Tampereen ammattikorkeakoulu

Mobiilipelin saattaminen soft launchiin pelitestauksen näkökulmasta

Laura Huovinen

OPINNÄYTETYÖ
Marraskuu 2022

Tietojenkäsittelyn tutkinto-ohjelma
Game Production

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma
Game Production

HUOVINEN, LAURA:
Mobiilipelin saattaminen soft launchiin pelitestauksen näkökulmasta

Opinnäytetyö 35 sivua
Marraskuu 2022

Opinnäytetyön tarkoituksena oli kerätä tietoa erilaisista soft launchiin ja pelitestaamiseen liittyvistä käytänteistä ja prosesseista sekä verrata niitä Traplight Oy:n prosesseihin. Työn tavoitteena oli tutustua soft launch -ja pelitestausprosesseihin ja luoda ehdotuksia, joilla yrityksen nykyisiä prosesseja voitaisiin parantaa. Toisena tavoitteena oli luoda pohja, jota myös työn ulkopuoliset henkilöt voisivat hyödyntää prosessiensa suunnittelussa ja kehittämisessä.

Opinnäytetyössä käsitellään pintapuolisesti soft launch -prosessi ja siihen liittyvät valmistelut. Tarkemmin työssä esitellään pelitestausprosessia ja siihen liittyviä käytänteitä ja pelitestaustapoja. Lukija saa peruskäsityksen siitä, mitä kumpaankin prosessiin kuuluu, vaikkei hänellä olisi aikaisempaa tietoa aiheesta.

Työssä kehitettiin toimeksiantajan pelitestausprosessiin parannusehdotuksia, joista kaksi hyväksyttiin. Toinen otettiin tarkemman suunnittelun jälkeen välittömästi käyttöön ja toinen vietiin eteenpäin myöhemmin käsiteltäväksi. Käyttöönotettu muutos helpotti toimeksiantajan pelitestausprosessia tehden bugien ilmoittamisesta selkeämpää ja tehokkaampaa.

Asiasanat: soft launch, pelitestaaminen, mobiilipeli

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in Business Information Systems
Game Production

HUOVINEN, LAURA:
Preparing a Mobile Game for Soft Launch from Game Testing Perspective

Bachelor's thesis 35 pages
November 2022

The purpose of this thesis was to gather information about soft launching a mobile game and game testing processes and compare them with Traplight Ltd's processes. The results and findings were used to create improvement suggestions to further develop the company's processes. The objective was to collect information regarding soft launching and game testing, which could be used by anyone in planning and developing their processes.

The theoretical section explores the basics of soft launching and game testing, delving deeper into different game testing methods and processes. In practical section, the company's game testing process was analysed and suggestions of improvement were made based on the findings.

As a result, methods to develop the company's game testing processes were found. One was deployed immediately to improve the manner bugs were reported and the other was recorded for later processing. The deployed change made reporting bugs clearer and more efficient.

Key words: soft launch, game testing, mobile game

SISÄLLYS

LYHENTEET JA TERMIT

ASO	App Store Optimization, sovelluskauppaoptimointi
Build	Koottu sovellus, jota voidaan käyttää laitteilla
KPI	Key Performance Indicator, suorituskykymittari. Mitattavia arvoja, joiden avulla arvioidaan kuinka tehokkaasti sovellus saavuttaa asetetut tavoitteet
QA	Quality assurance, laadunvarmistus
Retentio	Asiakaspysyvyys

1 JOHDANTO

Opinnäytetyön tarkoituksena oli kerätä tietoa soft launchista ja pelitestaamisesta ja verrata niitä Traplight Oy:n prosesseihin. Työn tavoitteena oli tutustua soft launch -ja pelitestausprosesseihin ja luoda parannusehdotuksia, joilla yrityksen nykyisiä prosesseja voitaisiin kehittää. Työn toisena tavoitteena oli luoda pohja, jota myös työn ulkopuoliset tahot voisivat hyödyntää omien prosessiensa suunnittelussa ja parantamisessa.

Työn teoriaosuudessa käydään pintapuolisesti läpi mobiilipelin soft launch -prosessi ja siihen liittyvät valmistelut. Tarkemmin teoriaosuudessa perehdytään pelitestaamiseen ja siihen liittyviin käytänteisiin ja prosesseihin, kuten esimerkiksi bugien ilmoittamiseen ja erilaisiin testaustapoihin. Tavoitteena oli kasvattaa ymmärrystä pelitestaamisesta ja sen roolista soft launchin yhteydessä.

Käytännön osuudessa keskityttiin kuvaamaan toimeksiantajan soft launch -ja pelitestausprosesseja ja esiteltiin erilaisia keinoja, joilla kyseisiä prosesseja voitaisiin parantaa. Työssä esitellään myös käyttöönotetut parannusehdotukset.

2 SOFT LAUNCH

2.1 Yleisesti soft launchista

Soft launch eli testijulkaisu on yleinen strategia, jossa tuote julkaistaan yleensä rajoitetulle käyttäjämäärälle ennen sen aikataulutettua markkinoille tuomista (IronSource n.d.). Tuote julkaistaan joko pienellä tai olemattomalla markkinoinnilla. Soft launchin tavoitteena on katsoa, miten potentiaaliset käyttäjät reagoivat tuotteeseen, löytää ja korjata bugeja sekä tarkistaa, löytyykö muita parannuskohteita. Yritys saattaa löytää parannettavaa esimerkiksi pelisuunnittelusta tai monetisaatiostrategioista. Soft launch antaa yrityksille myös tilaisuuden optimoida käyttäjien hankintastrategioitaan markkinoilla, jotka vastaavat kohdemaan markkinoita. (Knezovic 2022; IronSource n.d.)

Soft launchin avulla voidaan myös tarkastella sitä, onko peli kannattava. Jotta pelin uskaltaa julkaista kansainvälisillä markkinoilla, yrityksen pitäisi saada pelaajista enemmän tuloja kuin mitä pelaajien hankinta maksaa. Siltä varalta, että tämä ei toteudu, yrityksen tulisi määritellä missä menee raja sille, että projekti lopetetaan ja siirrytään seuraavaan. Soft launchista saatua dataa voidaan käyttää pelin korjaamiseen, mutta jos havaitaan ettei idea toimi lainkaan, yrityksen voi olla parempi sulkea projekti ennen kuin siihen käytetään enempää aikaa ja rahaa. (Knezovic 2022.)

Soft launchia saatetaan käyttää myös muiden tavoitteiden saavuttamiseksi. Erityisesti aloittelevat yritykset saattavat hyödyntää soft launchia luodakseen tuotteelleen ja yritykselleen näkyvyyttä, jolla voi kiinnittää niin potentiaalisten uusien työntekijöiden kuin mahdollisten sijoittajienkin huomion. Jos markkinoilla ei ole ennestään vastaavaa tuotetta, yritys saattaa testijulkaista tuotteensa määrittääkseen markkinat omilla ehdoillaan ja asettaakseen perustan sille, miksi heidän teknologiansa on ylivoimaista. (Hogan & Broadbent 2016.) Testijulkaisemalla tuotteensa he saavat myös tarpeeksi aikaisessa vaiheessa palautetta, jonka perusteella he voivat parannella tuotettaan ja varmistaa, että se todella täyttää markkinaraon (Lawley & Schure 2017).

2.2 Valmistelut

Ennen mobiilipelin testijulkaisua yrityksen täytyy perehtyä mobiilipelimarkkinoihin. Yrityksen on tärkeä tietää, mikä on sillä hetkellä suosittua, millaisia pelejä pelataan eniten ja onko olemassa markkinarakoa tietyn tyyppisille peleille. On hyvä myös perehtyä kilpailijoiden tekemisiin. Kannattaa esimerkiksi perehtyä siihen, millaisia ominaisuuksia kilpailijoiden peleissä on, miten ne on nimetty ja suunniteltu sekä millaisia monetisaatio- ja markkinointistrategioita he käyttävät. (Knezovic 2022.)

Yrityksen täytyy määrittää pelin kohdeyleisö. Hyvä lähtökohta on päättää yleisellä tasolla, millaiselle pelaajajoukolle peli on suunnattu. Rennommat pelaajat suosivat pelejä, jotka vaativat vain vähän taitoa ja joita voi pelata ajankuluksi, kun taas kokeneemmat pelaajat saattavat käyttää taitoja vaativiin peleihin tunteja päivässä. Nämä ovat ääripäitä, joiden väliin suurin osa pelaajista sijoittuu. Pelaajien toivotun tason lisäksi tulee huomioida myös kohdeyleisön ikä ja sukupuoli, koittaen välttää stereotypioita. (Lancaric 2018; Knezovic 2022.)

Peliä ei yleensä testijulkaista suoraan kohdemaan markkinoille. Yrityksen kannattaa valita testijullkaisua varten maa tai maita, joissa on kohdemarkkinoita vastaava kulttuuri. Jos yritys haluaa julkaista valmiin pelin esimerkiksi USAn markkinoilla, pelin voi testijulkaista Euroopan maissa kuten Hollannissa, Suomessa tai Ruotsissa. Jos yrityksellä on pienempi budjetti, niin yrityksen kannattaa testata peliä Israelissa, Turkissa tai Filippiineissä. (Knezovic 2022.) Maita pohtiessa kannattaa myös pitää mielessä pelin lokalisointi. Suositeltavaa olisi, että pelissä olisi mahdollisimman vähän tekstiä lokalisoinnin helpottamiseksi. Lokalisoinnissa voi käyttää ulkoistettuja lokalisointipalveluita. (McCann 2011.)

Valmistelujen aikana on tärkeää määrittää mitä testijulkaisulla halutaan testata. Tätä varten on hyvä määrittää KPIt eli suorituskykymittarit, jotta yritys tietää mihin mittareihin kiinnittää huomiota testijulkaisun aikana ja mikä on lopullinen tavoite. Yrityksen tulisi myös testata pelin luovia puolia, kuten pelisuunnittelua ja käyttöliittymää sekä sovelluskauppaoptimointia ja monetisaatio- ja markkinointistrategioita. Jos huomataan, ettei jokin toimikaan, sitä voidaan vielä muokata ja testata uudelleen ennen pääjulkaisua (Knezovic 2022.)

Soft launch tarjoaa hyvän tilaisuuden testata pelin markkinointistrategiaa, mikä usein ratkaisee, saako peli tarpeeksi pelaajia menestyäkseen vai ei. Testaus kannattaa toteuttaa A/B-testauksena, joka käytännössä tarkoittaa sitä, että testijulkaisun aikana käytetään kahta markkinointistrategiaa, joita lopuksi vertaillaan keskenään. Vertailevissa testeissä voi testata myös useampaa strategiaa yhtä aikaa. Tärkeimmät seurattavat mittarit näiden testien aikana ovat latausten määrät ja kuinka paljon yhden asennuksen saaminen kustantaa. Testien aikana voidaan vertailla myös muun muassa mainosten formaattia ja muotoilua, erilaisia mainosverkostoja sekä kohdistamisvaihtoehtoja. (Knezovic 2022.)

2.3 Hyödyt

Testijulkaisulla on tiettyjä etuja verrattuna hard launchiin eli valmiin tuotteen julkaisuun. Monelle yritykselle soft launch on kustannustehokkaampi lähestymistapa tuotteen julkaisuun, joka antaa enemmän aikaa tuotteen hiontaan ja käytännön kokemuksen kerryttämiseen paljastamalla sen rajatulle yleisölle ja oikean maailman skenaarioille. Soft launch auttaa yhtiöitä myös kerryttämään dataa oikeassa ajassa heidän tuotteestaan ja palveluistaan, antaen hyvän perustuksen kehityksen jatkamiselle. (IronSource n.d.)

Soft launch on myös turvallisempi lähestymistapa, sillä siinä sovelluksen annetaan kerryttää hitaasti uusia käyttäjiä ennen hard launchia ilman suurta rahallista panostusta (IronSource n.d.). Sitä on myös helpompi korjata ja muuttaa palautteen mukaan, sillä käyttäjät tietävät tuotteen olevan keskeneräinen eivätkä ylläty, jos tuote muuttuu myöhemmin (Schlossberg 2022).

2.4 Haitat

Soft launchissa on myös haittapuolensa tietyissä olosuhteissa. Jos tuote on jo täysin valmis ja yrityksen tavoitteena on maksimoida tulot, testijulkaisusta saa yleensä vain hieman tai ei ollenkaan tuloja, sillä sen menestymistä ei vauhditeta

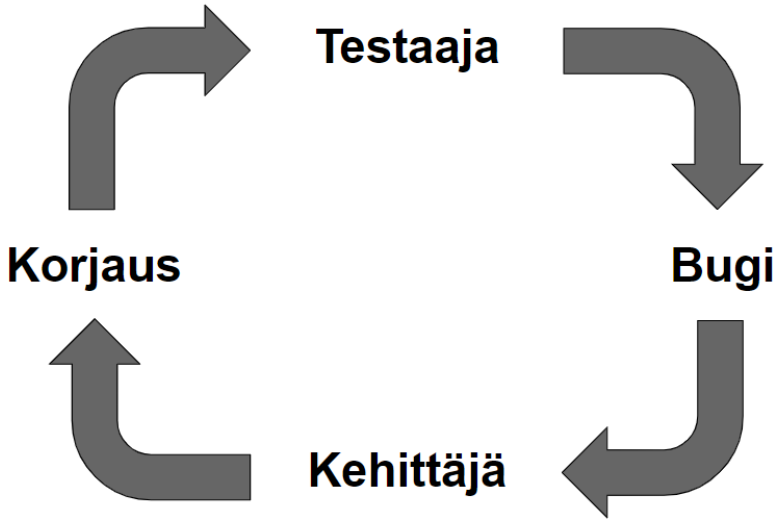
oikean julkaisun tavoin kunnon markkinoinnilla. Jos yritys soft testijulkaisee tuotteensa tällaisessa tilanteessa, sitä ei voi julkaista uudelleen myöhemmin, koska se on jo markkinoilla lopullisessa kunnossaan. Sen sijaan yrityksen täytyisi panostaa laajempaan ja kalliimpaan markkinointiin korvatakseen vähäisestä markkinoinnista johtuvan tietoisuuden puutteen. Tältä voi välttyä muistamalla, että soft launchin tarkoituksena on testata tuotteen mahdollisuuksia menestyä, yleensä keskeneräisellä tuotteella, jolloin sitä on vielä mahdollista korjata ja viedä oikeaan suuntaan ennen pääjulkaisua. (Lawley & Schure 2017.)

3 PELITESTAAMINEN

3.1 Yleisesti pelitestaamisesta

Pelitestaus on prosessi, jossa pelistä etsitään bugeja ja poikkeamia. Sen avulla varmistetaan, että sovelluksessa ei ole vikoja, kun se levitetään markkinoille. Pelitestausta suorittavat pelitestaajat, joiden tehtävänä on pelata peliä lukuisia kertoja löytääkseen bugeja ja vikoja, jotka he sitten raportoivat yksityiskohtaisesti eteenpäin. Pelitestaajia kutsutaan myös QA-testaajiksi. (Seka 2021.)

Tiivistettynä pelitestaus itsessään on palautekehä testaajan ja kehittäjän välillä. Testaaja suorittaa testejä, joiden avulla löydetyt viat raportoidaan kehittäjälle, joka korjaa ne. Kehittäjä lähettää korjatun version testaajalle, joka suorittaa jälleen testejä, ja kierre jatkuu (kuvio 1). (Schultz & Bryant 2017, 125.)



KUVIO 1. Testausprosessin palautekehä (Schultz & Bryant 2017, 125, muokattu)

3.2 Prosessi

Pelitestausprosessissa on kuusi vaihetta, joiden päätteeksi prosessi aloitetaan alusta. Prosessi alkaa suunnittelusta, jolloin tarkistetaan, mikä kaikki on muuttu-

nut viime buildin eli viimeisimmän sovelluksen kokoamisen jälkeen. Testaaja tarkistaa, mitä on lisätty tai poistettu pelistä, mitä on muutettu tai mitkä aiemmin merkityt bugit on korjattu. (Schultz & Bryant 2017, 124.) Suunnittelun aikana päätetään myös muut testauksen kannalta oleelliset asiat kuten työnjako - kuka osallistuu testaukseen, kuka kerää tarvittavat välineet ja kuka vastaa työympäristönä järjestämisestä (Seka 2021).

Kun tiedetään mitä täytyy testata, aloitetaan testiin valmistautuminen. Valmistautumisen aikana muut tiimit antavat testaajille ajantasaista tietoa pelin nykytilanteesta ja testaajat huolehtivat, että heillä on kaikki välineet ja testausympäristö valmiina testausta varten. Kehittäjätiimi myös huolehtii, että korjatut bugit on merkitty selkeästi vikalokiin ja että testaajat ovat tietoisia niistä. (Schultz & Bryant 2017, 124; Seka 2021.)

Suunnittelun ja valmistautumisen jälkeen peliä testataan suunnitelmien mukaisesti. Testaaja tarkistaa, että korjatut bugit ovat varmasti kunnossa ja ettei niitä pysty toistamaan missään, minkä lisäksi hän koittaa paikallistaa uusia bugeja, joita muutosten myötä on saattanut tulla. Jos testaaja löytää uuden bugin, hänen tulee testata sitä laajasti, jotta hän voi testin suorittamisen jälkeen kirjoittaa mahdollisimman yksityiskohtaisen bugiraportin. (Schultz & Bryant 2017, 124; Seka 2021.) Yksittäinen testi voi kestää muutamasta minuutista useampaan päivään ja samanaikaisesti voidaan suorittaa useampaakin testiä (Game-Ace 2021).

Testien suorittamisen jälkeen kaikki tehdyt löydökset tulee raportoida. Testaaja täyttää bugiraportit testin aikaisten havaintojensa perusteella kirjoittaen yksityiskohtaisesti ylös, mistä bugin voi löytää ja miten sen voi toistaa. Jos testitilanteesta on kuva- tai videomateriaalia, se on hyvä liittää bugiraporttiin. (Seka 2021.)

Raportoinnin jälkeen alkaa korjausvaihe, jolloin testaajat tekevät yhteistyötä kehitystiimin kanssa bugin korjaamista varten. Jos tarve vaatii, he antavat korjaajille lisätietoa bugista tai tekevät lisätestejä, jotta kehitystiimi ymmärtää mikä meni vikaan ja jotta bugin jäljittäminen olisi helpompaa. Kun korjaukset on tehty, testaaja palaa takaisin prosessin alkuun ja aloittaa uusien testien suunnittelun. (Schultz & Bryant 2017, 124; Seka 2021.)

3.3 Bugien käsittely

Kun testaaja on löytänyt ongelman ja kykenee kuvailemaan kaikki tavat, joilla se vaikuttaa peliin, hänen tulee ilmoittaa siitä eteenpäin. Tähän yleensä käytetään tiimin valitsemaa vianhallintatyökalua, jolla bugeja voidaan merkitä ja hallita. (Schultz & Bryant 2017, 25.) Vikalokin ylläpitäminen auttaa bugien merkitsemisessä, tarkastelussa, priorisoinnissa, kategorioinnissa ja jäljittämisessä (Seka 2021).

3.3.1 Bugien ilmoittaminen

Bugien ilmoittamisessa tärkeintä on mahdollisimman yksityiskohtainen kuvaus. Bugin kuvaaminen aloitetaan otsikolla, josta vähintään ilmenee mitä tapahtui. Otsikon olisi hyvä myös kuvata missä, miten, milloin ja kenen toimesta bugi ilmenee, jotta otsikon lukija saa käsityksen bugista ja sen tärkeydestä jo ennen bugin tarkemman kuvauksen lukemista. Yleispätevistä otsikoista ei ole hyötyä, joten niitä tulisi välttää. (Schultz & Bryant 2017, 26-27.)

Seuraavaksi kirjoitetaan bugin kuvaus. Testaaja vastaa laajemmin samoihin kysymyksiin kuin otsikossa (mitä, missä, milloin, miten). Samalla voi kirjata ylös myös keinot, joilla bugin voi mahdollisesti kumota tai välttää kokonaan. Testaaja voi vaihtoehtoisesti kirjoittaa askel askeleelta miten bugin voi toistaa, sisältäen opastukseen vain oleellisimmat tiedot. Näiden avulla bugin vakavuutta voidaan arvioida paremmin ja se antaa kehittäjille osviittaa siitä, mistä bugin korjaamisen voi aloittaa. Perusteellinen kuvaus auttaa myös myöhemmin, kun tarkistetaan, onko bugi korjattu kunnolla. Edellä mainittujen tietojen lisäksi kuvaukseen voidaan kirjoittaa missä kaikkialla pelissä testaaja on yrittänyt toistaa bugin, mutta epäonnistunut. (Schultz & Bryant 2017, 27-28.)

Kuvauksien lisäksi tulisi merkitä bugin vakavuus, jonka testaaja määrittelee parhaan kykynsä mukaan. Se kannattaa merkitä, vaikkei se olisikaan pakollinen kohta. Se auttaa tiimiä hahmottamaan kuinka kiire bugin korjaamisella on ja kuka

saattaisi olla parhain korjaamaan sen. Vakavuutta merkittäessä on hyvä huomioida, että kaikki löydetyt epäkohdat eivät ole bugeja siinä mielessä, että jokin ei toimi suunnitellusti. Testaaja saattaa löytää pelistä kohtia, joita voisi parantaa tai asioita, joita voisi lisätä peliin, jotta siitä saisi paremman. Tällaisissa tapauksissa vakavuus saattaa olla hyvinkin alhainen riippuen siitä, montako vaihtoehtoa valitussa vianhallintatyökalussa on tarjolla. (Schultz & Bryant 2017, 28-29.)

Pakollisten kohtien täyttämisen lisäksi kannattaa olla avulias ja täyttää parhaan kykynsä mukaan myös muut mahdolliset kohdat sekä antaa niin paljon lisätietoa bugista kuin mahdollista. Bugin liitteeksi voi esimerkiksi laittaa käyttöjärjestelmän virhekoodeja, palvelimen tai laitteen lokeja sekä kuvia tai videoita tilanteesta. (Schultz & Bryant 2017, 33.) Suotavaa olisi, että bugista tai koko pelitestaustilanteesta yritettäisiin aina ottaa kuva- tai videomateriaalia tilanteen analysointia varten (Seka 2021).

3.3.2 Bugien täsmentäminen

Jos löydetty bugi voidaan toistaa perustoiminnoilla, se todennäköisesti korjataan nopeammin kuin bugi, jonka syy on epäselvä, joka on löydetty myöhään tai jota on vaikea paikallistaa ja korjata. Tällaisissa tapauksissa testaajan tulisi täsmentää bugia korjattavuuden maksimoimiseksi. (Schultz & Bryant 2017, 24.)

Ensimmäinen keino bugin korjausmahdollisuuksien korottamiseksi on se, että se löydetään ajoissa. Aina kun peliin lisätään jotakin uutta, esimerkiksi uusia toimintoja tai tavaroita, testaajan tulisi testata ne välittömästi. Jos testaaja on epävarma muutoksista ja lisäyksistä, hänen tulisi pyytää raportti muutetuista asioista. Toinen keino parantaa erityisesti epäselkeän bugin korjausmahdollisuuksia on tutkia missä kaikkialla vastaavanlainen bugi saattaa esiintyä. Testaajan tulisi esimerkiksi testata kaikki funktiot, jotka käyttävät viallista ominaisuutta sekä kaikki esineet ja hahmot, jotka jakavat jonkin ominaisuuden viallisen objektin kanssa. Lopuksi testaaja voi kasvattaa bugin esiintymistiheyttä täsmentämällä prosessia, jolla bugi saadaan toistettua. Tämän voi tehdä karsimalla ylimääräiset toimet pois prosessista ja löytämällä yleisempiä skenaarioita, jotka sisältävät loput tarvittavat toimet bugin ilmestymistä varten. (Schultz & Bryant 2017, 24-25.)

3.4 Testaustavat

3.4.1 Black box -ja White Box -testaus

Testaustavat voidaan jakaa kahteen kategoriaan – black box -testaukseen ja white box -testaukseen (Game-Ace 2021). Black box -testauksessa testaajan ei tarvitse tietää koodista tai päästä siihen käsiksi, vaan peliä pelataan tavallisen pelaajan tapaan. Testaajat käyttävät testeissä samoja laitteita ja välineitä kuin pelaajat, oli kyseessä sitten hiiri, näppäimistö tai peliohjain. (Schultz & Bryant 2017, 120.) Black box -testaus onkin kaikkein saavutettavin testaustapa, koska testaajan ei tarvitse tietää mitään ohjelmoinnista tai pelistä ennestään, vaan hän voi keskittyä pelkästään käyttäjäkokemuksen testaamiseen (Game-Ace 2021).

Black box -testaukseen verrattuna white box -testaus on paljon tarkempaa ja vaatii syvällistä tietoa pelin arkkitehtuurista ja koodista, minkä takia testit tehdään yleensä kehittäjien toimesta ennen kuin peli lähetetään QA-tiimin testattavaksi (Game-Ace 2021). Testaaja suorittaa testit suoraan koodin kautta, koittaen ennustaa kaikki mahdolliset tavat, joilla testattavaa koodia voidaan käyttää. Pelin testaaminen pelkästään white box -testaustavoilla on kuitenkin hyvin vaikeaa, koska pelkästään koodia testaamalla on lähes mahdotonta ennakoida monimutkaisempia yhdistelmiä, joita pelaaja saattaa käyttää. (Schultz & Bryant 2017, 122.)

3.4.2 Toiminnallinen testaaminen

Toiminnallinen testaaminen on black box -testaustekniikka ja sen tarkoituksena on sovelluksen toimivuuden tarkistaminen annettujen spesifikaatioiden mukaisesti. Käytännössä tämä tarkoittaa bugien ja muiden sellaisten virheiden tunnistamista, jotka saattavat vaikuttaa käyttäjäkokemukseen. Toiminnallisessa testaamisessa etsitään ongelmia muun muassa pelattavuudesta, grafiikoista ja audiovisuaalisesta puolesta sekä tarkistetaan esimerkiksi maksuväylien toimivuus ja pelin asentamisen toimivuus. Toiminnallinen testaaminen vie tämän takia paljon aikaa ja voi ajoittain mennä monimutkaiseksi. (Seka 2021; Testbytes 2022.)

3.4.3 Kombinaatiotestaus

Kombinaatiotestausta käytetään yleensä sovellusten testaamisessa, mutta se soveltuu käytettäväksi myös pelitestauksessa (Seka 2021; Testbytes 2022). Sen tarkoituksena on tarkistaa kaikki mahdolliset parametrien yhdistelmät. Parametrit valitaan esimerkiksi pelin toiminnoista, asetuksista ja hahmon ominaisuuksista. Testaaminen on systemaattista, minkä ansiosta voidaan tehdä vaivattomammin helposti seurattavia raportteja. Kombinaatiotestauksella voidaan esimerkiksi testata, toimiiko kaikki mahdolliset ääniasetusten yhdistelmät. (Seka 2021.)

3.4.4 Yhteensopivuustestaus

Yhteensopivuustestauksessa tarkistetaan, että peli toimii oikein kaikilla halutuilla laitteilla ja ruutukoilla. Tämän takia se onkin yksi tärkeimmistä testeistä mitä pelille voidaan tehdä. Testeissä tarkastellaan tekstien luettavuutta ja yleisesti pelin käyttöliittymää eri laitteilla sekä verrataan niitä toisiinsa. Testien aikana tarkistetaan, että peli täyttää kehittäjien ja loppukäyttäjien vaatimukset ja varmistetaan sen vakaus, toimivuus, skaalautuvuus ja käytettävyys eri laitteilla. Näiden testien avulla varmistetaan myös kaikkien testausympäristöjen keskinäinen yhteensopivuus. (Seka 2021; Testbytes 2022.)

3.4.5 Regressiotestaus

Regressiotestauksessa korjatuksi merkitty bugi yritetään toistaa aiemmin kirjattujen ohjeiden mukaan, jotta saadaan varmistus siitä, onko bugi varmasti korjattu vai ei (Schultz & Bryant 2017, 132). Samalla tehdään uusia, samankaltaisia testejä, joilla varmistetaan ettei bugin korjaaminen rikkonut mitään muuta ominaisuutta (Kramarzewski & De Nucci 2018). Regressiotestauksen avulla voidaan säästää aikaa, kun uudet korjaukset ja mahdollisesti niiden mukana tulleet uudet bugit tarkistetaan heti sen sijaan, että ne kulkisivat projektin mukana (Testbytes 2022).

3.4.6 Ad hoc -testaus

Ad hoc -testaus on suunnittelematonta testaamista, jossa testaaja testaa peliä satunnaisesti haluamallaan tavalla (Testbytes 2022). Sen tarkoituksena on vastata testaajan improvisoituun kysymykseen “mitä tapahtuu, jos teen näin?”, joka hänelle on saattanut tulla mieleen alitajuisesti tai tiedostamatta muita testejä tehdessään. Se myös antaa testaajalle tauon hänen tavallisesta työstään ja auttaa häntä testaamaan peliä uusin silmin, joka taas ehkäisee sitä, ettei testaaja tule sokeaksi omalla osa-alueellaan ilmeneville bugeille. (Schultz & Bryant 2017, 273-275.)

Vaikka ad hoc -testaus onkin suunnittelematonta ja vapaampaa, testaajan olisi silti hyvä määrittää etukäteen tavoitteensa sekä nauhoittaa testaustilanne siltä varalta, että hän löytää uuden bugin (Schultz & Bryant 2017, 276-278). Ilman dokumentointia testaajan on vaikeampi toistaa mahdollisesti löydetyt bugit jälkikäteen (Testbytes 2022).

3.4.7 Pelaamistestaus

Pelaamistestauksessa testaajat pelaavat keskeneräistä peliä ikään kuin oikeina pelaajina, analysoiden pelin laatua ja ei-toiminnallisia ominaisuuksia kuten esimerkiksi pelin viihdearvoa, tarinaa, vaikeustasoa ja kenttäsuunnittelua. Päätavoitteena on tarkistaa, että peli on hyvin rakennettu. (Seka 2021; Testbytes 2022.) Pelaamistestaus eroaa muista testaustavoista siten, että siinä keskitytään enemmän pelin arviointiin kuin bugien etsimiseen (Seka 2021).

3.5 Pelitestaussessiot

Pelitestaussessioissa testaajina toimivat pelin kohdeyleisöön kuuluvat henkilöt. Yleensä testaajiksi valitaan kollegoita, mutta yritys voi myös rekrytoida ulkopuolisia testaajia tai ulkoistaa rekrytoinnin testaukseen erikoistuneelle yritykselle. Testaajiksi on mahdollista saada myös tuntemattomia ihmisiä, esimerkiksi julki-

silta paikoilta tai tapahtumista, etenkin jos kyseessä on helposti esiteltävä mobiilipeli. Pelitestaussessiot voivat olla valvottuja tai valvomattomia. Sessioiden tavoitteet, testattavat ominaisuudet ja mahdolliset kysymykset tulisi määrittää ennen sessiota. (Kramarzewski & De Nucci 2018).

Pelitestaussessiot voidaan jakaa kolmeen kategoriaan: yksittäinen, ryhmä ja julkinen. Jokaisessa on omat hyötynsä ja haittansa, joten yrityksen täytyy vertailla niitä ja valita se, jonka he kokevat hyödyllisimmäksi. (Kramarzewski & De Nucci 2018).

Yksittäisessä sessiossa pelaaja pelaa peliä itsenäisesti avustajan seurannan alaisena. Sessio yleensä myös nauhoitetaan. Yksittäinen sessio antaa tilaisuuden paikallistaa tarkemmin tiettyjä käyttöön liittyviä ongelmia ja antaa mahdollisuuden haastatella testaajaa ilman ryhmäpaineen vaikutusta tämän mielipiteisiin. Yksittäisen sessiot vievät kuitenkin aikaa, sillä jokaisessa sessiossa täytyy olla avustaja paikalla. Testaajia on yleensä myös vähemmän, jolloin tulokset voivat olla epäluotettavia. Yrityksen täytyykin arvioida testaajansa huolella, jotta sessioista kerätty palaute olisi mahdollisimman hyödyllistä. (Kramarzewski & De Nucci 2018).

Ryhmäsessioissa peliä testaa useampi henkilö samanaikaisesti. Ryhmäsessioissa saavutetaan enemmän tuloksia sen hinnalla, että ryhmän jäsenten välinen kommunikaatio voi vääristää tuloksia. Valvotuissa sessioissa testaajat ovat samassa tilassa ja heitä valvotaan kehitystiimin jäsenten toimesta. Testaajien välinen kommunikointi ja tilanteen luoma rennompi ilmapiiri voi heikentää arvioinnin laatua. Valvomattomissa sessioissa testaajat pelaavat peliä omalla ajallaan tietyn aikarajan sisällä ja palauttavat tulokset suullisesti tai kirjallisesti. Valvomattomia sessioita on helpompia järjestää, mutta ne heikentävät palautteen määrää ja laatua. Sessiot voivat kuitenkin kestää pidempään ja ovat täten hyödyllisiä, jos halutaan testata esimerkiksi pelin tarinaa tai tasapainotusta. (Kramarzewski & De Nucci 2018).

Julkinen testaussessio järjestetään yleensä alpha -tai beta -testinä, jolloin peli julkaistaan valitun väylän kautta. Peliä pääsee testaamaan kuka tahansa testaajaksi sopiva henkilö tai vain osa testaukseen sopivista kandidaateista. Tällöin se

saattaa vastata soft launchia. Palautetta voi kerryttää useamman kanavan kautta, esimerkiksi kyselyillä, arvosteluilla tai pelille tehtyjen foorumien kautta. Tilannetta ei kuitenkaan voida valvoa, jolloin palautteesta voi jäädä uupumaan pienempiä epäkohtia. (Kramarzewski & De Nucci 2018).

3.6 Yhteistyö muiden osastojen kanssa

QA-testaajat työskentelevät yhdessä ohjelmoijien kanssa bugien korjausta varten, jonka lisäksi he tekevät yhteistyötä myös pelisuunnittelijoiden kanssa. QA-tiimi hyödyntää pelisuunnittelijoiden tekemää dokumentaatiota testien suunnittelussa ja toteutuksessa, etenkin jos testattavaa ominaisuutta varten tarvitaan tarkempaa tietoa. Mitä enemmän tietoa pelisuunnittelijat voivat tarjota QA-testaajille, sen helpompaa ja tehokkaampaa testaaminen on. (Kramarzewski & De Nucci 2018).

Pelisuunnittelijat voivat myös pyytää QA-tiimiltä palautetta suunnitelmistaan hyvissä ajoin ennen kuin ominaisuus on edes pelissä tai testattavissa. QA-tiimi pelaa peliä lukuisia kertoja koko pelinkehityksen ajan ja heillä saattaa olla sen kokemuksen pohjalta näkemyksiä ja ajatuksia pelin tasapainottamiseen liittyen, joita suunnittelijoille ei ole tullut mieleenkään. (Kramarzewski & De Nucci 2018).

4 TOIMEKSIANTAJAN LÄHTÖTILANNE

4.1 Soft launch -prosessi

Toimeksiantajan soft launch -prosessissa on kaksi vaihetta ja riippuen ensimmäisen vaiheen tuloksista toista vaihetta ei saateta edes toteuttaa. Ensimmäisessä vaiheessa pelin suosiota oikeiden käyttäjien keskuudessa testataan lataamalla peli valitun maan pelikauppaan joko suljettuna alphana (Closed Alpha) tai avoimena betana (Open Beta, toiselta nimeltään Early Access). Suljetussa alphassa peliä pääsee testaamaan vain linkin kautta, kun taas avoimessa betassa pelin voi löytää myös orgaanisesti pelikaupasta. Kummassakin tapauksessa toimeksiantaja sulkee pelin lataussivun, kun valittu määrä latauskertoja tulee täyteen ja testi päättyy virallisesti 7 vuorokautta lataussivun sulkemisen jälkeen.

Toisessa vaiheessa peli testijulkaistaan virallisesti, eli se viimeistään tässä vaiheessa laitetaan avoimeen betaan. Toisin kuin ensimmäisessä vaiheessa, toisessa vaiheessa peli julkaistaan useammassa maassa eikä sen lataussivuja yleensä suljeta. Soft launch voi tapahtua vasta useammankin kuukauden kuluttua ensimmäisestä vaiheesta, riippuen siitä kuinka paljon peliä halutaan kehittää ja parannella ensimmäisen vaiheen testitulosten perusteella ja kuinka kiire sen soft launchilla on.

4.1.1 Ensimmäinen vaihe

Ensimmäisen vaiheen testin valmistelut vastaavat aika pitkälti soft launchin ja virallisen julkaisun valmisteluja. Pelille luodaan kauppasivut Google Playhin, jonka jälkeen se lähetetään tarkastettavaksi. Jos pelissä on Google Playn sääntöjen vastaista sisältöä tai muita epäkohtia, sen julkaisuoikeus evätään, kunnes epäkohdat on korjattu. Peli voidaan julkaista vasta, kun se menee kaupan tarkastuksesta läpi. Toimeksiantajalla on mennyt tarkastukseen yleensä muutama tunti, mutta siihen voi mennä maksimissaan jopa seitsemän vuorokautta. Tästä syystä toimeksiantaja pyrkii lähettämään pelin tarkastukseen yleensä noin viikko ennen

suunniteltua testin aloitusta. Tarkastuksen aikana peliin ei mieluusti tehdä muutoksia, sillä muuten peli joudutaan lähettämään uudelleen tarkastettavaksi.

Ensimmäisen vaiheen testi toteutetaan Brasilian Google Playssa edullisuutensa vuoksi. Google Playssa on myös paremmat työkalut suljettuihin testeihin, jos testi halutaan toteuttaa suljettuna alphana.

Valmistelujen aikana QA-tiimi tarkistaa pelin toimivuuden yhtä huolellisesti kuin normaalistikin annettujen tietojen perusteella. QA-tiimi testaa peliä sen mukaan mitä heille on kerrottu joko suoraan tai suunnitteludokumentin kautta. Jos tiimille ei ole informoitu jonkin ominaisuuden oikeasta toiminnasta, vastuu ominaisuuden toimivuudesta on sen tekijällä.

Ensimmäisen vaiheen testissä tarkastellaan lähinnä ensimmäisen, kolmannen ja seitsemännen päivän retentiota eli sitä, kuinka moni pelaaja palaa pelin pariin verrattuna edelliseen määriteltyyn ajankohtaan. Toimeksiantaja on määritellyt etukäteen retentiokynnykset, joiden mukaan katsotaan, menestyikö peli tarpeeksi hyvin testissä vai ei. Pelissä ei välttämättä ole tarpeeksi sisältöä seitsemännen päivän retentiota varten, joten se ei ole yhtä tärkeä kuin ensimmäisen ja kolmannen päivän retentiot. Se sisällytetään kuitenkin testiin, jotta tiedetään vetääkö testattava peli sinne asti vai ei. Testissä tarkastellaan myös alustavia monetisaatiolukuja ja sitä, saadaanko testiin laitetulle rahalle edes hieman vastinetta vai jäädäänkö tappiolle.

Ensimmäisen vaiheen testin tulokset analysoidaan ja jos peli ei menestynyt tarpeeksi hyvin, sen tuottaminen lopetetaan ja siirrytään suunnittelemaan seuraavaa projektia. Testillä voidaan täten varmistaa, onko peli soft launchin arvoinen vai ei jo hyvissä ajoin ennen kuin peliin upotetaan enemmän resursseja. Jos peli taas menestyi testissä, toimeksiantaja aloittaa seuraavan vaiheeseen eli soft launchiin valmistautumisen.

4.1.2 Toinen vaihe

Toisen vaiheen valmistelut eroavat hieman ensimmäisen vaiheen valmisteluista ja riippuen ensimmäisen testin menestyksestä, prosessia saatetaan nopeuttaa. Merkittävimpänä erona ensimmäiseen vaiheeseen on, että peli julkaistaan avoimessa betassa useammassa maassa – suljettu alpha ei ole edes vaihtoehtona. Sen annetaan kerryttää pelaajia sen sijaan, että se suljettaisiin tietyn tavoitteen saavuttamisen jälkeen. Peliä päivitetään tuotannon edetessä ja pelaajat voivat antaa palautetta uusista muutoksista sitä mukaa, kun niitä tulee. Palautteiden avulla kehitystiimi voi parannella ominaisuuksia aina vain paremmiksi. Peli on yleensä avoimessa betassa viralliseen julkaisuun asti.

Peli julkaistaan useammassa maassa, joihin sisältyy Brasilian lisäksi esimerkiksi Filippiinit ja Taiwan. Toisin kuin ensimmäisessä vaiheessa, peli julkaistaan Google Playn lisäksi myös Apple Storessa.

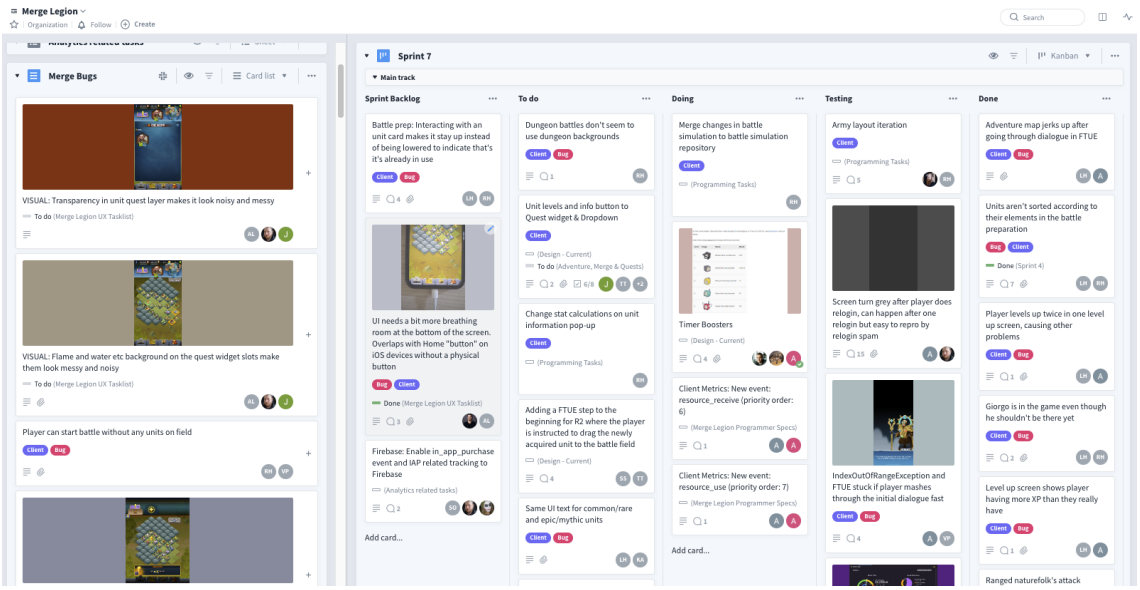
4.2 Pelitestaus

Toimeksiantajan pelitestaus on jatkuva prosessi. Peliä testataan kehitysvaiheessa päivittäin, erityisesti uusien ominaisuuksien implementoinnin jälkeen tai juuri ennen buildin tekoa sprintin lopussa. QA-tiimi on päävastuussa pelin testaamisesta sekä bugien etsinnästä ja tarkistamisesta, mutta myös muita työntekijöitä rohkaistaan testaamaan peliä.

4.2.1 Työkalut ja niiden käyttö

Toimeksiantaja hyödyntää kehitys- ja testausprosessissaan yhteissuunnittelusivusto Favroa. Sivustolle voi luoda erilaisia kokoelmia, joiden sisällön voi suunnitella omien tarpeidensa mukaan. Toimeksiantaja on luonut sivustolle kokoelman, jonka sisällä sprinttien suunnittelu, sprintin työtehtävien edistymisen seuranta sekä bugien kirjaaminen tapahtuu.

Kokoelman (kuva 1) vasemmalle puolelle on aseteltu erilaisia tehtävälistoja, joista pelitestaamisen kannalta tärkein on “Merge Bugs”. Kun uusi bugi tai visuaalinen ongelma havaitaan, siitä luodaan kortti kyseiselle tehtävälistalle. Korttiin merkitään bugin nimi, kuvaus bugista mahdollisesti kuvien ja videoiden kera, sen vakavuus ja kiireellisyys, kuka kortin on tehnyt ja kenelle kortin tarkistaminen kuuluu. Bugikortti merkitään myös tageilla, jotka näkyvät kortilla jo ennen sen avaamista, mikä helpottaa bugikortin erottamista muiden korttien seasta sen jälkeen, kun se on siirretty oman lohkonsa alle. Ohjelmoijat tarkistavat bugilistan päivittäin ja ottavat bugit korjaukseen sitä mukaa, kun pystyvät tai kun tarve vaatii.



KUVA 1. Toimeksiantajan Favro-näkymä

Kokoelman oikealla puolella on sprinttinäkymä. Jokaiselle sprintille luodaan oma lokerikkonsa lohkoineen, joiden alle laitetaan niille kuuluvat tehtävät korttimuodossa. Jokaisella lohkolla on viisi listaa: Sprint Backlog, To Do, Doing, Testing ja Done. Kortteja siirretään listojen välillä sen mukaan, missä vaiheessa kortti on.

Favron lisäksi toimeksiantaja hyödyntää viestintäsovellus Slackia. Slackissa voi tehdä organisaation alle useita erilaisia tekstikanavia, joiden avulla erilaista tietoa on kätevä jakaa muille työntekijöille. Toimeksiantajalla onkin jokaiselle osa-alueelle sopiva tekstikanava, jolla aiheeseen liittyvä keskustelu käydään. Työntekijöitä rohkaistaan käyttämään Slackia päivittäin, jotta jokaisella olisi mahdollisimman ajantasaista tietoa niin oman kuin muidenkin tiimien toiminnasta.

Testaus itsessään tapahtuu joko suoraan Unity-pelinkehitysalustassa tai puhelimilla, joille on asennettu Unityn kautta tehty build. Testaajat saavat uusimman version pelistä Git-pohjaisen Bitbucketin kautta, jota toimeksiantaja käyttää versionhallinnan ylläpidon palveluntarjoajana.

4.2.2 Prosessi

Toimeksiantajalla on kolme erilaista testausmenetelmää. Ensimmäinen menetelmä on hyvinkin suoraviivainen ja siinä QA-tiimi testaa uudet implementoidut ominaisuudet sitä mukaa, kun niitä tulee. Ohjelmoijat työstävät pelisuunnittelijoiden kuvaamia ominaisuuksia ja laittavat niihin liittyvät kortit aiheeseen liittyvän lohkon listalle riippuen kortin kehitysvaiheesta. Kun ominaisuudet ovat valmiita, ohjelmoija siirtää kortin Testing-listalle. Tämän jälkeen QA-tiimi testaa ominaisuuden toimivuuden. Jos ominaisuus on kerralla kunnossa, se siirretään Done-listalle. Jos taas ominaisuus on viallinen, testaaja kertoo ongelmasta ohjelmoijalle joko kommentoimalla suoraan korttiin tai laittamalla hänelle viestiä Slackin kautta. Tällöin kortti palautetaan joko To Do -tai Doing -listalle, riippuen tilanteesta. Kortti palautuu korjausten jälkeen Testing-listalle, jolloin se testataan uudelleen.

Toinen menetelmä on vapaampi, mutta samalla myös haastavampi, sillä se ei sisällä ennalta annettuja ohjeita. Sen sijaan QA-testaaja pelaa peliä vapaasti valitsemallaan tavalla, yrittäen löytää bugeja tai muita parannuskohteita. Näiden lisäksi testaaja tarkistaa grafiikat ja käyttäjäkokemuksen. Kun bugi tai muu korjausta vaativa kohta löytyy, testaaja tekee siitä bugikortin Favroon Merge Bugs -listalle. Ohjelmoijat, graafikot tai suunnittelijat korjaavat bugeja niiden vakavuuden ja kiireellisyyden mukaan, siirtäen niitä sprinttien lohkoille. Kun bugi on korjattu, se päätyy Testing-listalle, josta se liikkuu edellisen menetelmän tavoin joko eteen- tai taaksepäin riippuen siitä, onko se korjattu.

Jos löydetty bugi on vakava ja estää pelin pelaamisen, QA-tiimi on suoraan yhteydessä henkilöön, joka on vastuussa kyseisestä pelin osa-alueesta. Tällä tavalla bugi saadaan nopeammin korjaukseen, jotta pelin pelaamista ja testaamista voitaisiin jatkaa.

Kolmas menetelmä hyödyntää muita työntekijöitä, erityisesti niitä, jotka eivät ole aktiivisesti mukana pelin kehittämisessä. QA-tiimi tekee yleensä sprintin lopussa, mutta myös sprintin keskellä buildeja, jotka ladataan App Center -sivustolle. Sieltä jokaisen työntekijän on mahdollista ladata build omalle puhelimelleen pelattavaksi. Tällä tavalla saadaan kerättyä enemmän dataa siitä, millaista peliä on pelata ja saatetaan löytää bugeja, joita QA-tiimi ei ole vielä löytänyt, sillä testatessa voi tulla sokeaksi ongelmille, jotka ovat muille ilmiselviä.

Tässä menetelmässä bugit ja palautteet kerätään Favron sijasta Slackiin oikeille tekstikanavilleen. Tällöin kaikilla on mahdollisuus kommentoida löydettyjä ongelmia tai havaintoja. QA-tiimi käy bugi-kanavan läpi ja testaa löydetyt ongelmat. Jos he pystyvät toistamaan löydetyn ongelman eikä sitä ole jo merkitty bugilistalle, he merkitsevät sen sinne.

5 MUUTOKSET

5.1 Parannusehdotukset

QA-tiimissä työskentelyn aikana huomasin Favron bugilistan kasvavan ajan myötä, kun löydetyt bugit kasaantuivat ja vain vakavimmat ehdittiin korjaamaan muun työn ohessa. Tämä aiheutti sen, että pelin hiontaan tarkoitetuille sprinteillä ohjelmoijien To do -listoille siirrettiin lukuisia bugeja, joita heidän piti korjata samalla, kun he ohjelmoivat uusia ominaisuuksia ja hioivat jo olemassa olevia, puhumattakaan vakavampien bugien korjaamisesta. Kasvava bugilista haittaa myös QA-tiimin toimintaa, sillä listan hallinta ja selaaminen vaikeutuu joka bugin myötä. Mitä pidempi lista oli, sen varmemmin joku merkitsee jo tunnetun bugin toistamiseen listalle, koska edellistä mainintaa on hankala löytää muiden joukosta.

Ohjelmoijien työtaakan keventämiseksi ja bugilistan hallinnan ja seuraamisen helpottamiseksi ehdottaisin bugien järjestelmällisempää korjaamista sprinttien aikana. Käytännössä tämä toteutuisi siten, että ohjelmoijille varattaisiin sprintistä tietty tuntimäärä siihen, että he keskittyvät pelkästään bugilistan läpikäyntiin ja bugien korjaamiseen. Tähän käytettävä aika voitaisiin määritellä jokaisen sprintin alussa sen mukaan, kuinka paljon bugeja listalla on. Tällä tavalla pienempiä, mutta silti oleellisia bugeja ehdittäisiin korjaamaan jo aikaisemmin eikä niiden korjaaminen kasaantuisi myöhemmille sprinteille.

Toinen tapa helpottaa bugilistan hallintaa olisi muokata bugikortteja ja niiden tageja. Tällä hetkellä kortista näkee otsikon, tagit ja siihen merkityt ihmiset korttia avaamatta, mutta muut infot, kuten bugin kiireellisyyden ja vakavuuden, näkee vasta avauksen jälkeen. Kokisin sen hyödylliseksi laittaa bugin kiireellisyyden myös tagiksi, jotta sen näkisi suoraan korttia avaamatta. Bugin vakavuuden merkitseminen yhtä näkyvästi voisi olla myös eduksi, mutta se saattaisi ruuhkauttaa avaamattoman kortin ulkoasun, etenkin kun projektissa aktiivisesti mukana olevat jäsenet pystyvät melko hyvin päättelemään bugin vakavuuden jo otsikon perusteella. Kiireellisyyttä taas on hankalampi sanoa suoraan ja voi olla tilanteita, joissa

bugi itsessään on hyvin pieni, mutta sen verran toistuva, että se olisi syytä korjata mahdollisimman pian.

Pelitestaussessiosta voidaan saada paljonkin dataa siitä, mikä toimii ja mikä ei. Traplight testauttaa pelejään työntekijöillään, jonka lisäksi he hyödyntävät maksullista Playtestcloud-sivustoa saadakseen kommentoituja videoita kohderyhmään kuuluvien pelaajien pelitestaussessioista. He eivät itse järjestä valvottuja testitilanteita yrityksen ulkopuolisille henkilöille, koska niiden järjestämiseen uppoaa paljon aikaa, mutta sitä voisi mahdollisesti helpottaa tekemällä yhteistyötä TAMKin Games Academyn kanssa.

Yhteistyö TAMKin Games Academyn kanssa hyödyttäisi niin Traplightiä kuin Games Academyn opiskelijoita. Testaustilanteen järjestämiseen liittyviä tehtäviä voitaisiin delegoida opettajille, jotka voisivat huolehtia esimerkiksi opiskelijoiden saamisesta oikeaan paikkaan oikeaan aikaan. Tällöin Traplightille jäisi enemmän aikaa kehitystyöhön. Testaustilanteesta voisi tehdä myös kurssitehtävän opiskelijoille, jolloin he opettajan ohjeistuksella voisivat suunnitella ja toteuttaa pelitestaustilanteita ja raportoida tulokset yritykselle. Vastaavanlaista yhteistyötä on tehty aiemminkin, joten se ei olisi TAMKin opettajille uusi juttu. Tämä tarjoaisi opiskelijoille käytännön kokemusta pelitestaamisesta ja mahdollisuuden verkostoitua ja yritys taas saisi runsaasti dokumentoitua palautetta pelistään.

Toisaalta yhteistyössä voi olla myös haittansa. Playtestcloudin kautta toimeksiantaja voi saada videot pelitestaustilanteista 48 tunnin sisällä, jopa alle 24 tunnissa. Jos peli on vasta kehityksessä, työ pitäisi tauottaa palautteenkeruun ajaksi, ettei esimerkiksi lähdetä työstämään ominaisuutta, joka ei toimikaan. TAMKin kanssa tehdyssä yhteistyössä tulosten saamisessa voisi mennä paljon kauemmin kuin kyseistä palvelua käyttämällä. Palautteen laatua ei voisi myöskään taata, vaikka kyseessä olisikin pelialan opiskelijoita. Jos he eivät täsmää haluttua kohderyhmää, he saattavat keskittyä vääriin asioihin ja tulokset voivat vääristyä. Siinä on myös riskinsä, että tieto vasta kehityksessä olevasta pelistä vuotaa, kun se esitellään suurelle määrälle testaajia.

Toimeksiantajan tämän hetkisessä projektissa ei ole erikseen suunnitteludokumenttia, jossa olisi kootusti tiedot kaikista pelin ominaisuuksista, vaan vain suunnittelijat tietävät, mitä pelissä pitäisi olla ja luovat sen perusteella Favroon kortteja ohjelmoijille. Tieto halutusta ominaisuudesta ja sen kaikista piirteistä saattaa muuttua tai kadota siinä välissä, kun se pilkotaan pienempiin osiin ohjelmoijien työskentelyn helpottamiseksi ja tämän takia QA-tiimi ei välttämättä saa kaikkea oleellista tietoa testattavista ominaisuuksista. Tämän seurauksena peliin saattaa jäädä keskeneräinen ominaisuus, joka huomataan vasta myöhemmin. Tiedon puute voi olla haitaksi esimerkiksi testijulkaisuun valmistautuessa, koska ei voida olla varmoja siitä, onko kaikki testiin tarvittavat ominaisuudet varmasti pelissä.

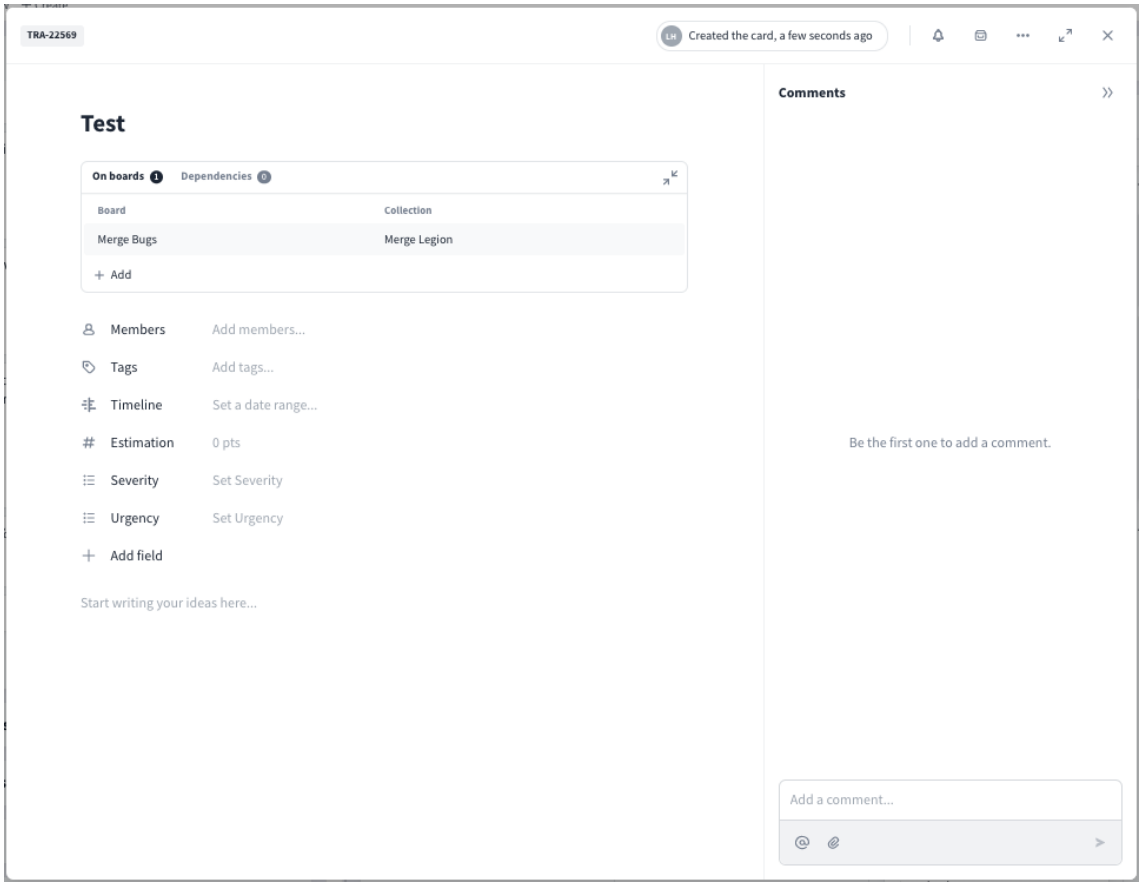
Edellä mainittujen syiden takia ehdottaisinkin, että ennen julkaisuja suunnittelijat kirjoittaisivat ytimekkäät tarkistuslistat siitä, mitä ominaisuuksia pelissä pitäisi olla, miten niiden kuuluisi toimia ja aiotaanko niihin vielä lisätä jotakin. Vaihtoehtoisesti suunnittelijat voisivat pitää palavereita QA-testaajien kanssa läpi projektin ja kertoa, miten minkäkin ominaisuuden pitäisi toimia siltä varalta, että jokin oleellinen tieto hukkuu kehityksen aikana. Jos palaverin pitää tarpeeksi aikaisessa vaiheessa, mieluusti jo ennen kuin ominaisuus annetaan ohjelmoijien tehtäväksi, suunnittelijat voisivat myös saada palautetta halutusta ominaisuudesta QA-tiimiltä. QA-tiimi on kuitenkin pelannut peliä lukuisia kertoja, joten heillä saattaa olla ideoita siitä, miten jotakin ominaisuutta tai yleisesti käyttäjäkokemusta voisi parantaa sen sijaan, että ominaisuudesta annettaisiin palautetta vasta, kun se on testauksessa.

5.2 Jatkotoimenpiteet

Parannusehdotuksista kaksi koettiin kokeilemisen arvoisiksi; bugikorttien uudistaminen ja suunnittelijoiden ja QA-tiimin välisen yhteistyön parantaminen. Sain tehtäväkseni suunnitella ja esitellä ideani siitä, miten uudistaisin bugikortteja. Yhteistyön parantamiseen liittyvä ehdotus viedään eteenpäin ja käsitellään, mutta siihen ei tule konkreettista muutosta vielä tämän opinnäytetyön työstämisen aikana.

Ehdotukseni bugilistan aikataulutetusta läpikäymisestä on taktiikka, jota toimeksiantaja jo hyödyntää, kun peli on tasaisemmassa kehitysvaiheessa. Tämän hetkinen projekti on vielä kehitysvaiheen alussa ja sitä kehitetään nopealla tahdilla, minkä takia taktiikkaa ei ole käytetty läsnäollessani enkä täten ollut siitä tietoinen. Toimeksiantaja on pohtinut yhteistyötä TAMKin kanssa jo aiemmin, mutta sitä ei ole otettu käyttöön. Kuten pohdin jo ehdotusta suunnitellessani, se vie liikaa aikaa ja toimeksiantaja kokee Playtestcloud-palvelun sopivammaksi pelitestaussessioiden järjestämiseen.

Parannusehdotusten esittelyn jälkeen ryhdyin suunnittelemaan bugikorttien uudistamista. Tutustuin aiempaa tarkemmin nykyiseen bugikorttipohjaan (kuva 2) ja siihen, mitä kaikkia kohtia voisi muuttaa. Olen käyttänyt pohjaa lähes päivittäin ja huomannut, ettei kohtia “Timeline” ja “Estimation” juurikaan käytetä nykyisessä kehitystyössä. Varmistin niiden vähäisen tai lähes olemattoman käytön myös toiselta tiimin jäseneltä ja pohdinkin, että ne voisi poistaa kokonaan tai mahdollisesti sijoittaa listan alimmaisiksi.



KUVA 2. Toimeksiantajan nykyinen bugikorttipohja

Seuraavaksi pohdin sitä, miten kiireellisyys olisi viisainta merkitä korttiin. Olin ehdotuksessani pohtinut, että laittaisin sen korttiin tagina, jotta se näkyisi korttia avaamatta, mutta pohjaa tehdessäni oivalsin, että listan asetuksista voi määrittää mitkä kohdat näkyvät tagin tavoin suoraan kortilla. Tämän oivalluksen ansiosta en muuttanutkaan kiireellisyyttä tagiksi, vaan sen sijaan laitoin sen näkymään kortilla listan asetuksista ja keskityin siihen, voisiko sitä parantaa jotenkin muuten.

Tällä hetkellä bugikorteissa kiireellisyyttä kuvataan kolmella termillä; low, medium, high. Olen kuitenkin työskennellessä huomannut, että neljännestä vaihtoehdosta voisi olla hyötyä, sillä välillä jokin bugi saattaa tippua vaihtoehtojen väliin tai olla vakavampi kuin nykyinen ylin vaihtoehto. Neljäs vaihtoehto voisikin olla nimeltään “immediate” ja sitä käytettäisiin välitöntä korjausta vaativiin bugeihin. Tämän lisäyksen ansiosta eri vaihtoehdot olisi helpompi erottaa toisistaan ja täten tekisi kiireellisyyden valinnasta helpompaa.

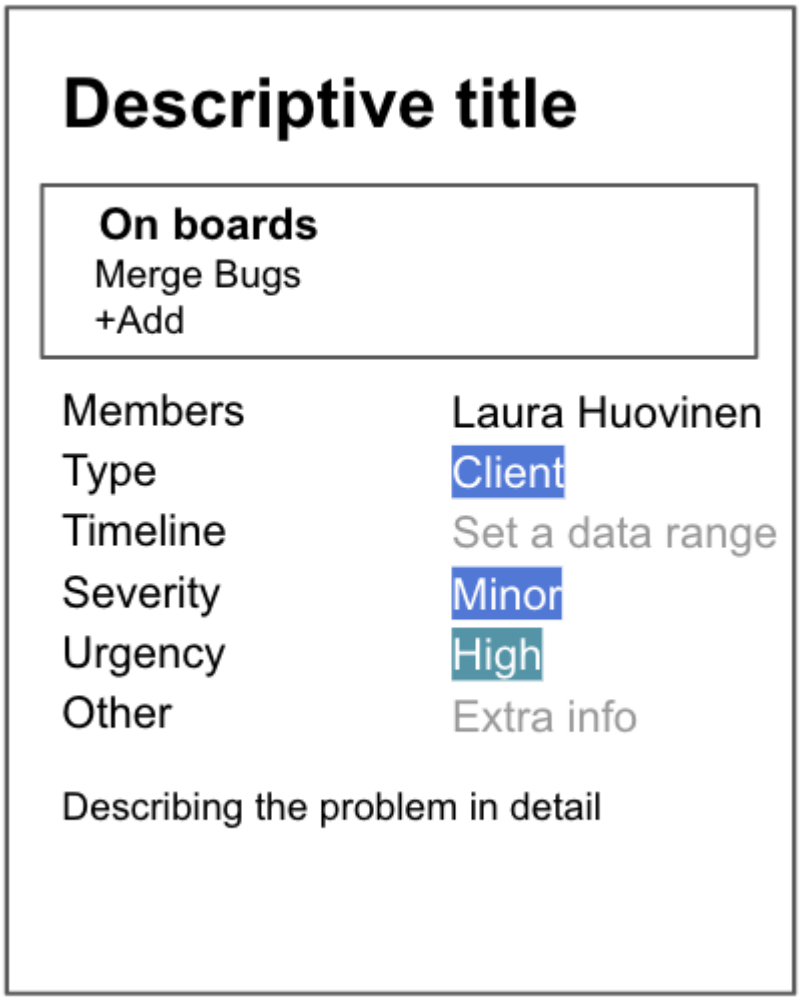
Myös vakavuuteen viittaavat termit voisivat kaivata vastaavanlaista muutosta. Tällä hetkellä vakavuutta kuvataan termeillä “minor”, “medium”, “critical” ja kuten kiireellisyydenkin kohdalla, jotkin bugit tippuvat vaihtoehtojen “medium” ja “critical” väliin. Ehdottaisinkin, että näiden kahden vaihtoehdon väliin lisättäisiin vaihtoehto “high”, joka kuvaisi normaalia vakavampaa bugia, joka ei kuitenkaan estä pelaamista.

Viimeinen ideani liittyy kortissa esiintyvien kohtien järjestykseen. Nyt kun kortin kohtia käy läpi ylhäältä alas, informaatio tulee hieman epäloogisessa järjestyksessä eli harvemmin tai ei ollenkaan käytössä olevat kohdat tulevat ennen bugin vakavuutta ja kiireellisyyttä. Koen, että vakavuus ja kiireellisyys pitäisi merkitä heti korttiin liittyvien työntekijöiden jälkeen. Näiden jälkeen voitaisiin merkitä bugin tyyppiin liittyvät tagit eli liittyykö se pelin koodiin, serveriin, grafiikkaan tai muuhun osa-alueeseen. Listan alimmaiseksi voitaisiin laittaa kohdat, joita harvemmin käytetään. Tällä muutoksella listan nopea vilkaisu olisi aikaisempaa nopeampaa, kun kaikki oleelliset tiedot olisivat peräkkäin.

Valitettavasti testikorttia tehdessäni huomasin, ettei kortille automaattisesti tulevien kohtien järjestystä voi muuttaa eli “Members”, “Tags” ja “Timeline” on lukittu

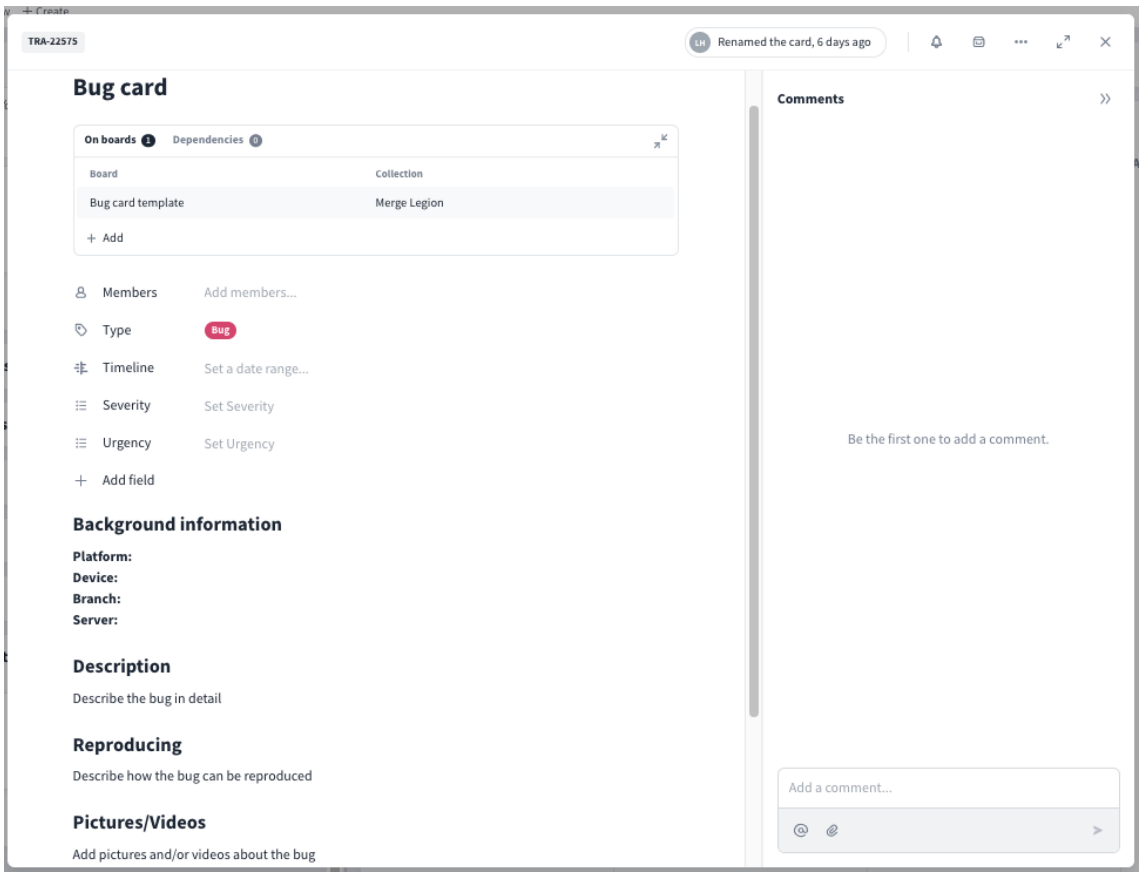
ylimmäisiksi kohdiksi, tässä järjestyksessä. Niiden nimiä voi tosin muuttaa. Muut, erikseen lisätyt kohdat, voi laittaa haluamaansa järjestykseen näiden alapuolelle. Suunnittelemani järjestys ei siis voi toteutua, mutta tein alkuperäiseen järjestykseen pienen muutoksen. En laittanut testikorttiin kohtaa “Estimation”, jonka ansiosta ennen vakavuutta ja kiireellisyyttä on yksi käyttämätön kohta vähemmän. Se on kuitenkin mahdollista lisätä alimmaiseksi kohdaksi, jos se koetaan vielä tarpeelliseksi. Järjestyksen muuttamisen lisäksi vaihdoin kohdan “Tags” nimeksi “Type”.

Näiden suunnitelmien pohjalta tein luonnoksen testipohjasta (kuva 3), jonka esittelin QA-tiimille, projektin tuottajalle ja ohjelmointivastaavalle. Sain muutaman lisäehdotuksen pohjaa varten, mutta muuten se todettiin toimivaksi. Ensimmäinen ehdotus oli se, että korttipohjaan laitettaisiin oletukseksi tagi “bug”, jotta sitä ei tarvitsisi erikseen lisätä aina, kun korttia käytetään. Toisena ehdotuksena oli se, että kortin kuvaukseen laitettaisiin oletusotsikot ja niiden alle täyttöohjeet, jotta korttia käyttävät henkilöt tietäisivät mitä kortille täytyy kirjata kuvatun bugin korjaamisen helpottamiseksi. Ehdotus tehtiin lähinnä niitä varten, jotka harvemmin täyttävät bugikortteja eivätkä täten ole varmoja siitä, mitä kaikkea bugista pitäisi ilmoittaa. Viimeinen ehdotus liittyi kortin automatisaatioon, mutta sen voisi säästää myöhemmin lisättäväksi ominaisuudeksi. Siihen liittyen ehdotettiin, että kortille voisi laittaa määräajan, jonka jälkeen siitä lähtisi ilmoitus korttiin merkityille henkilöille. Tällä huolehdittaisiin siitä, ettei yksikään kortti jää vaille huomiota ja korjausta.



KUVA 3. Luonnostelma uuden bugikorttipohjan sisällöstä

Muokkasin korttipohjaa lopultaan kahden ensimmäisen parannusehdotuksen mukaisesti. Tutkin hieman automaatioasetuksia, mutta niitä ei otettu vielä käyttöön. Tämän jälkeen hyväksytin kortin vielä viimeisen kerran ennen kuin tallensin sen korttipohjaksi ja vaihdoin sen bugilistan oletuskorttipohjaksi (kuva 4).



KUVA 4. Käyttöönotettu uusi bugikorttipohja

Korttipohjan käyttöönotto ei sujunut täysin ongelmitta. Bugilista ei suoraan hyväksynyt uuden bugikorttipohjan muokattuja Severity -ja Urgency-kohtia, joten minun piti lisätä ne uudelleen kyseiselle listalle. Minun ei tarvinnut muokata jokaista olemassa olevaa bugikorttia, vaan riitti, että tein muutoksia yhteen korttiin, jonka jälkeen kyseiset kohdat päivittyivät automaattisesti kaikkiin kortteihin.

6 POHDINTA

Pelitestaaminen on oleellinen osa pelinkehitysprosessia, jotta valmis tuote olisi mahdollisimman korkealaatuinen. Pelitestaamisessa ei keskitytä pelkästään pelillisiin ominaisuuksiin, vaan sen aikana tarkistetaan myös graafinen ilme, käyttäjäkokemus ja pelattavuus. Testaajan työ on ajoittain haastavaa, etenkin projektin edetessä, sillä testaaja voi sokeutua tietyille epäkohdille, jolloin niiden korjaaminen viivästyy. Siksi onkin tärkeää kerätä palautetta pelistä myös testaus- ja kehitystiimien ulkopuolelta, jolloin sokeat kohdat voidaan havaita ja korjata ajoissa. Testaajan tulee myös jatkuvasti kehittää itseään esimerkiksi kilpailevien yritysten pelejä pelaamalla löytääkseen niiden avulla parannuskohtia omasta projektistaan.

Työllä saavutettiin haluttuja tuloksia; toimeksiantajan testausprosessia parannettiin uuden bugikorttipohjan avulla, joka tekee bugien merkitsemisestä helpompaa myös niille, jotka eivät käytä pohjaa päivittäin. Yhteistyön parantamiseen liittyvä ehdotus vietiin myös eteenpäin käsiteltäväksi ja sitä voidaankin käyttää pelitestausprosessin mahdolliseen jatkokehittämiseen tulevaisuudessa. Bugikorttipohjan parantamista olisi voitu tarkastella jo aikaisemminkin, sillä entinen pohja otettiin edellisestä peliprojektista suoraan nykyiseen projektiin. Projektin vaihtuessa muutosten tekeminen ja niiden käyttöönotto olisi ollut luontevampaa kuin kesken projektin.

LÄHTEET

Game-Ace. 2021. What Are the Main Types of Game Testing? Your Questions Answered. Viitattu 27.7. https://game-ace.com/blog/types-of-game-testing

Hogan, T. & Broadbent, C. 2016. The Ultimate Start-Up Guide. Marketing Lessons, War Stories, and Hard-Won Advice from Leading Venture Capitalists and Angel Investors. E- kirja. New Jersey: The Career Press Inc. Viitattu 8.7. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/the-ultimate-start-up/9781632650733/

IronSource. n.d. Soft Launch. Viitattu 8.7.2022. https://www.is.com/glossary/soft-launch/#:~:text=A%20soft%20launch%20refers%20to,world%20interactions%20with%20their%20app

Knezovic, A. 2022. How to Soft Launch a Mobile Game? Best Strategies for 2021. Udonis. Viitattu 8.7.2022. https://www.blog.udonis.co/mobile-marketing/mobile-games/soft-launch-mobile-game

Kramarzewski, A. & De Nucci, E. 2018. Practical Game Design. E-kirja. Birmingham: Packt Publishing. Viitattu 24.8. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/practical-game-design/9781787121799/

Lancaric, M. 2018. Ultimate Guide to Soft Launch your Mobile Game. Viitattu 22.8. https://www.gamedeveloper.com/business/ultimate-guide-to-soft-launch-your-mobile-game

Lawley, B. & Schure, P. 2017. Product Management For Dummies. E-kirja. UK: For Dummies. Viitattu 8.7. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/product-management-for/9781119264026/

McCann, T. 2011. The Art of the App Store. The Business of Apple Development. E- kirja. Indianapolis: John Wiley & Sons Inc. Viitattu 8.7. Vaatii käyttöoikeuden. https://learning.oreilly.com/library/view/the-art-of/9781118221129/

Seka, M. 2021. Game Testing 101 - The Ultimate Beginner's Guide. Viitattu 26.7. https://www.softwaretestingmaterial.com/game-testing/

Schlossberg, M. 2022. Soft Launch vs Hard Launch - What's the Difference? Capterra. Viitattu 15.7. https://blog.capterra.com/soft-launch-vs-hard-launch/

Schultz, C. & Bryant, R. 2017. Game Testing. All in One. 3. painos. Dulles: Mercury Learning and Information.

Testbytes. 2022. 9 Different Types of Game Testing Techniques. Viitattu 26.7.2022. https://www.testbytes.net/blog/types-of-game-testing/