Karl Peter Skjelvik, Trond Ugelstad, Martin Wangen

Utvikling av nettleserspill

Bacheloroppgave i Dataingeniør
Veileder: Nils Tesdal
Mai 2019

Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU
Kunnskap for en bedre verden

Karl Peter Skjelvik, Trond Ugelstad, Martin Wangen

Utvikling av nettleserspill

Bacheloroppgave i Dataingeniør
Veileder: Nils Tesdal
Mai 2019

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU
Kunnskap for en bedre verden

Forord

Ved utforming av oppgaven bestemte vi oss for å utvikle et produkt hvor vi kunne ha aktive brukere. Vi kom frem til at det å lage et underholdningsprodukt ikke trenger å være revolusjonerende, samtidig som det ikke må utkonkurrere et annet produkt for å ha brukere. Ved å utvikle et spill sørger vi for at vi får videreutvikle et produkt med aktive brukere og kan utvikle basert på tilbakemeldinger fra disse.

Ved å utvikle et enkelt spill og dele med venner fikk vi samlet en mindre brukerbase mens vi videreutviklet. Herfra utviklet vi parallet med innsamling av tilbakemeldinger til avhandlingen. Å basere oss på tilbakemeldinger fra brukere gjorde at vi måtte utnytte flere forskjellige innhentingsmetoder å sikre å få tilbakemeldinger. Oppgaven vår ble da å se på forskjellige tilnærminger til innhenting av tilbakemeldingene og hvor nyttige disse er. Å kunne samle gode tilbakemeldinger vil være en viktig del i vår framtid som systemutviklere. Vi vil gjerne benytte denne anledningen til å takke Kantega AS avdeling Trondheim.

Ved å stille med kontorplasser, server- og databasekapasiteter og faglig assistanse har de lagt til rette for utmerkede forhold for oss under skriving av denne oppgaven. En spesiell takk til våre veiledere, Håvard Wigtil fra Kantega og Nils Tesdal, universitetslektor ved NTNU, for tilrettelegging, veiledning og all hjelp og støtte.

Karl Peter Skjelvik Trond Ugelstad Martin Wangen

20. mai 2019, Trondheim

Oppgavetekst

Uvtvikle et nettleserbasert spill, i sjangeren ”Incremental game”, og få ut et MVP så fort som mulig for å begynne å bygge en brukerbase. Utviklingen skal følge disse retningslinjene:

- MVP må være bra nok (viable) før det slippes ut til brukere. Hvor bra er bra nok?
- Få mye tilbakemelding fra brukerne. Hvordan få tilbakemeldinger uten at brukerne føler at de utfører en jobb?
- Lytte til brukerne, og velge ut nyttige tilbakemeldinger. Hva er bra for spillets videre utvikling og hva kan være destruktivt?
- Håndtere nedetid under oppdateringer. Hva aksepteres av en spiller?
- Opprettholde balanse etter oppdatering. Hvor ofte må man ta backup, og hvor lenge tar man vare på dem, for å være trygg på at brukerne ikke opplever for store tap om noe går galt?

Litt om konseptet:
Spillet skal være en MMORPG (Massively multiplayer online role-playing game) der spiller er en person i Romerriket, som eier landområder og ressurser som må styres og vedlikeholdes. Brukerne logger inn på en nettside og utfører regelmessige handlinger, for eksempel å plante frø til avlinger, utvinne ressurser fra gruver, og gjøre handel med andre spillere på markedet.

For mer spesifikke krav til systemet, se vedleggene Kravdokumentasjon, Systemdokumentasjon og Visjonsdokument.

Sammendrag

I denne rapporten skal vi se på hvordan ulike måter å hente inn tilbakemeldinger fra brukere påvirker kvantitet og kvalitet på tilbakemeldingene. For å finne ut dette har vi utviklet et spill som spilles i nettleser, og brukt fem ulike metoder for å samle inn tilbakemeldinger. Vi har samlet inn over 200 tilbakemeldinger, som vi har kategorisert og analysert for å finne sammenheng mellom metode og resultat. Våre resultater peker mot at å legge mye jobb inn i å samle tilbakemeldinger, og å gjøre det lett for brukere å gi tilbakemeldinger, gir en betydelig økning i antall tilbakemeldinger. Vi har sett at brukertester kan gi innsikt man ikke får av andre typer tilbakemelding, og at hvilken type bruker som gir tilbakemeldinger kan påvirke kvaliteten.

Innholdsfortegnelse

Figurer

Introduksjon og relevans

I denne avhandlingen skal vi se på hvordan måten brukere blir oppfordret til å gi tilbakemeldinger på påvirker hva det blir gitt tilbakemeldinger på og om innholdet i disse tilbakemeldingene er forskjellige eller ikke.

Vi ville se nærmere på dette fordi tilbakemeldinger fra brukere er noe som vil påvirke vår hverdag som systemutviklere, men som ingen av oss har noen tidligere erfaringer med.

1.1 Metoder for tilbakemeldinger

For å komme i mål med denne avhandlingen var vi avhengige av å få tak i tilbakemeldinger på forskjellige metoder. Det ble derfor benyttet flere forskjellige måter å få tilbakemeldinger på.

Den ene metoden var å ha to knapper inne i spillet som sendte brukeren videre til Reddit-sider. En av disse knappene var for tilbakemeldinger angående feil i spillet, ”Bug Submissions”, mens den andre knappen var for hvis brukeren ville komme med nye idéer for nytt innhold i spillet, ”Content Suggestions”. En annen metode ble benyttet av personer som vi har et personlig forhold til, som ga oss tilbakemeldinger muntlig og via private chat-er som Facebook Messenger. I tillegg til dette ble også Discord brukt ved senere anledninger, da spesielt av personer som er mindre kjent.

Enda en metode som vi brukte for å få tilbakemeldinger var å holde brukertester der nye brukere fikk prøve starten av spillet, slik at vi fikk observere hvordan det var å sette seg inn i spillet. Vi brukte også en metode der vi lagde spørreundersøkelser inne i spillet for å spørre brukerne. Med denne metoden fikk de opp dialogbokser der de ble bedt om å sende inn sine tilbakemeldinger eller svare på noen spørsmål fra oss, for eksempel om de fant feil i spillet eller om det var noe spesielt de gjerne skulle hatt av innhold eller endringer. Denne metoden for å få tilbakemeldinger på ble gitt til brukerne etter at de hadde utført spesifikke deler av spillet. Her ga vi også brukerne en belønning for å hjelpe oss, i form av ”coins” i spillet.

1.2 Minste levedyktige produkt

Et av aspektene som vi måtte tenke litt på når det gjaldt bacheloroppgaven var når vi skulle slippe MVP-en av spillet. Her måtte vi tenke litt annerledes enn det vi ville gjort hvis det ikke var et spill som vi lagde. Grunnen til dette er fordi et spill er et underholdnings-

produkt, som brukere egentlig ikke må bruke noen gang, og ikke et bruksprodukt, som brukere er nødt til å bruke for at de for eksempel skal kunne klare å gjøre en eller flere deler av arbeidet sitt. På grunn av dette måtte vi prøve å balansere når vi følte at MVP-en var klar til å slippes ut til brukerne.

Det vi fryktet var at hvis spillet ble sluppet for tidlig, så ville kanskje brukerne kunne bli ferdig for fort og dermed gå lei av spillet og legge det vekk, før vi rakk å komme ut med mer innhold. Med tanke på at vi gjerne vil ha aktive brukere av spillet og at vi måtte ha tilbakemeldinger av disse brukerne til denne avhandlingen, ville det ikke vært gunstig hvis brukerne følte at de ikke orket å fortsette å spille spillet fordi det hadde for lite innhold og dermed kanskje ikke følte det som nødvendig å komme med noen tilbakemeldinger heller. Samtidig som at vi måtte passe på å ikke slippe spillet for tidlig, måtte vi også passe på at vi slapp MVP-en tidsnok til at vi kunne få noen brukere til å spille det, slik at de hadde mulighet til å rekke å prøve spillet og finne ut hva de ville gi tilbakemeldinger på, som vi kunne benytte oss av til videre utvikling av spillet.

1.3 Prosess

Denne oppgaven ble gjennomført ved å bruke aspekter tatt fra Lean. Lean ble ikke fulgt helt ut fordi produktet som ble laget var et underholdningsprodukt og måtte på grunn av dette være laget på en slik måte at det gikk fortere å få laget nytt innhold til spillet. Dette innebærer at metodene som ble laget, spesielt frontend, ble laget for å være enkle å gjenbruke. Dette bryter en del med i hvert fall Lean Startup, men det ble bestemt at dette var den beste måten det kunne bli gjort på når oppgaven ble startet.

1.4 Hvorfor

Tilbakemeldinger fra aktive brukere av et produkt er viktig for produktutviklingen. Brukerne er de som skal benytte seg av produktet og dermed er det viktig å vite hva de synes om produktet, både det negative og det positive.[1] Konstant innhenting av tilbakemeldinger er det som gjør smidige metodikker Lean. Tilbakemeldinger er avgjørende for å vite hvordan man kan øke verdien til et produkt.[2]

1.5 Videre i denne avhandlingen

Videre i avhandlingen vil vi først ta for oss teori rundt brukertesting, og så MVP og Lean utvikling. Her vil vi prøve å få satt studien vår inn i et eksisterende teoretisk rammeverk. I dette kapittelet vil vi også gjøre rede for de spesifikke teoriene og begrepene som vi vil anvende senere i avhandlingen.

I kapittelet Valg av teknologi og metode vil vi forklare hvorfor vi har valgt metodene vi har gjort i tillegg til hva vi ser for oss kan være negative sider ved de valgene. Vi vil også forklare hvordan vi har samlet inn dataene som vi har brukt i avhandlingen. I Resultater redegjør vi og forklarer resultatene som vi har fått i løpet av bacheloroppgaven.

De to siste kapitlene i avhandlingen handler om å diskutere resultatene og konkludere funnene våre. I Diskusjon drøfter hva vi har gjort i studien, samt hva som var sterke og svake sider ved metodene som vi bestemte oss for. I siste kapittel kommer vi med en konklusjon på problemstillingen og en liten oppsummering av det vi har kommet fram til under bacheloroppgaven og avhandlingen, samt noen ord om videre arbeid.

Akronymer og forkortelser

Agile	Iterativ utviklingsmetodikk med fokus på å respondere på endringer i forholdene
Backend	Tjenersiden av en klient-tjener arkitektur
Client-side prediction	Frontend predikerer backend og fortsetter i den tro at de er synkronisert
Continuous Development	Iterativ prosess hvor fungerende kode settes i produksjon umiddelbart etter testing
Cross-site-scripting	Injisering av kode på nettsteder som så vises og kjører på andre klienters maskiner
Developer sandbox	Eget miljø til hver utvikler som ikke påvirkes av andre i prosjektet
Discord	Gratis kommunikasjonstjeneste. Mye brukt i spillmiljøer
Frontend	Klientsiden av en klient-tjener arkitektur
Hibernate	ORM-verktøy for Java
Høy kohesjon	Relatert kode er koblet tett sammen
Incremental game	Spill hvor en over tid tjener opp ressurser som brukes for å tjene mer ressurser
Issue board	Tavle med oppgaver med fordeling blant utviklere
Java	Programmeringsspråk
JAX-RS	Verktøy for å lage tjenerprogramvare med RESTful arkitektur
Jenkins	Software for tjener-automatisering, continuous integration og continuous delivery
jQuery	Javascript-bibliotek
Kanban	Metode for smidig utvikling
Kotlin	Programmeringsspråk
Lean	Utviklingsmetodikk med fokus på å unngå waste
Lean startup	Agil tilnærming til utvikling av oppstartsbedrift
Løs kobling	Forskjellige moduler avhenger lite av hverandre
MVP	Minimum Viable Product
ORM	Object-relational mapping, kobler relasjonsdatabase til objekter
Pivot	Strukturert kurskorreksjon designet for å teste en ny grunnleggende hypotese om produktet, strategi og vekstmotor

Preproduction sandbox	Miljø hvor ny funksjonalitet testes før det settes i produksjon. Miljøet er så likt produksjonsmiljøet som mulig
React	Javascript-bibliotek
Reddit	Nettside/Forum
Redux	Javascript-bibliotek for globale variabler
Scrum	Metode for smidig utvikling
Spring Boot	Verktøy for å lage tjenerprogramvare med RESTful arkitektur
SQL-injections	Injisering av SQL-kode i input-felter for å påvirke resultatet av en operasjon
TypeScript	Typesatt JavaScript
Waste	Uønsket eller ubrukelig avfall i en prosess

Teori

Dette kapittelet opplyser kort om brukertester og litt bakgrunn rundt Lean. Det opplyses om Lean, ettersom det er via Lean produktet er blitt laget.

3.1 Tilbakemeldinger

3.1.1 Brukertester

Brukertester kan spare brukerne for mye frustrasjon, samtidig som det gir en målbar effekt på sluttproduktet. Til og med en enkel papirprototyp kan hjelpe med å få mer fornøyde kunder, økt salg via nett, eller mer effektive og fornøyde ansatte.

Å kjøre en brukertest er billigere enn å ikke gjøre det. Det er svært nyttig, men likevel velges det altfor ofte bort. Noen ganger på grunn av mangel på tid eller penger, andre ganger fordi man selv stempler løsningen som brukervennlig på vegne av sluttbrukerne.

Hvis man skal kjøre en brukertest så bør den kjøres så tidlig som mulig. Venter man til løsningen nesten er ferdig så vil endringer koste mer tid og penger.

Ved en brukertest så bør man kjøre brukertestene på de representative brukerne til produktet. De brukerne som man kjører testene på må forsøke å løse reelle oppgaver og gjerne bør dette også gjøres i en realistisk setting. Hvis man bestemmer seg for å kjøre brukertester på noen som er involvert i prosjektet, om det er deltakerne, ledere eller noen som betaler for utviklingen, kan dette gi feilaktige testresultater. Samtidig som man bør teste på brukerne av produktet er det viktig at disse brukerne tester det man ønsker at de skal teste ut. Det er derfor viktig at man tenker over hvorfor man gjennomfører brukertesten.

Når man gjennomfører brukertesten er det viktig at man gjennomfører testen på en upartisk måte. Man skal benytte seg av anledningen til å lære mest mulig om hvordan løsningene blir oppfattet av brukerne. For at man skal klare dette kan man for eksempel be brukeren om å tenke høyt hvordan de løser oppgavene. Under testen skal det ikke gis hjelp, samt at de oppfølgingsspørsmålene man eventuelt har etter testen må være nøytrale. Man skal ikke løse problemene under testen, man skal kun samle inn informasjon som man trenger for å jobbe videre med løsningen, og derfor skal heller ikke oppfølgingsspørsmålene lede brukeren i en retning.

Samtidig må man tenke gjennom hva man ønsker å teste. Om man tester for eksempel en nettside og intranettet på datamaskinen kan den se og fungere annerledes enn hvis man tester den på mobil.[3]

3.2 Lean systemutvikling

Lean systemutvikling, direkte oversatt til slank utvikling, er en smidig metodikk, eller tankegang, som er basert på Toyotas produksjonssystem og går ut på å unngå waste og fokusere på verdiskapning. Utrykket stammer fra boka ”Lean Software Development: An Agile Toolkit” av Tom og Mary Poppendieck, 2003[4]. Boka beskriver og omdøper prinsipper innen Lean, og ses på som grunnlaget for Lean systemutvikling.

3.2.1 Lean produksjon

Lean produksjon, ofte bare Lean, er en systematisk metode for å minimere uønsket eller ubrukelig ”avfall”, bedre kjent som waste, uten at det går på bekostning av produktivitet eller verdi. Konseptet går ut på å kvitte seg med alt som ikke er med på å skape verdi, hvor verdi er sett på som det en er villig til å betale for, sett fra kundens perspektiv. De syv opprinnelige waste, fra Toyotas produksjonsfilosofi, er [5]:

- Transport (flytting av produkter som ikke er nødvendig for produksjonen)
- Inventar (alle komponenter, enten ferdige eller under produksjon, som ikke prosseseres)
- Bevegelse (mennesker og/eller utstyr som flyttes mer enn høyst nødvemndig for en effektiv process)
- Venting (alle stopp i produksjonslinja, enten grunnet flaskehalser eller bytte av personale)
- Overproduksjon (større produksjon enn etterspørsel)
- Overprossesering (dårlige verktøy eller produksjonsplan som akkumulerer unødvendig arbeid)
- Defekter (søk etter og utbedringer av defekter)

3.2.2 Lean systemutviklingsprinsipper

[6]

Eliminate waste

Å fjerne alt som ikke skaper verdi for kunden er et av nøkkelpunktene i Lean. I Mary og Tom Poppendiecks bok beskriver de hvilke waste som må elimineres i programvareutvikling. Her beskrives waste som unødvendig kode og funksjonalitet, å påbegynne mer

enn det som kan fullføres, forsinkelser i utviklingsprosessen, utydelige og endrende kravspesifikasjoner, byråkrati, dårlig kommunikasjon, delvis ferdig arbeid, defekter og kvalitetsproblemer og bytte av oppgaver. En studie av waste ved programvareutvikling kom frem til ni typer waste. Tabellen under er fra publiseringen av studiet og viser de ni med beskrivelse og observasjoner[7].

Waste	Description	Observed
Building the wrong feature or product	The cost of building a feature or product that does not address user or business needs.	User desiderata (not doing user research, validation, or testing; ignoring user feedback; working on low user value features) Business desiderata (not involving a business stakeholder; slow stake-holder feedback; unclear product priorities)
Mismanaging the backlog	The cost of duplicating work, expediting lower value user features, or delaying necessary bug fixes	Backlog inversion Working on too many features simultaneously Duplicated work Not enough ready stories Imbalance of feature work and bug fixing Delaying testing or critical bug fixing Capricious thrashing
Rework	The cost of altering delivered work that should have been done correctly but was not.	Technical debt Rejected stories (e.g. product manager rejects story implementation) No clear definition of done (ambiguous stories; second guessing design mocks) Defects (poor testing strategy; no root-cause analysis on bugs)
Unnecessarily complex solutions	The cost of creating a more complicated solution than necessary, a missed opportunity to simplify features, user interface, or code	Unnecessary feature complexity from the user's perspective Unnecessary technical complexity (duplicating code, lack of interactiondesign reuse, overly complex technical design created up-front)
Extraneous cognitive load	The costs of unneeded expenditure of mental energy.	Suffering from technical debt Complex or large stories Inefficient tools and problematic APIs, libraries, and frameworks Unnecessary context switching Inefficient development flow Poorly organized code
Psychological distress	The costs of burdening the team with unhelpful stress.	Low team morale Rush mode Interpersonal or team conflict
Waiting/multitasking	The cost of idle time, often hidden by multi-tasking.	Slow tests or unreliable tests Unreliable acceptance environment Missing information, people, or equipment Context switching from delayed feedback
Knowledge loss	The cost of re-acquiring information that the team once knew.	Team churn Knowledge silos
Ineffective communication	The cost of incomplete, incorrect, misleading, inefficient, or absent communication.	Team size is too large Asynchronous communication (distributed teams; distributed stakeholders; dependency on another team; opaque processes outside team) Imbalance (dominating the conversation; not listening) Inefficient meetings (lack of focus; skipping retros; not discussing blockers each day; meetings running over (e.g. long stand-ups))

Build Quality In

Build Quality In handler om ha fokus på kvalitet i kode og systemet som helhet. Det kan virke åpenbart at de fleste ønsker kvalitet, men sannheten er at mange team skaper waste i søken etter kvalitet. For mye testing og logging av defekter vil igjen virke mot sin hensikt og skape waste. I tillegg til å få en god forståelse for problemdomenet og opprettholde god kommunikasjon med kunden er det i nyere tid flere verktøy/metoder som har vist seg å gi suksess. Noen av de mest populære er parprogrammering, testdreven utvikling, inkrimentell utvikling, minimering av venting og automatisering av oppgaver utsatte for menneskelig feil.

Create knowledge

Nok et prinsipp som kan virke åpenbart, men vanskelig å gjennomføre. I stedet for å dokumentere eller planlegge i detalj burde en se på andre metoder for kunnskapsdeling. Blant dem er parprogrammering, code reviews, kommentering av kode og sesjoner med kunnskapsdeling. Disse sesjonene med kunnskapsdeling kan legges til slutten av en sprint og en bør ta med kunden for å gi både utviklere og kunden en best mulig forståelse av problemdomenet.

Defer Commitment

Å utsette å ta avgjørelser handler om å ikke ta forhastede avgjørelser. Ved smidig utvikling skal det kunne gjøres endringer basert på ny tillært kunnskap om problemdomenet. Vi ønsker derfor å utsette avgjørelser så lenge som mulig slik at vi har best mulig grunnlag når avgjørelsene tas.

Deliver fast

Prinsippet handler ikke om å bli raskt ferdig, men å levere verdi til kunden så tidlig som mulig. I størst grad må en se etter hva som øker leveringstiden. Unngå å tenke for langt frem eller å lage overkompliserte løsninger. Å levere raskt i Lean betyr å lage en enklest mulig løsning som kan leveres for å få tilbakemeldinger så tidlig som mulig.

Empower the team

Prinsipper om å gi makt til teamet handler i stor grad om å stole på utviklerene og la dem ta egne avgjørelser. En leders oppgave er i større grad å motivere enn å fortelle utviklerene hvordan og hva de skal utvikle. Lean bygger på det smidige prinsippet ”finn flinke utviklere og la dem gjøre jobben sin”.

Optimize the whole

I deres bok, beskriver Tom og Mary Poppendieck to vonde sykluser. Den første syklusen er utviklere som nærmer seg levering og får dårlig tid. Utviklerene føler seg presset til å levere raskt og prioriterer da levering i tide fremfor kvalitet. Koden er da gjerne full av defekter som må utbedres etter levering. Dette går da på tida til neste levering og for å nå neste frist må det igjen gå på bekostning av kvalitet.

Den andre syklusen baseres på overbelasting av testere. Har testerene for mye å gjøre kan det gå lang tid før koden testes og de kan gi tilbakemeldinger på koden. Utviklerene fortsetter å utvikle med den defekte kodebasen som fører til flere defekter og igjen krever mer testing.

Optimize the whole handler om å fjerne disse destruktive syklusene. Ved å ha en god forståelse av problemdomenet, teamet og dets kapasitet vil en kunne finne en balanse i utviklingen som gir mest mulig verdi over tid.

3.2.3 Lean Startup

I 2011 slapp Eric Ries, en av grunnleggerene bak den svært populære sosiale platformen IMVU, boka The Lean Startup.[8] I likhet med Lean programvareutvikling handler Lean Startup om å eliminere waste og maksimere produsering av verdi tidlig i oppstarten for å gi selskapet best mulig grunnlag for suksess uten å bruke betydelige midler eller tid. Brukertilbakemeldinger gjennom utviklingen er grunnstenen i Lean Startup og er helt essensielt for å bestemme retning for videre utvikling og for å sikre at selskapet ikke bruker tid og penger på noe forbrukere ikke er interesert i å bruke penger på.

Pivot

En pivot er en «strukturert kurskorreksjon designet for å teste en ny grunnleggende hypotese om produktet, strategi og vekstmotor».[8, s. 149] Så snart en får tilbakemelding, enten positiv eller negativ, handler det om å peile seg inn på det produktet som har størst sjanse for å være attraktivt for kundene. Utrykket kommer fra basketball hvor en pivot er når en dreier kroppen med en fot, mens den andre er i bakken. Fra sammenligningen har vi en fot i bakken som er det vi har lært og tar med oss videre, mens vi forandrer en annen ting og som i helheten forandrer retningen til produktet som utvikles. Møter en motgang eller stagnerer utføres en ny pivot.

Minimum viable product (MVP)

En MVP er første testbare versjon av et produkt. Så snart man har en versjon som er mulig å teste gjøres denne tilgjengelig for en gruppe brukere og en samler data. Her kan man sjekke om det er interesse for produktet som det er eller om man må utføre en pivot. MVP

kan være så lite som en skyggeknapp som ikke gjør noe som helst annet enn å registrere hvor mange som trykker. MVP skal helst være bare akkurat det som må til for å finne ut om planen for produktet er riktig eller ikke.

Learn-Build-Measure

Ideas → Build → Product → Measure → Data → Learn, det er syklusen for Lean Startup. Målet er å komme seg gjennom syklusen flest mulig ganger så tidlig som mulig. For hver gjennomgang får man nye tilbakemeldinger og muligheten til å utføre en pivot eller å fortsette i samme retning.

Figur 3.1: Syklusen i Lean Startup. Hentet fra `http://leanstartup.com/`

Valg av teknologi og metode

Dette kapittelet tar for seg prosessen som ble brukt under bacheloroppgaven, og det blir beskrevet hvilke teknikker som ble bestemt å bruke for å produsere produktet.

4.1 Prosess

Lean Startup

I all hovedsak har prosessen basert seg på Lean Startup. Vi hadde som mål å utvikle en MVP så hurtig som mulig, men i stedet for å utvikle minste testbare produkt valgte vi å utvikle minste levedyktige produkt. Realistisk sett antok vi at de venner og kjente vi nådde ut til var de som kom til å spille aktivt gjennom hele utviklingsprossessen. Ved å utvikle en selvstendig løsning for MVP-en sikret vi at vi ikke mistet dyrebare brukere og derav mistet den mest verdifulle kilden for tilbakemeldinger.

Sandbox

Et av verktøyene vi har brukt er et eget miljø hvor vi kan bygge, kjøre og teste ny funksjonalitet. Det kan diskuteres i hvilken grad vi har hatt Developer sandbox. En burde gjerne ha egne databasekapasiteter per utvikler, men da vi ikke har sett nødvendigheten i dette og sjelden ble påvirket av hverandres oppdateringer i databasen har vi valgt å dele på to databaser. Vi har dog brukt en Preproduction sandbox.

Continuous Development

Vi valgte å bruke Continuous Development. Vi ønsket å få ut alt nytt innhold for å få testet og få tilbakemeldinger så snart som mulig. Ved å sette alt i produksjon med en gang det er utviklet og testet så vi tidlig hvordan det ble tatt imot og om vi skulle fortsette i samme retning eller pivot. Det var veldig nyttig og gjorde oss fleksible med tanke på utvikling basert på tilbakemeldinger.

Issue board

For å holde styr på hvem som arbeidet med hva brukte vi Issue board, som en forenklet versjon av Kanban board. Vi valgte å dele spillet opp i større hoveddeler, hvor hver del ble satt opp som en issue og tildelt en av utviklerene. På denne måten holdt vi godt styr på hvem som gjorde hva, hindret dobbeltarbeid og unngikk en del waste som følge av for detaljerte issues. I Lean Startup regnes for detaljerte Kanban Boards som waste, og når

vi har åtte sider med user stories som funksjonelle krav akkumulerer vi mye waste om alt skal gjøres om til issues som håndteres på detaljnivå.

4.2 Tilbakemeldinger

MVP-en ble stadig utviklet etter utgivelse, både ut i fra hva vi visste at fortsatt trengtes for at spillet skulle være brukbart over en lengre periode og hva brukerne ga tilbakemeldinger på. Spillet ble sluppet i flere steg. Først ble det sluppet til bare de aller nærmeste vennene, for senere å bli sluppet til en bredere folkemasse.

Etter det første slippet ble tilbakemeldinger tatt i mot muntlig eller via private chat'er, som for eksempel Facebook Messenger. Senere ble det også tatt i bruk Reddit og Discord.

Figur 4.1: Bilde av knappene som linker til Reddit

I tillegg til disse innhentingsmetodene ble det også tatt i bruk andre innhentingsmetoder, brukertester og et tilbakemeldingsverktøy innad i spillet.

Mot slutten av prosjektet henvendte vi oss direkte til utviklere og spillere av lignende spill, for konkrete tilbakemeldinger om vårt spill.

Det ble valgt å ha forskjellige innhentingsmetoder for å se om det var forskjell på innholdet i tilbakemeldingene hvis brukerne kunne gi beskjed på forskjellige måter.

4.2.1 Kategorisering

Etter å ha samlet inn tilbakemeldinger så kategoriserte vi dem i 4 kategorier, ”veldig nyttig”, ”nokså nyttig”, ”nokså unyttig”, og ”fullstendig unyttig”.

Med ”veldig nyttig” menes tilbakemeldinger som vi enten umiddelbart handler utifra eller definitivt kommer til å ta hensyn til i fremtiden

Med ”nokså nyttig” menes tilbakemeldinger som kan komme til nytte, men som gjerne ikke er akkutte eller som vi vil ha fleres mening om før vi bestemmer oss for om vi skal gjøre noe basert på dem

Med ”nokså unyttig” menes tilbakemeldinger som vi sannsynligvis ikke kommer til å gjøre noe med, men som har noen tanker eller idéer det går an å ta med videre. Dette kan også være tilbakemeldinger som ikke relaterer direkte til spillet vårt eller tilbakemeldinger som vi tviler på sannheten i, enten at de er laget for å tulle med oss eller at de mener å ha opplevd noe i spillet som vi ikke finner noe bevis for har skjedd.

Med ”fullstendig unyttig” menes tilbakemeldinger helt uten nytteverdi, herunder tomme tilbakemeldinger, ufullstendige setninger, eller tilbakemeldinger som ikke har noe med spørsmålet å gjøre i tilbakemeldingsverktøyet.

4.2.2 Brukertester

Det ble foretatt to brukertestrunder[1] i løpet av oppgaven. Brukertestene skulle se på starten av spillet, om nye brukere syntes at det var enkelt eller vanskelig å ta i bruk spillet. Testene ble tatt på forskjellige personer, slik at det var mulig å se om det var noen endringer på tilbakemeldingene fra testpersonene.

Fra første brukertestrunde til andre brukertestrunde ble det foretatt endringer av spillet, basert på tilbakemeldingene fra brukertestene og de tilbakemeldingene som ble mottatt fra de andre innhentingsmetodene.

Første brukertestrunde

Den første brukertestrunden[2] ble testet på personer som ikke hadde noen tidligere erfaringer med lignende spill. Dette ble gjort for å se om det var intuitivt nok for alle, også nye brukere som ikke hadde erfaring med lignende spill.

[1] Se vedlegg Brukertester for testplan

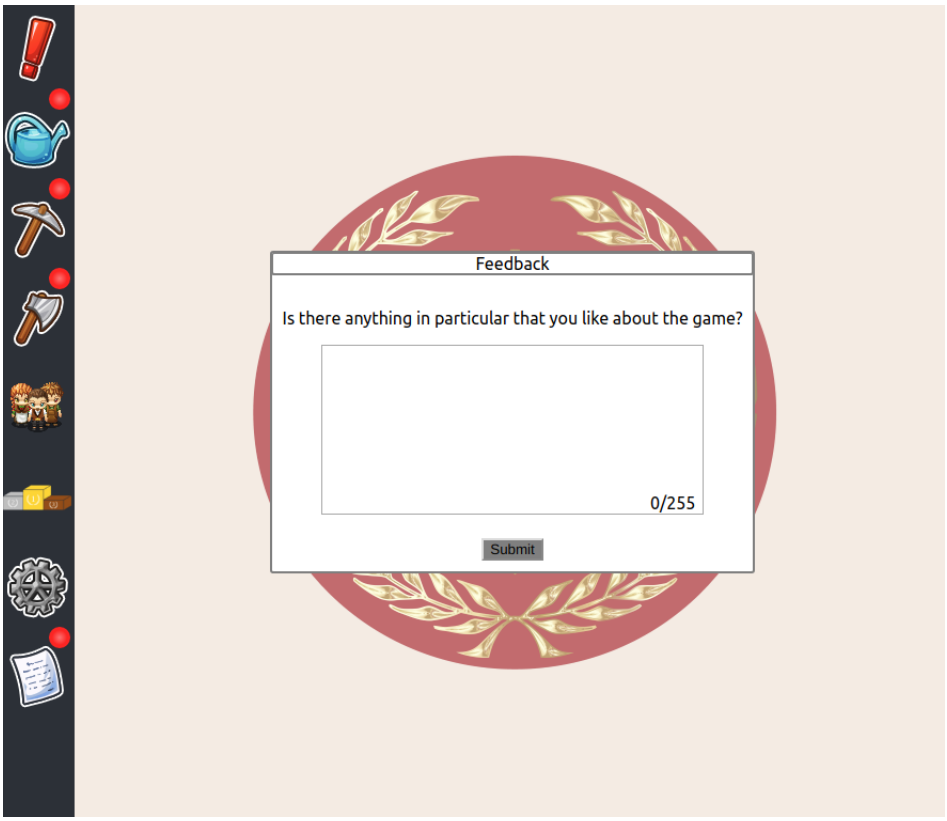
[2] Se vedlegg Brukertester for sammendrag av første brukertestrunde

Andre brukertestrunde

Den andre brukertestrunden[3] ble utført på andre personer enn de som ble testet i den første brukertestrunden. I tillegg til dette hadde heller ingen i denne brukertestrunden noen tidligere erfaringer med lignende spill.

4.2.3 Tilbakemeldingsverkøy i spillet

Tilbakemeldingsverktøyet innad i spillet var ikke klart da spillet ble sluppet første gang, men ble lagt til etter at spillet hadde vært ute en stund. Det ble laget to sett med spørsmål, ett som brukerne fikk når de var ferdig med den første oppgaven i spillet og ett som brukerne fikk når de var ferdig med den tredje oppgaven. De brukerne som hadde gjort den tredje oppgaven før tilbakemeldingsverktøyet ble lagt til i spillet, fikk bare det siste av de to settene. Å få det første settet med spørsmål vil ta en ny bruker et par timer, mens det andre settet med spørsmål tar omtrent fem til syv dager.



Figur 4.2: Utklipp av dialogboksene

Noe som skiller tilbakemeldingsverktøyet fra de andre metodene er at brukerne ble lokket med en belønning i spillet, som gir en motivasjon for å gi tilbakemelding utover å bare hjelpe oss. Belønningen som brukerne fikk etter å ha svart på spørsmålene ble tilpasset etter hvor langt de hadde kommet for å få tilgang til settet med spørsmål.

[3] Se vedlegg Brukertester for sammendrag av andre brukertestrunde

Det ble vurdert om det skulle være kvalitetskrav på tilbakemeldingene, og manuell gjennomgang før brukerne fikk belønningen, men det ble til at brukerne fikk belønningen umiddelbart når de var ferdige med settet, uavhengig om tilbakemeldingene var nyttige eller ikke. Dette ble gjort fordi tiden ikke strakk til for manuell gjennomgang av hver tilbakemelding fra hver bruker. Samtidig ble det skrevet i introduksjonen til spørreundersøkelsene at dette gjaldt en bachelor-avhandling og håpet var da at de var villige til å sette av to minutter til å gi nyttige tilbakemeldinger.

4.3 Valg av teknologi

Valget av hvilken teknologi som skulle brukes i bacheloroppgaven havnet på Kotlin og Spring Boot backend og TypeScript, React og Redux frontend, samtidig som det ble valgt å brukte Hibernate for databasesamarbeid. Ingen på gruppa hadde erfaring med verken Kotlin, Hibernate eller Redux, men vi hadde såvidt hadde vært borti TypeScript, React og Spring Boot.

Fordi dette prosjektet teknologimessig ikke hadde trengt å skille seg så mye fra tidligere systemutviklingsprosjekter vi har gjort, ville vi bruke teknologier som ingen av oss var veldig kjente med, for å få erfaring med mer variert teknologi. Vi endte på Kotlin backend fordi det er veldig i vinden og vokser fort. Det var det raskest voksende språket på GitHub i 2018[9], så det virket som et språk som kan være godt å ha litt erfaring med. Vi valgte Spring Boot fordi det er nytt og spennende, én på gruppa hadde så vidt prøvd det ut tidligere og syntes det virket lovende, og det passet godt sammen med Kotlin. Hibernate ble valgt fordi vi ville bruke en ORM, og den er utbredt generelt, og også spesielt sammen med Spring Boot. For utvikling av Frontend gikk vi for React fordi det er et av de mest populære Javascript-rammeverkene, og derfor noe vi ville ha mer erfaring med. Vi ønsket å bruke typesikker JavaScript, og valgte TypeScript, for å gjøre koden lettere å forstå og bruke i etterkant. Som et av våre krav til spillet skulle det være client-side prediction. For å oppfylle dette kravet i størst mulig grad er det praktisk å ha globale variabler som kan aksesseres fra alle komponenter., og for dette brukte vi Redux. Det er et relativt utbredt rammeverk, som vi på forhånd hadde sett oss ut og kunne tenke oss å lære om i løpet av prosjektet.

4.4 Arbeids- og rollefordeling

Gjennom proskjektoppgaven har vi hatt en jevn og god arbeidsfordeling. Vi har for det meste sittet samlet og arbeidet samtidig. I starten var det et større skille mellom hvem som skulle sette seg inn i hvilken teknologi, og hvem som tok seg av dokumentering av oppstarten, som fullføring av visjonsdokument og user stories. Så snart vi kom skikkelig

i gang ble det en mer flytende rollefordeling som har fungert bra.

Resultater

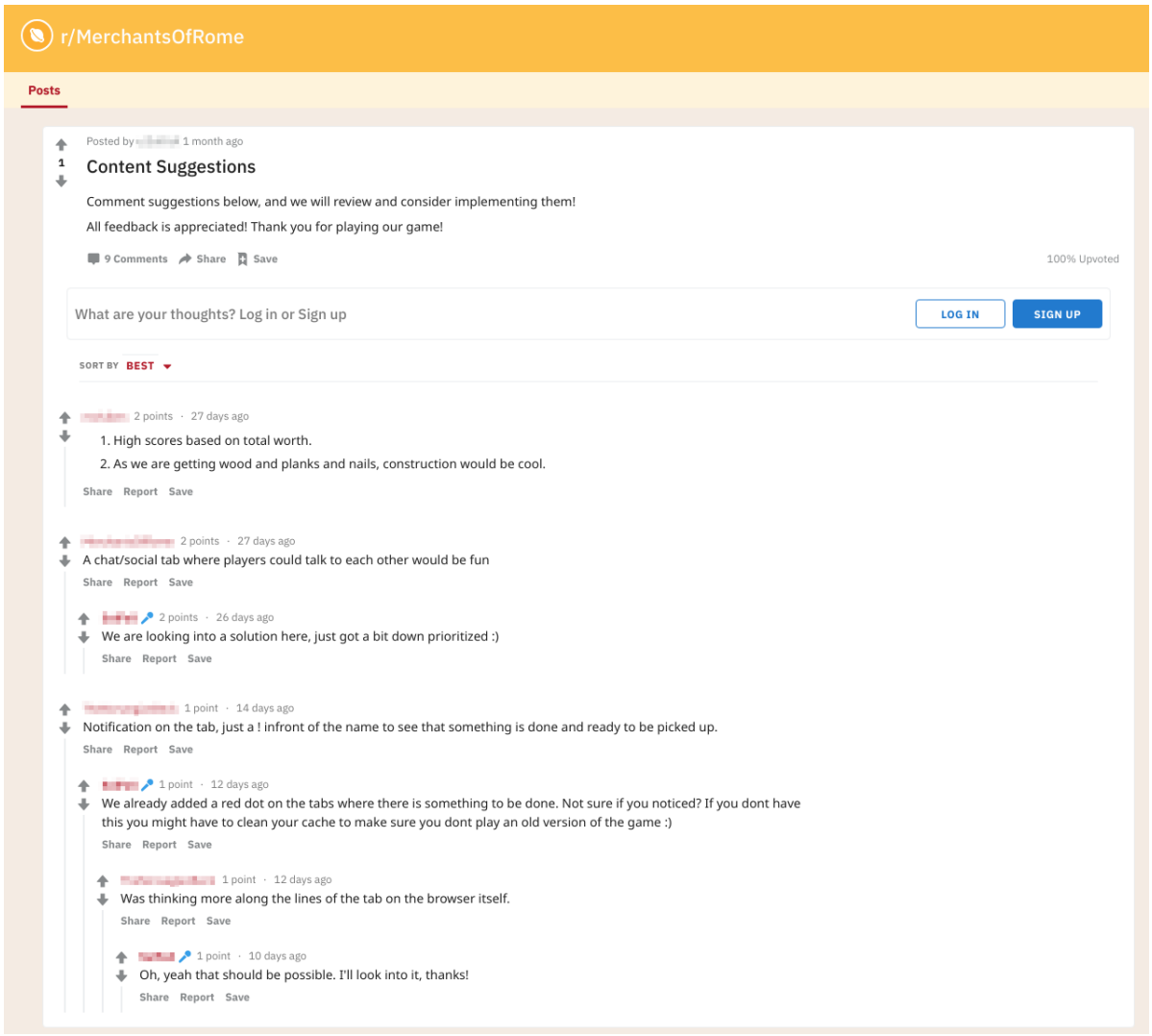
Dette kapittelet tar for seg resultatene som er oppnådd i løpet av bacheloroppgaven, både de vitenskapelige, de ingeniørfaglige og de administrative.

5.1 Vitenskapelige resultater

Vi har samlet inn mange tilbakemeldinger, til sammen 202 stykker, som vi har kategorisert etter hvor nyttige de var. Vi har brukt fem forskjellige metoder for å samle inn, med varierende suksess. En samlet oversikt over alle tilbakemeldinger finnes i eget vedlegg Tilbakemeldinger.

5.1.1 Offentlig

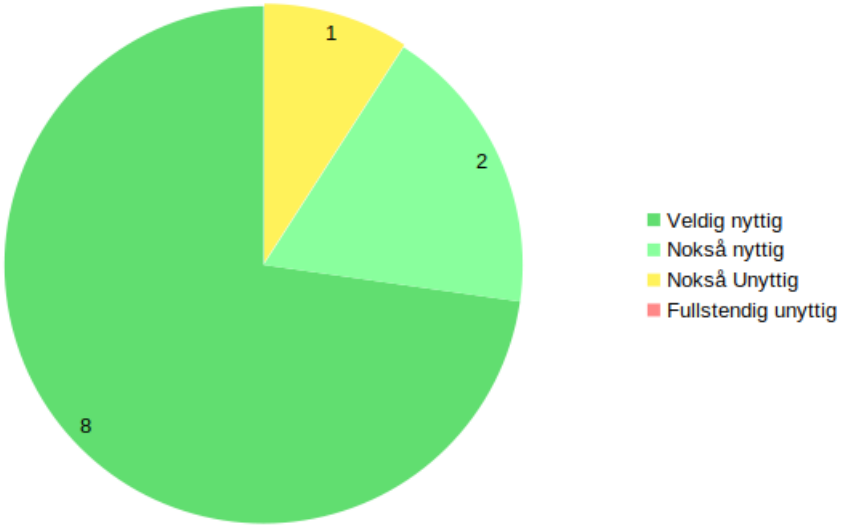
Vi la ganske tidlig inn to knapper som lenket til Reddit (se Figur 4.1), men vi fikk begrenset med tilbakemeldinger gjennom denne kanalen. Det kom noen tilbakemeldinger om nytt innhold og én tilbakemelding om feil i spillet.



Figur 5.1: Eksempel på tilbakemelding via Reddit

Litt senere opprettet vi en egen Discord for spillet, som vi også linket til fra inne i spillet, og vi fikk et par tilbakemeldinger her også, men ikke veldig mange sammenlignet med andre innsamlingsmetoder.

Vi fikk totalt inn 11 tilbakemeldinger i denne kategorien, og de var generelt veldig nyttige.

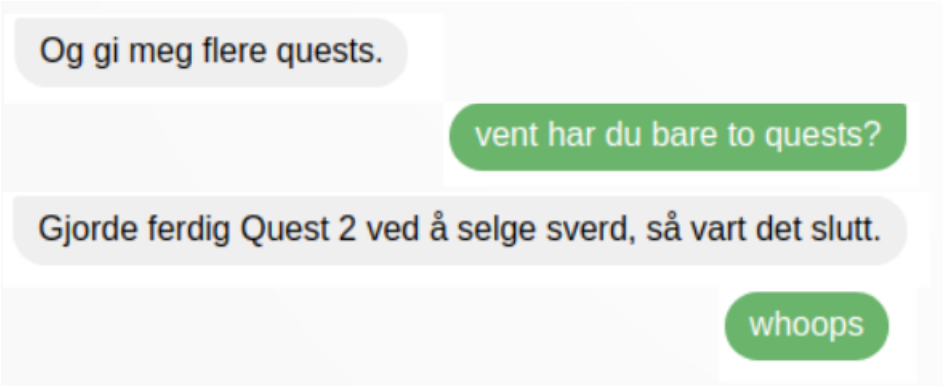


Figur 5.2: Graf som viser nyttigheten av offentlige tilbakemeldinger.

Vi fikk inn veldig få tilbakemeldinger, men til gjengjeld var et stort flertall av dem veldig nyttige. Kun én var nokså unyttig, mens åtte var veldig nyttige og to nokså nyttige.

5.1.2 Privat

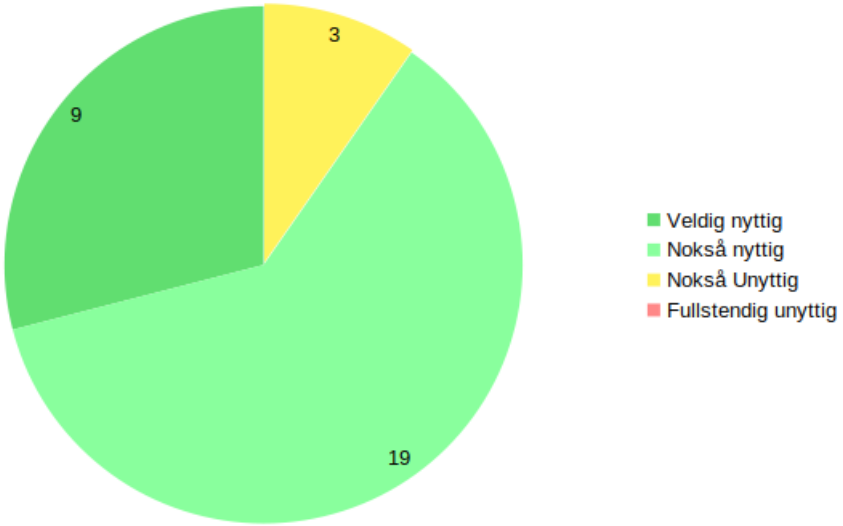
Gjennom de private kanalene har fått nest flest tilbakemeldinger. De tilbakemeldingene som er kommet muntlig eller via private chat'er har handlet om nytt innhold, feil i spillet, endringer eller bare etterspørsel etter mer innhold. Private meldinger går tydelig igjen som førstevalg hvis brukerne har noe de vil si uten å ha blitt spurt direkte av oss om tilbakemeldinger. Har brukeren tilgang til en privat chat brukes oftest denne fremfor de tilrettelagte kanalene.



Figur 5.3: Eksempel på privat tilbakemelding

I Figur 5.3 ser vi et eksempel der en bruker ikke hadde fått tilgang til Quest 3, selv om brukeren oppfylte kravene. Vi gikk kjapt inn i databasen og gav tilgang til Quest 3, for så å gå i kildekoden og fikset feilen som gjorde at spilleren ikke fikk tilgang automatisk.

Vi fikk totalt inn 31 tilbakemeldinger i denne kategorien, og de var generelt ganske nyttige.



Figur 5.4: Graf som viser nyttigheten av private tilbakemeldinger

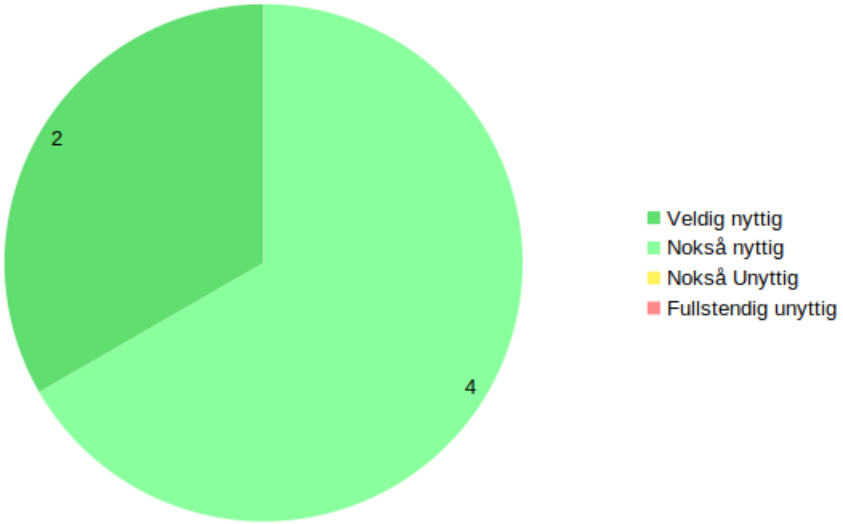
Vi fikk ikke inn noen fullstendig unyttige tilbakemeldinger, men et par nokså unyttige. Et stort flertall var bare nokså nyttige, men det var også ganske mange vi fikk mye nytte av.

5.1.3 Brukertester

Ved bruk av brukertester fikk vi mange gode tilbakemeldinger, både ved at brukerene i etterkant av testen delte sine inntrykk og forslag til utbedringer, men kanskje viktigere så fikk vi viktige tilbakemeldinger gjennom observasjoner. Ved å observere førstegangsbrukere så vi tydelig hvilke deler av spillet som er intuitive og hvilke som ikke er det. Denne typen tilbakemelding ser vi ytterst sjelden andre steder.

Første runde brukertesting

Fra den første runden har vi samlet inn seks konkrete tilbakemeldinger, men det er i tillegg til observasjonene vi har gjort og som er hovedformålet med brukertestene.

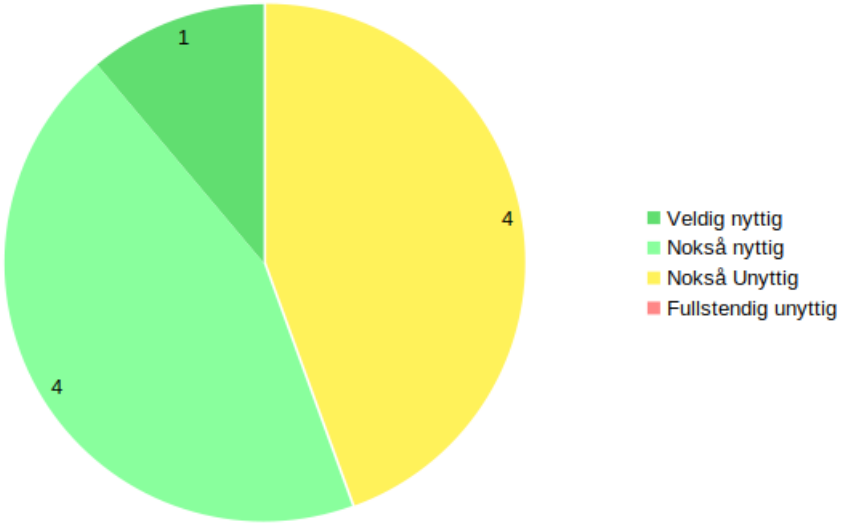


Figur 5.5: Graf som viser nyttigheten av tilbakemeldingene fra første runde med brukertesting

Her var alle tilbakemeldingene nyttige, men det er også veldig få tilbakemeldinger.

Andre runde brukertesting

Fra den andre runden har vi samlet inn ni konkrete tilbakemeldinger.

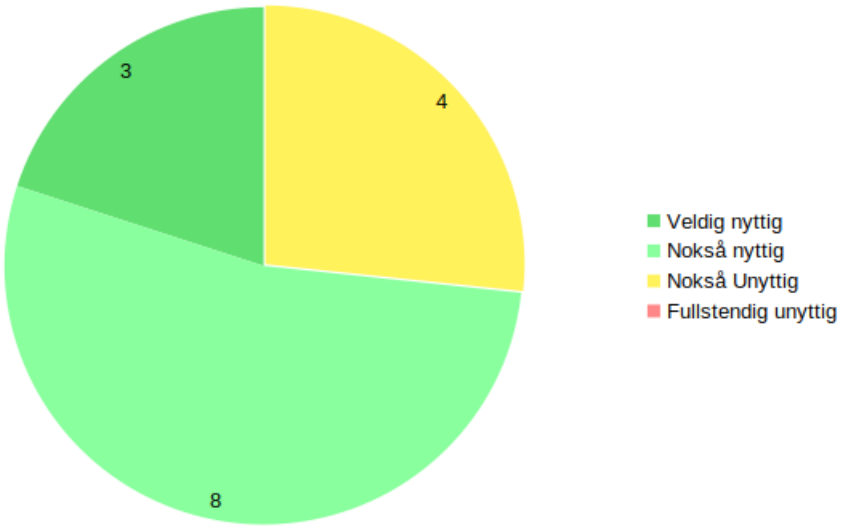


Figur 5.6: Graf som viser nyttigheten av tilbakemeldingene fra andre runde med brukertesting

Her var det generelt færre nyttige tilbakemeldinger enn i første runde. Dette må ses i sammenheng med at flere av de mest åpenbare tilbakemeldingene fra testrunde 1 var utbedret, samt gruppen fra testrunde 2 hadde betydelig mindre kjennskap til data og spill.

Sammenlagt data fra brukertesting

Fra brukertesting tilsammen har vi samlet inn 15 tilbakemeldinger



Figur 5.7: Graf som viser nyttigheten av tilbakemeldingene fra brukertesting

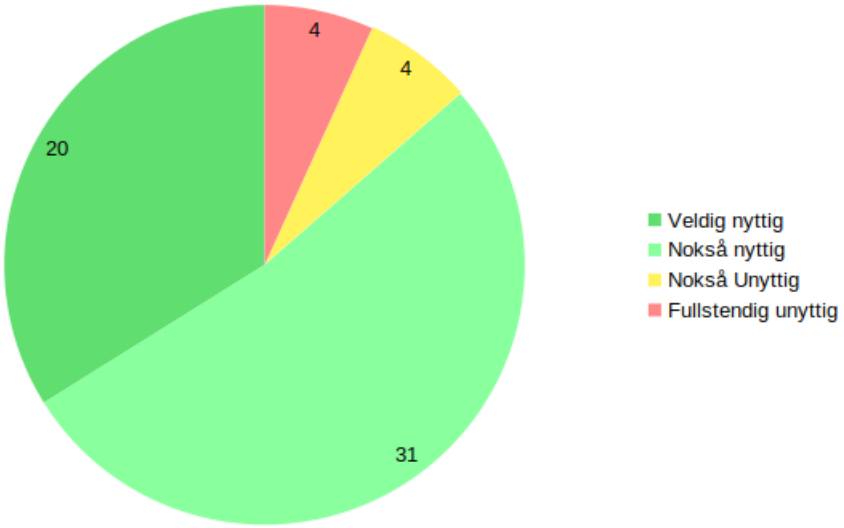
15 tilbakemeldinger er en del, og det kommer også i tillegg til alle observasjonene som gjøres under en brukertest. Det er hovedsaklig nokså nyttige tilbakemeldinger som er gitt.

5.1.4 Tilbakemeldingsverktøy

Denne typen innhentingsmetode var den som ga flest tilbakemeldinger. De aller fleste av tilbakemeldingene via verktøyet var relevante, enten i form av positiv omtale eller konkrete forslag til endringer. Dette kan ha en sammenheng med at de fleste brukerne som har kommet langt nok til å få opp skjema er venner og bekjente som vet at vi trenger tilbakemeldinger i denne avhandlingen, og derfor legger inn en større innsats enn de ville gjort i et spill de bare tilfeldigvis kom over.

Første spørsmålssett

I det første av de to spørsmålssettene i spillet fikk vi totalt inn 59 tilbakemeldinger, og de var generelt ganske nyttige.

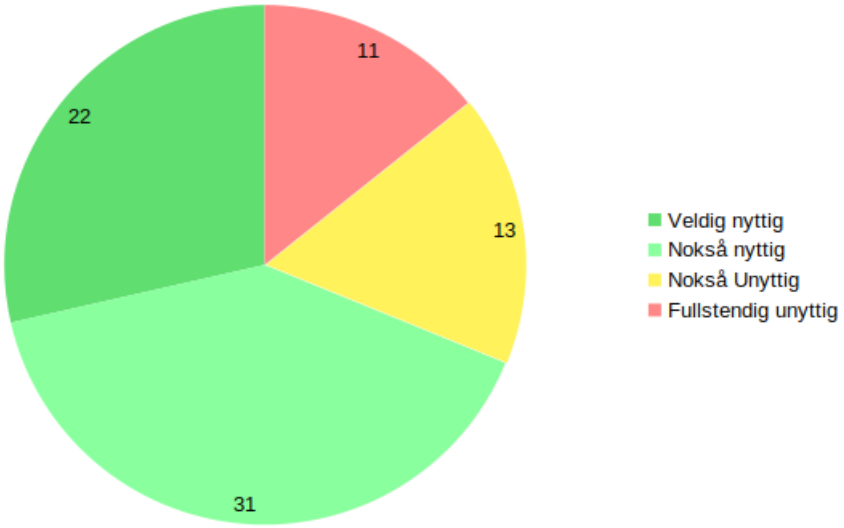


Figur 5.8: Graf som viser nyttigheten av tilbakemeldingene fra første spørsmålssett

Vi fikk inn mange flere med denne metoden enn de to foregående, og de var også generelt ganske nyttige. men vi fikk også et par helt unyttige tilbakemeldinger.

Andre spørsmålssett

Fra det andre spørsmålssettet i spillet fikk vi totalt inn 77 tilbakemeldinger, og de var generelt nokså nyttige.

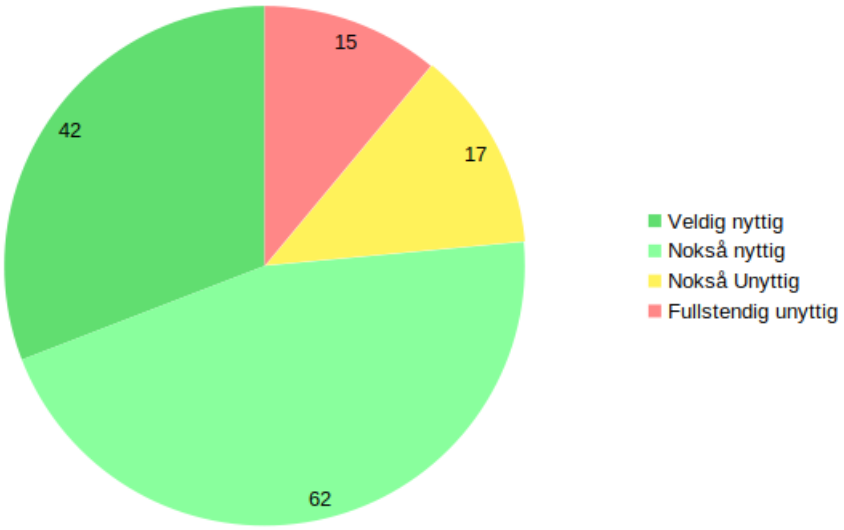


Figur 5.9: Graf som viser nyttigheten av tilbakemeldingene fra andre spørsmålssett

Dette var den metoden vi fikk aller flest tilbakemeldinger fra, men også den vi fikk desidert flest fullstendig og nokså unyttige tilbakemeldinger. Antallet veldig og nokså nyttige tilbakemeldinger er veldig likt som antallet veldig og nokså nyttige tilbakemeldinger fra det første spørsmålssettet.

Sammenlagt data fra tilbakemeldingsverktøyet

Samlet fikk vi totalt inn 136 tilbakemeldinger med tilbakemeldingsverktøyet i spillet, og de var generelt ganske nyttige.

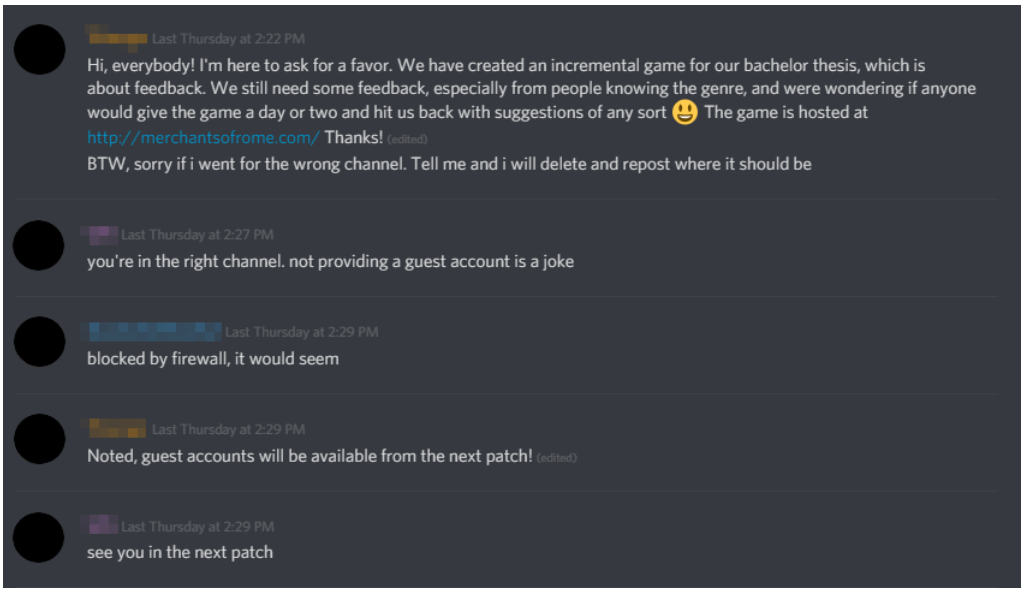


Figur 5.10: Graf som viser nyttigheten av tilbakemeldingene fra tilbakemeldingsverktøyet

Med over 100 veldig og nokså nyttige tilbakemeldinger var dette et veldig godt verktøy, men med nesten en fjerdedel fullstendig og nokså unyttige tilbakemeldinger, så krever det også en del jobb å finne de gode tilbakemeldingene.

5.1.5 Andre utviklere og spillere av sjangeren

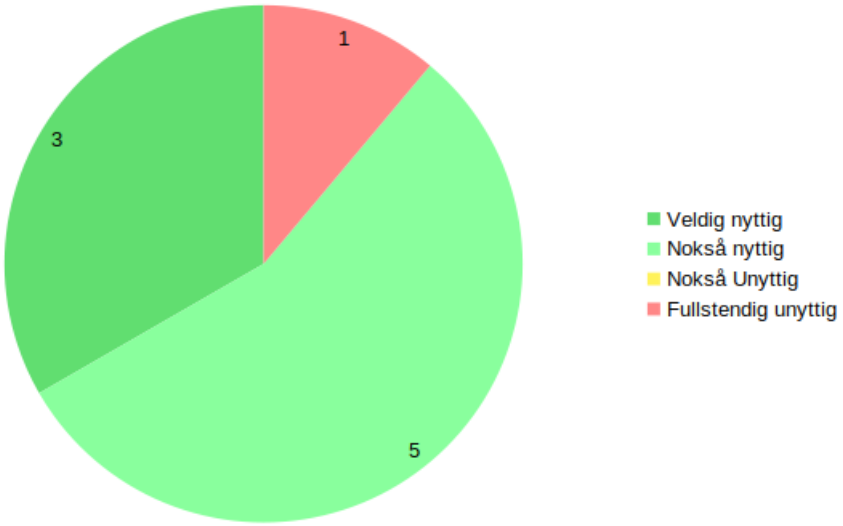
Mens flesteparten av tilbakemeldingene er gitt av venner og kjente har det kommet en del tilbakemeldinger fra andre utviklere og spillere vi har direkte nådd ut til i håp om tilbakemeldinger fra brukere kjente med spillsjangeren. Her er det lett å trekke et skille mellom kjente og ukjente brukere. Våre venner og bekjente virker i større grad åpne for å legge inn en innsats før de gir tilbakemeldinger på deres inntrykk. I et forsøk på å være snille virker det som de er litt ukritiske og unngår for mye negative tilbakemeldinger.



Figur 5.11: Første tilbakemelding fra utviklermiljøet

Rake motsetninger kommer når vi ber andre med erfaring med denne typen spill ta en titt og fortelle om deres inntrykk av spillet. Et godt eksempel ses i Figur 5.11. Umiddelbart etter vi inviterer utviklerene svarer en at et spill uten tilbud av gjestebrukere er en spøk og at han ikke har noen planer om å prøve spillet uten.

Vi fikk totalt inn ni tilbakemeldinger i denne kategorien.



Figur 5.12: Graf som viser nyttigheten av tilbakemeldinger fra utviklere og spillere av sjangeren

Tilbakemeldingene var generelt nyttige, med et unntak for en tilbakemelding som var helt unyttig fordi den kutta ut etter bare en halv setning.

5.2 Ingeniørfaglige resultater

Fra vedlegget Visjonsdokument kan vi se at det ble satt noen krav til produktet. Av disse er noen oppfylt, mens andre har vi ikke rukket å arbeide med, siden spillet ikke ble stort nok i løpet av bacheloroppgaven. Gjennom bruk av jenkins har systemet sikkerhet mot SQL-injections og cross-site-scripting, som dekker OWASP (2013) A1 & A3.

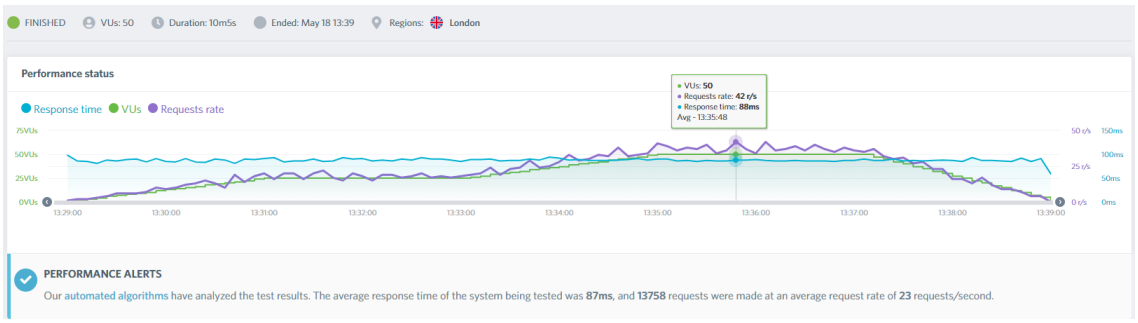
I løpet av bacheloroppgaven har serveren som spillet kjører på vært nede i noen få timer midt på natta, noe som var utenfor vår kontroll, men vi har ikke tatt ned spillet mer enn omtrent 30 sekunder om gangen for oppdateringer, som betyr at spillet ellers har vært oppe og kjøre i over 99,99% av tiden, 'the four nines'. Selv medregnet nedetiden til serveren så har spillet vært oppe i over 99.6% av tiden.

Når det gjelder kravet om balanse ble ikke spillet stort nok til at dette ble ferdig i løpet av bacheloroppgaven. Balanse handler om at det er brukerne som skal bestemme prisene på tingene som de vil selge til andre brukere, men i løpet av bacheloroppgaven ble det ikke laget noe marked til brukerne.

Konkurransedyktighet er et krav som er vanskelig å si om er oppfylt. Vi vet at flere brukere har spilt lignende spill tidligere, men likevel spiller vårt spill aktivt, som tyder på at det i hvert fall har noe å tilby markedet. Mer enn dette er vanskelig å si etter så kort tid. Et annet krav til produktet var client-side prediction. Dette er oppfylt der det er mulig i spillet. De stedene der det ikke er tilfeldige tall involvert, så predikerer frontend det som skjer backend og fortsetter i den tro at frontend er synkronisert med backend.

Brukervennligheten til starten av spillet er dokumentert med to brukertestrunder. Resultatet fra disse brukertestrundene viser at brukervennligheten, i hvert fall for starten av spillet, ikke er helt topp. De tilbakemeldingene som vi fikk fra brukertestene viser at vi har en vei å gå før brukervennligheten er der vi vil at den skal være.

Responstiden for spillet skal etter kravet i Visjonsdokumentet være under ett sekund ved 100 ulike brukere. Ved bruk av stresstest med 50 ulike brukere og på det meste 42 forespørsler per sekund ser vi en stabil responstid, på rundt 90 millisekunder. Lite tyder på at det vil være over ett sekund responstid ved 100 brukere.



Figur 5.13: Graf for stresstest

5.3 Administrative resultater

Ved å bruke prinsipper fra Lean Startup fikk vi en god arbeidsprosess gjennom utviklingen av MVP-en. Vi slapp MVP-en en del senere enn planlagt, dog var det nødvendig da det som dro ut tiden var utviklingen av en sikker innloggingsløsning. Med unntak av at oppstartsfasen ble dratt ut, fikk vi brukt de smidige vektøyene godt og så nytten av å bruke godt utarbeidede metodikken.

Da MVP-en ble sluppet var vi inne i en fin flyt og valgte å fortsette med samme prosess som vi allerede hadde, både fordi vi så på det som mest effektivt, og fordi vi ønsket å gå mer og mer bort fra utvikling og fokusere på innsamling av data, til skriving av avhandling. Ettersom MVP-en ble ferdig senere enn det som var planlagt ble også både Gantt-diagrammet og rapporten fra oppstartsmøtet[1] unøyaktig i forhold til faktisk prosess.

[1] Se vedlegg Prosjekthåndbok

Diskusjon

I dette kapittelet diskuterer vi hvorfor vi har gjort som vi gjorde i løpet av bacheloroppgaven. Det vil også bli argumentert for og i mot valgene våre med tanke på både det vitenskapelige, det ingeniørfaglige og det administrative. I tillegg tar vi opp gruppearbeidet i løpet av bacheloroppgaven.

6.1 Vitenskapelige resultater

6.1.1 Tilbakemeldinger

Tilbakemeldinger på eget initiativ

Ved å se på de tilbakemeldinger som kommer på brukernes eget initiativ er det tydlig at Reddit blir for høy terskel til at brukerene bruker forumet aktivt. Det er vanskelig å trekke non konklusjon på om det ville gitt flere tilbakemeldinger om vi hadde hatt et bedre lavterskeltilbud eller om tilfellet var at de som ønsket å gi tilbakemeldinger allerede hadde kanaler de heller ønsket å bruke.

Endringer

En av endringene vi la til i spillet for å prøve å få det mer intuitivt var at vi la til et rødt merke på tab-ene i spillet hvis det var noe man kunne gjøre i denne tab-en. Her var ønsket at disse fargede merkene skulle hjelpe brukerne til å forstå hva de burde gå inn på, og dermed få spillet til å bli mer intuitivt. En annen endring som vi gjorde var etter tilbakemeldinger om at det var litt vanskelig å vite hvor man skulle selge beholdningen sin. Det noen brukere gjorde var at de gikk innom en tab som egentlig ikke var nyttig for spillet. Denne tab-en fjernet vi. Det neste de gjorde var å gå innom den riktige tab-en, men det var vanskelig å vite at man skulle trykke på bildene øverst på siden, fordi hverken disse eller musepekeren ga noen indikasjon på at de var klikkbare. Dette endret vi ved at musepekeren endrer form når den 'hovrer' over bildene. Når vi hadde gjort disse endringene hadde vi noen flere brukertester, samt at vi fikk noen nye brukere som begynte å spille spillet.

Etter de nye brukertestene fikk vi få de samme tilbakemeldingene om at de ville ha det mer intuitivt i begynnelsen av spillet.

Siden dette ble gjort på forskjellige personer er det vanskelig å si om de nye personene hadde oppdaget det samme som de første personene gjorde, men det er noe som kanskje vil vise seg senere, hvis det er enda flere nye brukere som spiller spillet og ikke

gir noen tilbakemeldinger på akkurat dette problemet. Det vi merket var at vi fikk tilbakemeldinger om andre ting i spillet. Kanskje var dette fordi de endringene vi gjorde fikset de vanskelighetene og dermed la de nye spillerne merke til andre ting å endre på i stedet for. En veldig viktig ting å legge merke til er at tilbakemeldingene vi fikk fra observasjoner gikk igjen flere ganger uten at det noen gang kom frem fra noen av de andre formene for tilbakemelding.

Tilbakemeldingsverktøy

En av metodene som vi brukte for å få tilbakemeldinger på var å ha litt lokkemiddel i spillet, ved at brukerne fikk belønning for å svare på spørsmål. Svarene på spørsmålene var fritekstfelter, som gir brukerne anledning til å skrive det de vil, både svar som hjelper oss eller svar som ikke gir noen mening, bare for at de skal få belønningen. Det vi satset på var at de ville hjelpe oss og gi oss gode tilbakemeldinger, siden vi i teksten til spørsmålene skrev at dette handlet om en bacheloroppgave.

En av grunnene til at vi hadde en tilbakemeldingsmetode som ga brukerne belønning for tilbakemelding var for å se om dette påvirket villigheten til brukerne, samt at selv om brukerne fikk belønning uansett tilbakemelding, om de var villige til å gi gode, konstruktive tilbakemeldinger. Det som var en av ulempene med å bare gi brukerne belønning uansett tilbakemeldinger, var at vi kunne ende opp med å bare få tomme tilbakemeldinger.

Vi får her noen fordeler ved at vi ber om tilbakemelding på et spill og ikke en vanlig applikasjon, som gjør at vi kan tilby poeng i spillet i bytte mot tilbakemeldinger, som betyr at brukerne får et insentiv uten at det koster oss noe. Det er ikke mulig på alle prosjekter.

Generelt om innhentingsmetodene

De forskjellige innhentingsmetodene krever forskjellig mengde arbeid fra vår side. Som en hovedregel vil de innhentingsmetodene som krever mer arbeid fra oss, kreve mindre arbeid fra brukerne for å gi tilbakemeldinger.

Det negative med de innhentingsmetodene som krever mer arbeid fra brukerne er at det kan være med å begrense i hvor stor grad brukerne føler at det er verdt å gi tilbakemeldinger. Kanskje tenker ikke brukerne på at de kan gi tilbakemeldinger heller, hvis det er noe de ønsker eller lurer på. En av disse innhentingsmetodene er Reddit. Her må brukeren ha en bruker hos Reddit for å kunne gi tilbakemelding. Dette er noe som kan være en barriere for noen brukere, og da får vi ikke de tilbakemeldingene.

Antall tilbakemeldinger

Vi samlet totalt inn 202 tilbakemeldinger, og av disse kom 136 fra tilbakemeldingsverktøyet i spillet. Det er ingen tvil om at dette var det mest nyttige verktøyet. Av de 136 var 32 nokså eller fullstendig unyttige, som betyr at vi måtte bruke litt tid på å finne

frem til de gode tilbakemeldingene, men selv med bare de 100 nyttige tilbakemeldingene er dette det desidert beste verktøyet. Ulempen er selvfølgelig at dette var den innhentingsmetoden som tok mest tid å implementere.

6.1.2 Mindre observasjoner

En observasjon til som det ble lagt merke til var at når brukere begynte å spille spillet for første gang, så tok jentene seg bedre tid til å lese gjennom oppgavene som ble gitt, mens de fleste guttene bare klikket seg gjennom til det sa stopp og deretter prøvde å forstå hva de skulle gjøre. Dette kan være da de har mindre kjennskap til spill, eller bare fordi de er mer nøysomme i ting de gjør. Dette er for lite grunnlag til å trekke en konklusjon på, men det var bare en liten observasjon på det som vi rakk å gjøre.

6.2 Ingeniørfaglige resultater

6.2.1 Nye teknologier

Det at vi valgte å utvikle konsekvent i språk og med rammeverk vi ikke hadde særlig kjennskap til gjorde at MVP-en ble ferdig senere enn planlagt. En ting er at vi kom i gang en del senere ettersom vi trengte lenger tid på å bli konfortable med teknologivalgene, men det som gjerne trakk ned mest var det at vi måtte lære å lage en innloggingsløsning med sesjonshåndtering som kunne samhandle med teknologiene vi allerede brukte. Dette endte med å bli den store flaskehalsen i utviklingen av MVP-en, og veldig mye tid kunne vært spart på å velge teknologier vi har brukt i tidligere prosjekter og hentet en innloggingsløsning fra en av disse. Etter vi fikk ferdig MVP-en kunne vi dog nyte godene av det gode forarbeidet vi utførte med gode rammeverk. Nytt innhold kom hyppig, og vi fikk gode tilbakemeldinger på at brukerene satte pris på å hele tiden ha innhold å strebe mot. Oppveiing blir mellom å kunne holde følge med brukernes ønske om nytt innhold på bekostning av tid i produksjon med muligheter for tilbakemeldinger. Å ikke holde følge med brukernes ønske om hyppige oppdateringer kan også akkumulere nyttige tilbakemeldinger fra frustrerte brukere.

Likevel står vi ved beslutningen om nye teknologier da vi ser på oppgaven som en læringssituasjon og ønsker å komme ut av det som beste mulige utviklere, om så litt på bekostning av produktet.

6.2.2 Prosess

Fordeler

Fordelen med at vi valgte å kun bruke issue board, i stedet for Kanban board, var at vi kunne bruke mer tid på utvikling og ikke waste. På grunn av dette valgte vi å ikke lage et mer detaljert issue board.

Vi brukte også Preproduction sandbox. Dette gjorde til at vi fikk testet alt nytt innhold i spillet, før vi la det inn i master. Dette gjorde til at vi fikk kontroll på alle feilene i spillet før det ble sluppet til brukerne.

En annen fordel med prosessen som vi valgte var å bruke Continuous Development. Dette gjorde at vi fikk lagt ut nytt innhold ofte til brukerne og dermed kunne opprettholde interessen til brukerne mer.

Det at vi bestemte oss for å lage en MVP gjorde at vi fikk spillet tidligere i produksjon og dermed kunne begynne å samle inn tilbakemeldinger tidligere.

Ulemper

En ulempe med at vi valgte å utvikle et minste levedyktig produkt, og ikke minste testbare produkt, var at vi ikke kunne benytte oss av pivot tidligere enn det vi gjorde. Dette igjen førte til at vi fikk samlet inn mindre vitenskapelige data.

6.2.3 Brukervennlighet

Som målestokk på brukervennlighet har vi obervasjonene gjennom brukertester. Ved første testrunde hadde vi et inntrykk av at spillet var en hel det mer intuitivt enn det viste seg å være. Spillet var i produksjon en god stund før vi utførte første brukertest, og vi hadde til da ikke for noen tilbakemeldinger på at spillerene slet med å komme i gang. Først under første runde brukertester så vi hvordan testdeltakerene slet med trivielle oppgaver. De klarte utføre dem på sikt, men brukte langt mer tid en vi forventet. Etter utbedringer basert på observasjoner ved første testrunde gav testrunde to langt bedre resultater, og spillet virker å være nokså brukervennlig.

6.2.4 Responstid

Responstiden for spillet nå er omtrent 50 millisekunder for en forespørsel. Denne responstiden er med omtrent 20 aktive spillere. En av grunnene til dette er at det brukes mye client-side prediction, samt at spillet i seg selv ikke krever handlinger så ofte. En gjennomsnittspiller gjør ikke forespørsler mot serveren mer enn et par ganger i minuttet på det mest aktive, og man kan ofte gå flere timer uten å sende en forespørsel i det hele tatt selv om man er inne i spillet og følger med. En annen grunn til at responstiden er som

den er, er fordi det er lite sannsynlig at to forespørsler kommer nærme nok hverandre til at de vil påvirke hverandre.

Det er gjort en stresstest av tjeneren[1], som tester responstiden ved 50 ulike brukere og på det meste 42 forespørsler per sekund. Denne stresstesten viser også det som vi har sett på responstiden manuelt, at det er lite som tyder på at responstiden vil bli over ett sekund ved 100 ulike brukere.

6.3 Administrative resultater

Et av nederlagene vi hadde gjennom oppgaven var utsettelse av spillslipp. Gjennom studiet har vi aldri lagd en en sikker innloggingsløsning, noe som er forventet i et spill hvor spillere deler informasjon og samhandler. Å utvikle denne tok betydelig lenger tid enn planlagt som førte til atMVP-en ble ferdig senere enn det som var planlagt. Dette førte til at vårt delmål om å slippe spillet i slutten av februar ble utsatt til etter vår eksamensperiode [2] for å sikre hurtig respons på tilbakemeldinger. Dette førte i all hovedsak til at vårt ene delmål ble nådd sent. Dette førte til at vi igjen bortprioriterte utviklingsprossessen vi planla for siste del av prosjektet som gir et brudd på planen fra Gantt-diagrammet. Likevel ble delmålet nådd i tide til å oppnå vitenskaplige resultater og målet med oppgaven ble nådd.

6.3.1 Profesjonsetikk bak oppgaven

I dagens samfunn utvikles mer og mer teknologi. Noen systemer er åpenbart uetiske å utvikle mens andre regnes som tvers gjennom gode. Eksempler fra hver sin ende av skalaen er selvstyrte våpensystemer og systemer for kontroll av feilmedisinering ved sykehus. Hvor på skalaen produktet vårt kommer er kanskje ikke like svart på hvit. Spill er underholdningsprodukt og er opprinneling til for å skape glede hos spilleren. Spillet vårt ligger i en sjanger, Incremental game, som i stor grad er avhengighetsskapende. I en bransje som i stor grad handler om komersiell profitt er det lett å legge til kjøpbare fordeler. Mange av spillerene som velger å betale for fordeler opplever umiddelbar fremgang. Det gjør igjen at spilleren opplever en lykkerus som igjen er insentiv for å kjøpe nye fordeler igjen. Spilleren kan føle at spillet herav går for sakte uten å kjøpe fordeler og fortsetter med dette for å holde den samme fremgangen gående. Slip ender mange opp med å bruke mye mer penger enn planlagt, og det er veldig ofte spillere som ikke er i en økonomisk situasjon hvor dette er bærekraftig. Ofte er dette også barn som ikke har økonomisk forståelse og bruker foreldrenes penger. Per i dag har ikke ikke kjøpbart materiale, og planleghger heller ikke å legge det inn. Derfor føler vi at oppgaven et godt innenfor de profesjonsetiske

[1] Se Figur 5.13

[2] Se vedlegg Prosjekthåndbok for timelister

retningslinjene vi som dataingeniører skal følge, og lager et spill som kun er til glede for spillerene.

6.4 Gruppearbeidet

Helhetlig har gruppearbeidet fungert bra. Spesielt i oppstartsfasen, da rollefordelingen av tydelig, var gruppedynamikken veldig bra og vi utfylte hverandre godt. Vi samarbeidet og utviklet om hverandre og fikk derfor en god forståelse av hverandres ansvarsområder. Etterhver som vi begynte å bli ferdig med MVP–en ble det vanskeligere å samhandle på samme måte da vi fikk større sprik i oppgavene vi utførte og ikke lenger hadde kapasitet til å holde styr på alt andre gjorde. Likevel hadde vi god arbeidsmoral, fordelte oppgaver godt og samarbeidet godt om oppgaver og problemstillinger som ble omfattende for en enkeltperson.

Konklusjon og videre arbeid

I dette kapittelet konkluderer vi bacheloroppgaven med tanke på de resultatene som ble oppnådd, og forteller våre tanker om videre arbeid.

7.1 Tilbakemeldinger

I denne avhandlingen har vi prøvd å se på hvordan våre handlinger påvirker hvilke tilbakemeldinger brukere gir. Er tilbakemeldingene forskjellige hvis vi bruker forskjellige metoder for å innhente tilbakemeldingene? Vi har sett en del tendenser, deriblant at man får mange fler tilbakemeldinger når man spør om det direkte, enn om man bare oppretter en arena for å dele dem uten noe mer oppfordring. Vi har sett at brukertester kan gi innsikt om brukervennlighet som brukerne ellers nøler med å dele, kanskje av frykt for å se dumme ut. Vi har også sett at det er forskjell på kvaliteten avhengig av hvem man samler tilbakemeldinger fra, og at det kan være veldig nyttig å få kritikk fra personer som har erfaring med utvikling eller bruk av produkter som ligner på ditt.

Hvis vi ser på antallet tilbakemeldinger mellom de forskjellige innhentingsmetodene, ser vi at de metodene der vi er mer aktive i innhentingen genererer flere tilbakemeldinger enn i de tilfellene der brukerne selv må ta initiativet. I tillegg til dette er den innhentingsmetoden som vi har fått flest tilbakemeldinger via også den tilbakemeldingen som gir brukerne en liten belønning når de har svart på noen spørsmål, men dette gir også en betydelig andel mindre nyttige svar, fra brukere som bare er interessert i belønningen

Hvilke metoder som gir mest nyttige tilbakemeldingene og tilbakemeldinger fra flest antall brukere er viktig innen alle områder der man skal tilfredstille brukere. Hvis man klarer å finne ut hvilke metoder som passer best for sin målgruppe innenfor sitt felt, vil man nok ha større sjanse for å lykkes enn ellers.

7.2 Minste levedyktige produkt

Da vi slapp MVP-en av spillet for de nærmeste vennene var det nok innhold til å holde brukerne opptatt i omtrent én dag med nytt innhold, men fordi brukerne vet at mer innhold kommer snart, så vil nok de fleste kunne fortsette å spille lenger, om enn litt mindre aktivt, frem til det kommer nytt innhold. Fordi vi hadde gode rammeverk på plass så fikk vi i løpet av bare noen få dager sluppet mer innhold, og vi klarte å lage innhold fortere enn det ble konsumert frem til vi ble nødt til å prioritere ned utvikling til fordel for rapportskriving.

Vi mener at vi slapp MVP-en på en ganske riktig tid utviklingsmessig, men skulle

ønske at det hadde skjedd tidligere i kalenderåret. Det hadde nok vært mulig om vi hadde valgt i hvert fall litt teknologi vi var mer kjent med, men da på bekostning av læringsutbytte.

7.3 Videre arbeid

Ved videreføring av dette studiet anbefaler vi i første omgang å samle en del mer data. For å kunne se tydligere tendenser vil det være viktig med et stort datasett med en større variasjon av brukere. Det kan være utterst nyttig å prøve å samle data fra brukere som ikke har tilknytning til utviklingsteamet da dette kan påvirke tilbakemeldingene og trekke konklusjon i feil retning. Ved gjenskaping av et lignende studie vil vi anbefale å ikke kombinere utforsking av ny teknologi med Lean og MVP. Vi valgte å gjøre det for læringsverdien, og det er ingen tvil om at det har vært lærerikt, men det har også generert mye waste, og var en av årsakene til at vi ikke klarte å følge tidsplanen vi la på forhånd. Det å skulle basere en tidsbegrenset oppgave, som en bacheloroppgave, på tilbakemeldinger fra brukere er en risiko, og vi vil nok anbefale en større margin enn den vi hadde da vi startet.

Referanser

[1] S. E. DeFranzo. (2015). 5 Reasons Why Feedback is Important, side: https://www.snapsurveys.com/blog/5-reasons-feedback-important/. Lastet ned: 18.05.2019.

[2] L. Daly. (2016). How feedback helps you develop better software: insights from the Jira Mobile team, side: https://www.atlassian.com/blog/software-teams/how-to-collect-feedback-for-software-development. Lastet ned: 18.05.2019.

[3] Ø. Lillerødvann. (2015). Fem råd for gjennomføring av brukertest, side: https://blogg.kantega.no/content/1351/Fem-rad-for-gjennomforing-av-brukertest. Lastet ned: 13.05.2019.

[4] T. P. Mary Poppendiek, *Lean Software Development: An Agile Toolkit*. Addison-Wesley Professional, 2003.

[5] (). The Seven Wastes — 7 Mudas, side: http://leanmanufacturingtools.org/77/the-seven-wastes-7-mudas/. Lastet ned: 08.05.2019.

[6] (). 7 GUIDING PRINCIPLES OF LEAN DEVELOPMENT, side: https://leankit.com/learn/lean/principles-of-lean-development/. Lastet ned: 08.05.2019.

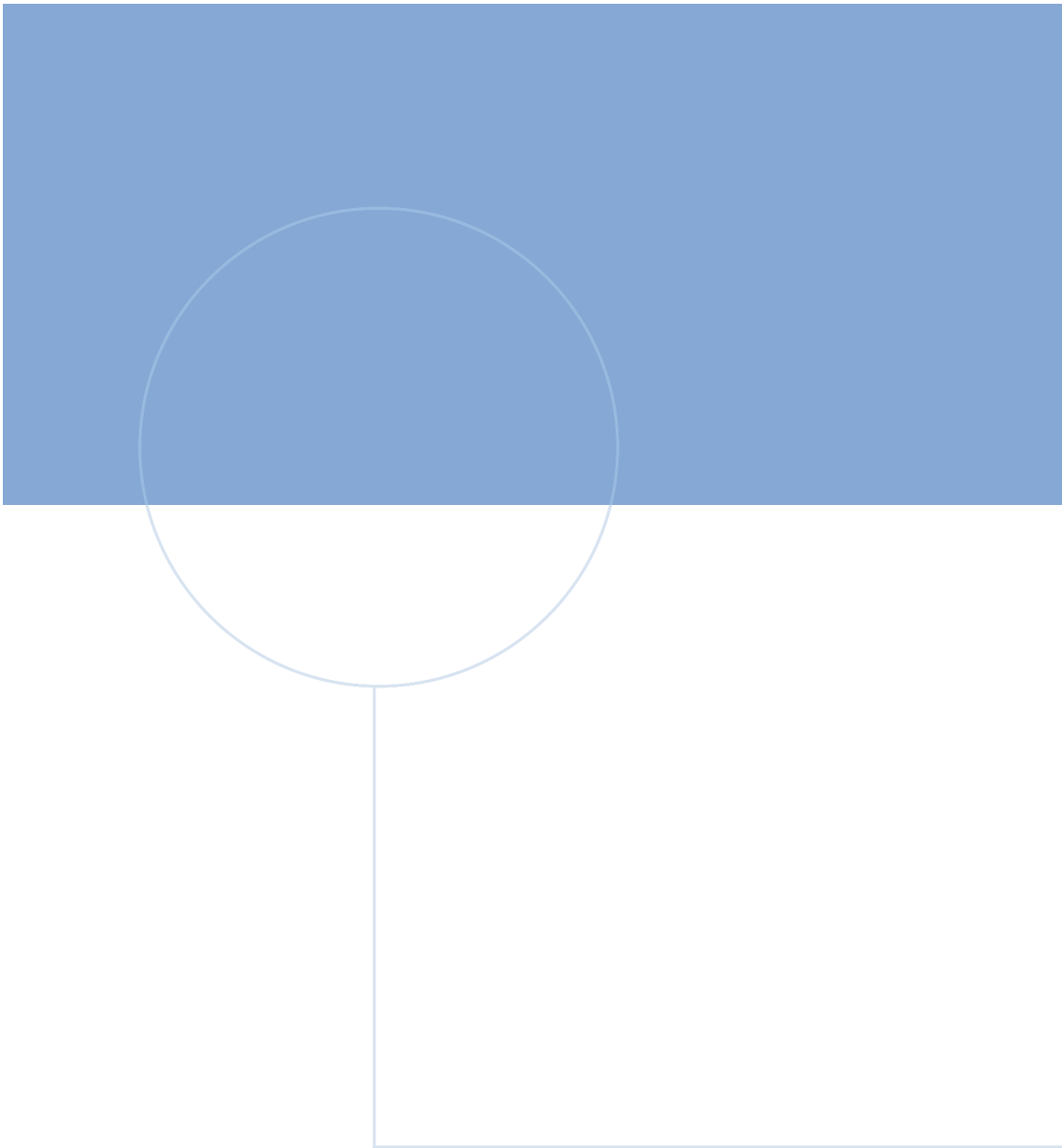
[7] C. P. Todd Sedano Paul Ralph, «Software Development Waste», tekn. rapp., 2017. side: https://www.researchgate.net/publication/313360479_Software_Development_Waste, Lastet ned: 09.05.2019.

[8] E. Ries, *The Lean Startup*. Crown Publishing Group, 2011.

[9] GitHub. (2019). Projects — The State of the Octoverse, side: https://octoverse.github.com/projects#languages. Lastet ned: 18.05.2019.

Vedlegg

- Brukertester
- Kravdokumentasjon
- Prosjekthåndbok
- Systemdokumentasjon
- Visjonsdokument



NTNU

Kunnskap for en bedre verden