunassa. Lisäyksien tietoja ei voi laajentaa (menevät listan alle).
Android 9	Mozilla Firefox	Pass	Sisältökategoriat eivät skaalaudu kenttien mukaan, vaan ovat vierekkäin vasemmassa reunassa. Lisäyksien tietoja ei voi laajentaa (menevät listan alle).

TESTITAPAUS KAAVIOSIVUN KÄYTETTÄVYYS

Testitapaus ID	Kuvaus	Odotettu tulos
TC_Charts_Page_Usability_18	Testataan “Charts”-sivun käytettävyys.	Sivu on helppo ymmärtää ja tarvittavat toiminnot löytää.

Test steps	
1	Avataan “Charts”-sivu.
2	Arvioidaan opittavuus.
3	Arvioidaan tehokkuus.
4	Arvioidaan muistettavuus.
5	Arvioidaan virheettömyys.
6	Arvioidaan tyytyväisyys.
7	

Päiväys
15.5.2021
Järjestelmäversio
1.0.3

Testaaja
Sirja R.

Käyttöjärjestelmä	Selain	Pass/Fail	Testauksen tulos / Kommentti
Windows 10	Google Chrome	Pass	Sekalaisin sivu, vaatii perehtymistä ja ohjekirjan. Tehdyt kaaviot jäävät muistiin väliaikaisesti, mutta niitä ei saa poistettua, ladattaessa CSV:tä ladataan kaikki tehdyt kaaviot oli niitä 1 tai 30. Jokaisen lataus pitää erikseen hyväksyä tai hylätä. Kaavioiden nimeäminen on hankalaa koska nimikohta deaktivoituu ensimmäisen kirjaimen jälkeen, jos haluaa lisää pitää nimikohta aktivoida uudelleen. Kaavion nimi voisi siirtyä ladattavan tiedoston nimeksi. Sivun "slideshow" ja "piilota header/footer" painikkeet ovat epäselvät ilman ohjetta ja kaipaisivat esim. hoover tooltipin.
Windows 10	Mozilla Firefox	Pass	ks yllä
Android 9	Google Chrome	Fail	Mobiililla lähes käyttökelvoton skaalausten ja sijoittelun vuoksi. Filttereiden pitäisi olla ensin päällä ja kaavio sitten alla. Nyt ne menevät vierekkäin kuten tietokoneella.
Android 9	Mozilla Firefox	Fail	ks yllä

TESTITAPAUS TUNTIKIRJAUKSEN RAJAT

Testitapaus ID	Kuvaus	Odotettu tulos
TC_Log_Limits_19	Testataan, onko tuntikirjaukseen asetettu päivittäiset ylä- ja alarajat.	Samalle päivälle ei voi kirjata yli 24 h verran tunteja. Lisäyksen pitää olla vähintään 1min.

Test steps	
1	Siirrytään "Log" sivulle.
2	Annetaan "+Add" painiketta käyttäen samalla kertaa usea merkintä samalle päivälle ja painetaan "Submit".
3	Annetaan seuraavaksi samalle päivälle päällekkäisiä tunteja erillisinä lisäyksinä. Yhden merkinnän jälkeen painetaan aina "Submit" (ei käytetä +add).
4	Annetaan tuntikirjaus, jossa työmäärän ja taukojen erotus on negatiivinen.
5	Tarkistetaan tallennetut tiedot "Log History"-sivulta.
6	
7	
8	
9	

Päiväys
15.5.2021

Järjestelmäversio
1.0.3

Testaaja
Sirja R.

Käyttöjärjestelmä	Selain	Pass/Fail	Testauksen tulos / Kommentti
Windows 10	Google Chrome	Fail	Järjestelmä hyväksyy monta erillistä kirjausta samalle päivälle ja sallii yli 24 tuntia. Negatiiviset tunnit hyväksytään ja muutetaan muotoon 0h.
Windows 10	Mozilla Firefox	Fail	ks. yllä
Android 9	Google Chrome	Fail	ks. yllä
Android 9	Mozilla Firefox	Fail	ks. yllä

TESTITAPAUS LOKIN MUOKKAUS

Testitapaus ID	Kuvaus	Odotettu tulos
TC_Log_Edit_20	Muokataan kirjattuja tunteja.	Kirjattuja tunteja voi muokata.

Test steps	
1	Painetaan "Log History" sivulla kynän kuvaa ja valitaan haluttu tuntikirjaus.
2	Annetaan uudet tiedot ja painetaan "Update".
3	Tarkistetaan että uudet tiedot tallentuivat.
4	
5	
6	
7	
8	
9	
10	

Päiväys
15.5.2021

Järjestelmäversio
1.0.3

Testaaja
Sirja R.

Käyttöjärjestelmä	Selain	Pass/Fail	Testauksen tulos / Kommentti
Windows 10	Google Chrome	Pass	"Update" painike aktivoituu vasta kun tekstikentän ulkopuolelta klikkaa.
Windows 10	Mozilla Firefox	Pass	ok
Android 9	Google Chrome	Pass	Muokkausikkuna skaalattu huonosti, joutuu scrollaamaan sivuttain. Muokkauspainike liian pieni.
Android 9	Mozilla Firefox	Pass	ks. yllä