

---

# Prosjektrapport

Bacheloroppgave i IT og Informasjonssystemer

## Playfu

---



Prosjekt utført av:

Govert Dahl – 233565

Sigve Elisassen – 233511

Sivert Heisholt – 233518

Ørjan Dybevik – 233530

20. mai 2022

# 1 Forord

Denne rapporten er skrevet som en del av faget BOP3000 ved Universitetet i Sørøst-Norge. Vi er alle studenter ved IT og Informasjonssystemer på USN Campus Bø. Gjennom studiet har vi vært igjennom ulike områder innen IT og Informasjonssystemer. Vi har drevet med frontend, backend og database, samt organisatoriske emner og emner som knytter flere områder sammen. Bacheloroppgaven er vår egen ide, og utviklet etter vår egen evne basert på det vi har lært gjennom studiet.

Grunnen til at vi valgte å skape et produkt ut ifra vår egen ide er at vi ønsket å bruke den kompetansen vi hadde i en praktisk øvelse. Vi satte fokus på teknologi og verktøy vi kommer til å bruke i arbeidslivet. Når vi er ferdige ønsker vi å ha god kjennskap til arbeidsmetodikk, samarbeid og ferdighetene som inngår i et slikt prosjektarbeid vi har valgt.

For å få oversikt over dokumenter vi har skrevet gjennom emnet kan du gå til [denne](#) lenken.

Vi ønsker i tillegg å takke våre studieveiledere, lærere og Gunn Marit. Gunn Marit Aspheim Hagen har igjennom hele skoleløpet vårt vært en god hjelp når det kommer til rettskrivning og grammatikk.

# Innhold

1	Forord	2
2	Innledning	6
2.1	Bakgrunn	6
2.2	Nødvendig forkunnskap	6
2.3	Ressurser	7
2.4	Sammendrag	7
3	Administrativ beskrivelse	8
3.1	Kort beskrivelse av oppgaven	8
3.1.1	Ideen	8
3.1.2	Kort drøfting om potensiale i markedet	8
3.2	Bruk av litteratur og internett	9
3.3	Roller- og ansvarsfordeling, samarbeid	10
3.3.1	Roller	10
3.3.2	Ansvarsfordeling	11
3.3.3	Ledelse	11
3.3.4	Samarbeid	11
3.4	Fremdriftsplan med ulike faser	12
3.4.1	Kort om fremdriftsplanens faser	12
3.4.2	Gantt Chart	13
3.4.3	Vurdering av planleggingen og gjennomføringen	15
4	Faglig beskrivelse	15
4.1	Brukerhistoriene	15
4.1.1	Kort om arbeidet	15
4.1.2	Tabell med brukerhistoriene	16

4.1.3	Use case diagram	19
4.2	Endringer	20
4.2.1	Endringer underveis	20
4.2.2	Ufullførte brukerhistorier	20
4.3	Utfordringer	21
4.3.1	Fravær	21
4.3.2	Steam	22
4.3.3	Steam sine API-er	22
4.3.4	CORS	23
4.3.5	Hosting	24
4.3.6	Implementasjon og feil	24
4.3.7	Server- og kode-effektivitet	24
4.4	GDPR	24
4.5	Utviklingsmetode	25
4.5.1	Planleggingsfasen	25
4.5.2	Kanban	25
4.6	Scrum	26
4.6.2	Utviklingsverktøy	28
4.6.3	Utviklingsbibliotek	31
4.7	Brukergrensesnitt	32
4.7.1	Logoen	32
4.7.2	Arbeidet med prototype-designet	33
4.7.3	Arbeidet med utvikling av Design	33
4.7.4	Eksempler på prototype og endelig brukergrensesnitt	35
5	Systemdokumentasjon	41
5.1	Overordnet beskrivelse	41

5.2	Frontend	43
5.3	Backend	43
5.3.1	API	43
5.3.2	Websocket	45
5.4	Database	46
5.5	Kodestruktur	49
5.6	Sikkerhet	50
5.6.1	Sikkerhetstester	50
5.7	Utplassering	53
5.7.1	Backend	53
5.7.2	API Dokumentasjon	54
6	Brukertesting	55
6.1	Om testplanen	55
6.1.1	Fremgangsmåte	55
6.2	Brukertesting fase 1.	56
6.2.1	Brukertesting instruks	56
6.2.2	Spørsmål i spørreundersøkelsen	57
6.2.3	Testresultat fase 1.	60
6.2.4	Resultater brukertesting fase 1.	64
6.3	Brukertesting fase 2	65
6.3.1	Testresultater fase 2	65
6.3.2	Resultater fra brukertesting fase 2	70
7	Konklusjon	71
8	Referanser	72
9	Oversikt over tabeller og figurer	74
9.1	Tabeller	74

## 2 Innledning

### 2.1 Bakgrunn

I forkant av eksamenstiden i 5. semester ble bachelorgruppen opprettet. Dette var basert på positive erfaringer med samarbeid i tidligere prosjekter og oppgaver. Vi avgjorde tidlig å gjennomføre et prosjekt hvor det handler om å lage en webapplikasjon, lignende prosjektet i APP2000. Hva dette ville bestå av avgjorde vi gjennom brainstorming og diskusjon i forkant av prosjektstart. Med flere spill-interesserte på gruppen falt dette valget naturlig på «Playfu». Et møtested for gamere som alle på gruppen forstod nytten for og var interesserte i å utvikle.

### 2.2 Nødvendig forkunnskap

For at rapporten skal være så forståelig som mulig så har vi laget en beskrivelse av eksterne programmer som er tatt i bruk slik at det er klart for leseren hva disse går ut på.

Discord er en plattform som er kjernen for mange spill-interesserte sin kommunikasjon. Her har de ulike grupper og venner de kan kommunisere med. Discord har et gratis program som kan lastes ned, men en kan også bruke programmet gjennom Discord sin nettside. Discord kan sammenlignes med Skype og Teams bare at det er skreddersydd med tanke på kommunikasjon for spill-interesserte.

Steam er en plattform som nesten alle PC-spillere bruker daglig. Steam fungerer som et bibliotek, butikk, forum og kommunikasjonsplattform. Her kan du kjøpe digitale kopier av flere tusen spill, og få tilgang til dem gjennom ditt eget bibliotek. Det er gratis å bruke, og du kan få tilgang til en rekke spill uten å betale. Steam er den største og mest brukte plattformen av denne typen.

## 2.3 Ressurser

Nedenfor er dokumenter som viser arbeidet gjort i prosjektet. På prosjektweb-siden kan en finne tekstdokumenter som prosjektbeskrivelsen, prosjektskisse, teamkontrakt og presentasjoner. Disse dokumentene er også vedlagt ved rapporten.

<b>Prosjektweb:</b> <a href="https://happy-river-0edb05f03.1.azurestaticapps.net/">https://happy-river-0edb05f03.1.azurestaticapps.net/</a>
<b>Kildekode prosjektweb:</b> <a href="https://github.com/dyb0/BOP3000ProsjektWeb">https://github.com/dyb0/BOP3000ProsjektWeb</a>
<b>Hovedapplikasjon:</b> <a href="https://bop3000.azurewebsites.net/">https://bop3000.azurewebsites.net/</a>
<b>Kildekode:</b> <a href="https://github.com/sivertheisholt/BOP3000">https://github.com/sivertheisholt/BOP3000</a>
<b>API-Dokumentasjon:</b> <a href="https://sivertheisholt.github.io/BOP3000/">https://sivertheisholt.github.io/BOP3000/</a>

Tabell 1: Oversikt over kildekode og viktig dokumentasjon

## 2.4 Sammendrag

I de neste delene av rapporten skal vi gjøre rede for arbeidet som er gjort i prosjektet. Til å starte med har vi del 3 «Administrativ beskrivelse» hvor vi beskriver det organisatoriske arbeidet i prosjektet. Deriblant har vi planlegging, rollene og ressursbruk. Del 4 «Faglig beskrivelse» handler om hvordan vi har arbeidet med utviklingen og hvordan det gikk. Del 5 «Systemdokumentasjon» består av en gjennomgang med beskrivelse av de ulike delene systemet består av. Der gjøres det rede for hvordan systemet er satt sammen med modeller som illustrerer dette og tester gjennomført for systemets sikkerhet. Del 6 «Brukertesting» beskriver planlegging, gjennomføring og resultatene til brukertestene som er gjennomført. Til slutt i del 7 «Konklusjon» reflekterer vi kort over de erfaringene vi har gjort og resultatene vi har hatt gjennom prosjektet.

## 3 Administrativ beskrivelse

### 3.1 Kort beskrivelse av oppgaven

Vår oppgave er å utvikle en webapplikasjon som gjør det mulig for folk med interesse for spill på nett å møte og kommunisere med andre. Dette vil vi oppnå gjennom registrering av brukere, valg av spill, opprettelse av grupper, gruppechatter og integrasjon med Discord og Steam.

#### 3.1.1 Ideen

Til å begynne med var vi sikre på at vi ville lage en webapplikasjon, men var usikre på hva den skulle handle om. Derfor valgte vi å gjennomføre en enkel brainstorming tidlig i januar. Sist semester hadde vi hatt emnet «Innovasjon i Team» hvor vi fikk god kjennskap til brainstorming. For å korte ned prosessen avtalte vi å forberede ideer for prosjektet til det neste møtet. Under dette møtet gikk vi gjennom hver person sine ideer og alle kommenterte på hvilke ideer vi likte best. Til slutt var det klart at Sigve sin idé var mest populær. Ideen var kalt «Tinder for gamere». Når det var klart for alle hva ideen gikk ut på var alle motiverte til å gjennomføre dette som prosjekt.

#### 3.1.2 Kort drøfting om potensiale i markedet

Vår påstand innad i gruppen da vi valgte dette prosjektet var at denne typen sosialt nettverk ikke eksisterer, og at Discord og Steam ikke dekker dette behovet. Mens den andre antakelsen stemmer, er det feil å anta at denne typen tjeneste ikke allerede eksisterer. På markedet finnes det allerede tjenester som WeGamers, Unblnd, Gamerlink, GameTree og Plink. Alle disse tjenestene tilbyr matchmaking for gamere på sine egne måter. Dermed kan vi trygt si at det er store sjanser for at behovet allerede er dekket i markedet. I tillegg er de nevnte bedriftene små, med noen få investorer og ukjente inntekter. Dette til tross at de har vært aktive i flere år.

I slutten av prosjektet gjennomførte vi brukertester. Dette var på en liten gruppe personer som alle er brukere av Discord. Under testen spurte vi om de kunne tenke seg å bruke en nettside som Playfu. Resultatene var varierte og gav etter vår mening ikke god nok antydning til at det er et stort behov for dette. Dermed er vi fornøyde med at dette forblir et prosjekt for øvelse og bevis på hva vi kan.



Dersom en tross motargumenter skulle fortsatt å operere Playfu ville det vært utfordrende, men ikke umulig. En bør i så fall gjøre det på fritiden siden det er dårlige utsikter finansielt på kort og mellomlang sikt. En mulighet kan være å sette søkelys på bestemte områder som for eksempel Norge eller enkelte spill arrangementer. Pluss å sørge for at kvaliteten er på linje eller bedre enn konkurrentenes. Her er det også viktig at produktet hadde levert noe unikt som markedet ville funnet interessant. (Unblnd, 2020)

## 3.2 Bruk av litteratur og internett

Mye av tiden i starten av prosjektet ble brukt på kursing og forberedelse av arbeidet vi skulle gjøre. Disse ressursene er listet i tabellen under med typene ressurser og forklaring på hvordan vi brukte de.

Type ressurs	Beskrivelse
Kurs på Udemy	Flere i gruppen gjennomførte kurs innen Angular og .NET.
YouTube	Under arbeidet med Figma for å lage prototype-designet brukte Govert flere kurs på YouTube for å kurse seg og finne løsninger.
Google	For å finne frem til dokumentasjon og ressurser som vi har tatt i bruk, så har vi benyttet Google.
Dokumentasjon	Vi brukte mange forskjellige verktøy og programvarer for å utvikle applikasjonen. Dokumentasjon er det viktigste verktøyet innenfor utvikling, og vi har blant annet brukt dokumentasjonen til: .NET, Angular, Github, Azure, DigitalOcean, Discord, Steam og Meilisearch med mer.

Tabell 2: Oversikt over ressurser

## 3.3 Roller- og ansvarsfordeling, samarbeid

### 3.3.1 Roller

I dette prosjektet lagde vi noen roller som fordeler ansvarsområder til hvert gruppemedlem. Disse rollene spesifiserer hvilket hovedansvar hvert medlem har, og hva de skal gjøre i prosjektet. Nedenfor er rollene listet med en kort beskrivelse for hver.

#### **Fullstack**

Har ansvaret for utviklingsarbeid innen både backend og frontend.

#### **Design**

Har hovedansvaret for designet og brukergrensesnittet til applikasjonen.

#### **Backend**

Har hovedansvaret for utviklingen på backend.

#### **Frontend**

Har hovedansvaret for utviklingen på frontend.

#### **API**

Har hovedansvaret for programmeringsgrensesnittet, også kalt API (Application Programming Interface).

#### **Hosting**

Har hovedansvaret for hosting av applikasjonen.

#### **Rapport**

Har hovedansvaret for rapporten i emnet.

#### **Testing**

Har hovedansvaret for testing av applikasjonen.

#### **Diagrammer**

Har hovedansvaret for diagrammer som skal lages for applikasjonen.

#### **Databaseadministrator**

Har hovedansvaret for databasen som skal brukes til applikasjonen.

## Lover og regler

Har hovedansvaret for relevante lover og regler for prosjektet.

### 3.3.2 Ansvarsfordeling

Rollefordeling og ansvarsforhold:

**Ørjan Dybevik:** Fullstack, Design

**Sivert B. Heisholt:** Backend, API, Hosting

**Govert Dahl:** Rapport, Testing, Lover og Regler

**Sigve A. E. Eliassen:** Rapport, Testing, Diagrammer, Databaseadministrator og Frontend

### 3.3.3 Ledelse

Vi valgte å opprette vårt helt eget prosjekt på våre vilkår. Derfor arbeider vi uten noen bedrift som oppdragsgiver, dermed er oppdragsgiver oss. Rollen oppdragsgiver kan ses på som et felles ansvar i gruppen.

Rollen Scrum master har vi rullert på. Vi var tre på gruppen som ønsket å være Scrum master siden de ikke hadde vært Scrum master tidligere. Dermed endte vi med at vi tre var Scrum mastere under hver vår tredjedel av prosjektet og Sivert som har mest kjennskap til rollen ville fungere som vara Scrum master gjennom hele prosjektet.

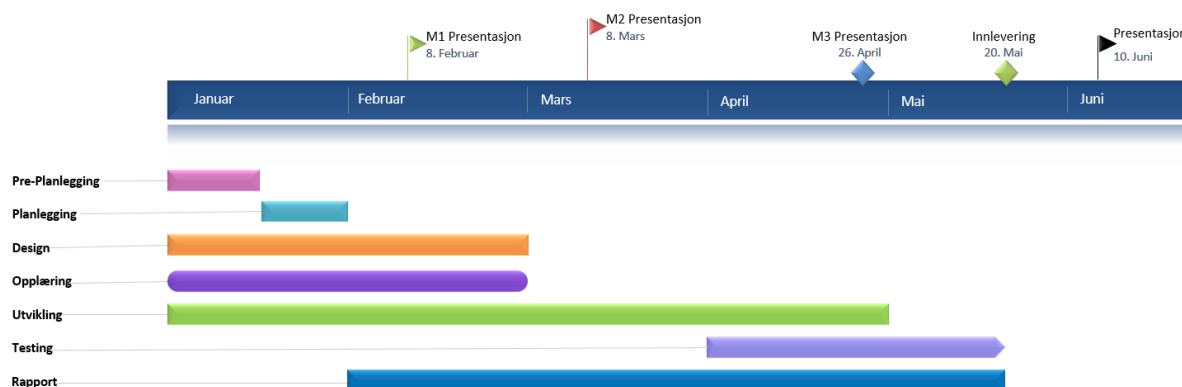
### 3.3.4 Samarbeid

Vi har fra starten av vært veldig klare på at godt samarbeid vil resultere i bedre resultat for prosjektet. Det har vært viktig at vi holder en jevn og god kommunikasjon igjennom hele prosjektet. I teamkontrakten (Se vedlegg «Teamkontrakt») har vi også spesifisert hva som forventes når det kommer til samarbeidet og hvilke handlinger det forventes å gjøre dersom noe er uklart.

Vi møttes to ganger ukentlig i takt med sprintene. En sprint varer fra hver tirsdag til og med mandag neste uke. På tirsdager hadde vi «Scrum Review» og på fredager hadde vi «Daily Scrum». Alle møtene våre er på nett, dette er fordi vi har mye av arbeidet vårt på stasjonære PCer som vi viser frem på møter. Det blir mer jobb dersom vi må sette opp bærbar pc-er for hvert møte.

Vi jobber alle sammen i Visual Studio Code, noe som gjør det lettere å hjelpe hverandre dersom noe skulle komme opp i den kategorien. Ørjan, Sigve og Sivert bruker Linux som operativsystem når de jobber på prosjektet. Dette er fordi vi synes Linux er lettere å sette opp og jobbe med når det kommer til utvikling.

### 3.4 Fremdriftsplan med ulike faser



Figur 1: Fremdriftsplan for prosjektets faser

#### 3.4.1 Kort om fremdriftsplanens faser

**Pre-planlegging:** Dette er den planleggingen og det arbeidet som ble gjort i forkant av prosjektets initielle oppstart. Her startet vi oppsett av siden før vi visste hva den ville handle om. Gruppen hadde i denne fasen satt i gang planen for hvordan de ville jobbe og hva de ville jobbe med uten at det ble veldig konkretisert.

**Planleggingsdelen:** Dette er planleggingen som ble gjort mellom prosjektoppstart og innlevering av prosjektbeskrivelsen. Gjennom kravene til prosjektbeskrivelsen ble prosjektet planlagt med valg av tema, problemstilling, fremdriftsplan og brukerhistorier.

**Design:** Dette er det arbeidet som ble gjennomført i Figma. Her ble det lagd flere enkle eksempler på løsning av oppgaven med fokus på design og brukervennlighet. Resultatet av arbeidet ble skisserte sider som vil brukes for inspirasjon til løsningene innen frondend-utviklingen. I denne fasen ble logoen lagd.

**Opplæring av verktøy:** Dette er den perioden der vi fokuserte mest på å lære og forberede oss for utviklingsfasen, med å blant annet ta kurs innen programmeringsspråk og lese oss opp på arbeidsmetodikk.

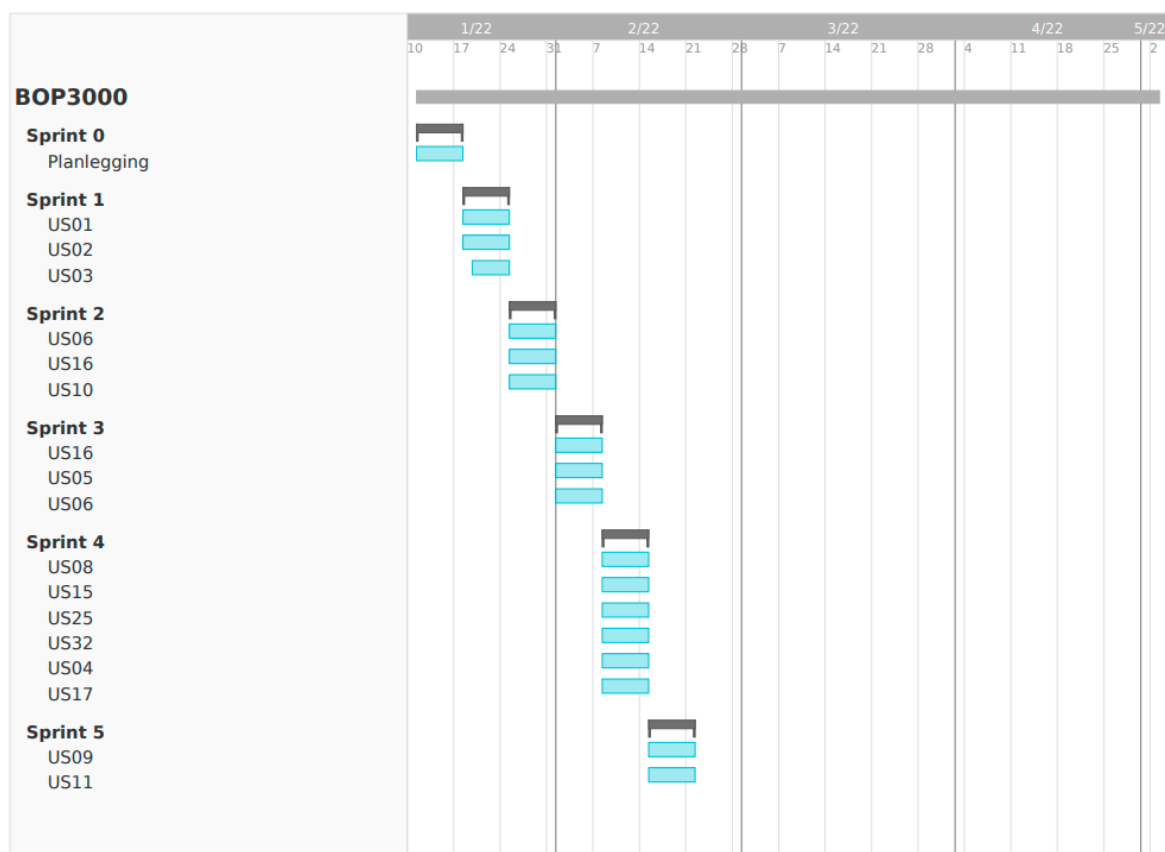
**Utvikling:** Her ble løsningen utviklet.

**Testing:** Her ble testingen av nettsiden gjennomført. Den bestod av brukertesting, kodetesting og sikkerhets-tester.

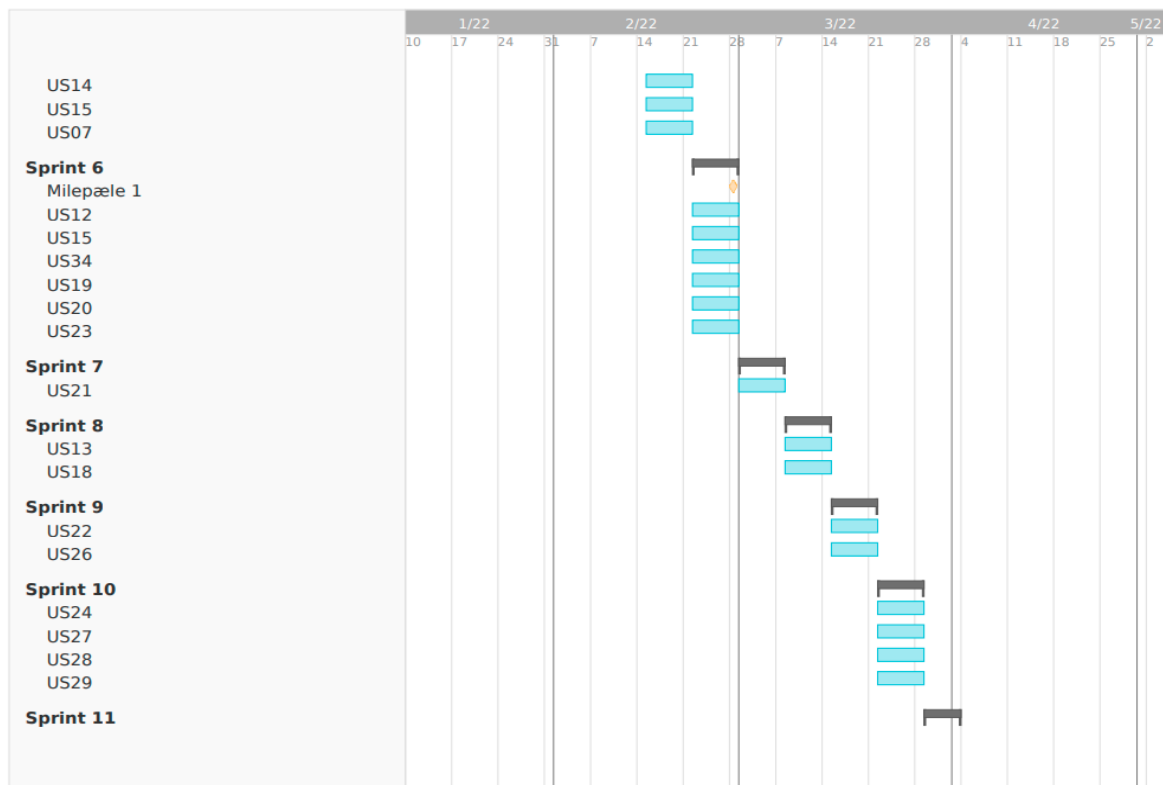
**Rapport:** Her er fokuset mest på å fullføre og levere prosjektets dokumenter.

### 3.4.2 Gantt Chart

Bildene her er planlagt fremgang i henhold til Prosjektbeskrivelsen som ble lagd i starten av prosjektet.



Figur 2: Gantt Chart del 1/3



Figur 3: Gantt Chart del 2/3



Figur 4: Gantt Chart del 3/3

### 3.4.3 Vurdering av planleggingen og gjennomføringen

Når vi ser tilbake på hvordan vi planla prosjektet og hvordan det gikk i praksis er vi stort sett fornøyde. Arbeidet vi gjorde i planleggingsfasen la godt til rette for gjennomføringen. Da vi satte sammen rekkefølge og arbeidsmengde for de ulike brukerhistoriene var vi klar over at estimeringene kunne avvike fra faktisk tidsbruk. Selv om vi var forberedt på endringer og det var mye usikkerhet rundt faktisk tidsbruk, så ser vi i etterkant at vi undervurderte tidsbruken på de oppgavene vi var minst sikre på.

I planleggingsfasen gjennomførte vi også en risikoanalyse for prosjektet. Denne er en del av prosjektbeskrivelsen (se vedlagt fil «Prosjektbeskrivelse»). I gruppemøtet hvor vi ferdigstilte denne analysen var vi veldig klare på hvilke typer risikoer som kan fremkomme og hvordan vi kan håndtere dem. Da kom vi også frem til at det er begrenset med tiltak vi kan iverksette på forhånd for uforutsette hendelser utenfor prosjektet, og at vi dermed måtte se an enkelte situasjoner dersom de skulle oppstå. I løpet av prosjektet oppstod det både kortsiktig og langsiktig fravær. Løsningene i de tilfellene ble at vi tilpasset oss og fremdriftsplanen med både utsettelse og ny ansvarsfordeling av oppgaver.

## 4 Faglig beskrivelse

### 4.1 Brukerhistoriene

#### 4.1.1 Kort om arbeidet

I planleggingsdelen forberedte vi en fullstendig liste over hvilke funksjonaliteter vi ønsket gjennom brukerhistorier. Disse ble forberedt til gruppemøter og gjennomgått i felleskap under møtene. Når brukerhistoriene var klare, gjennomførte vi en «Planning poker» for å estimere tidsbruk. Dette ble gjort under nettmøter og på nettsiden *Planning Poker Online*. Da estimerte hvert gruppemedlem oppgavens størrelse for å så diskutere resultatet og eventuelt «stemme» på nytt. Det var alltid den høyeste estimeringen for hver brukerhistorie som ble den brukerhistoriens estimerte tid. Når «Planning Poker» var gjennomført ble tabellen som er beskrevet i neste del lagd. (Planning Poker Online, u.d.)

#### 4.1.2 Tabell med brukerhistoriene

I tabellen nedenfor er prosjektets brukerhistorier listet. Disse brukerhistoriene er de som originalt ble planlagt å gjøre fra starten av. Det er totalt 34 brukerhistorier i tabellen. Listen viser hvilke brukerhistorier som skal gjøres først, og er derfor sortert etter «Sprint/Prioritet». Med prioritet har vi fordelt alle brukerhistorier på tall fra 1-4, der 1 er høyest prioritet og 4 er lavest. Med prioritet 1 og 2 er brukerhistoriene i større grad nødvendige for en funksjonell side som utfører det viktigste Playfu skal gjøre. Med prioritet 3 og 4 er det funksjoner som ikke i like stor grad vil påvirke andre brukerhistorier og dermed prosjektets grad av funksjonalitet. Kolonnen «Sprint» beskriver når brukerhistorien skal gjøres. Kolonnen «Poker» viser resultatet av «Planning Poker». Vi valgte T-skjorte-størrelser for å representere hvilke mengder arbeid oppgavene krever. Det varierte fra XXS som vil si lavest vanskelighetsgrad til XXL som har høyest vanskelighetsgrad. Valg av når oppgavene skal gjøres (Sprint) er et resultat av brukerhistorienes prioritet og størrelse (Poker).

Nummer	Brukerhistorie	Prioritet	Sprint	Poker
01	Som en gjest vil jeg kunne registrere meg	1	1	S
02	Som et medlem vil jeg kunne logge meg inn	1	1	S
05	Som et medlem vil jeg kunne bli med i et rom	1	3	M
06	Som et medlem vil jeg kunne opprette et eget rom	1	2 til 3	S
08	Som et medlem vil jeg kunne endre passordet mitt	1	4	XS
09	Som et medlem vil jeg kunne bruke glemt passord funksjon	1	5	M
11	Som et medlem vil jeg kunne koble til min Discord konto	1	5	M
12	Som et medlem vil jeg kunne bli lagt til i discord samtaler	1	6	M
14	Som et medlem vil jeg kunne finne rom i spesifikke spill	1	5	S
15	Som et medlem vil jeg kunne chatte med andre i rommet	1	4 til 6	XL
16	Som et medlem vil jeg kunne se en liste over rom	1	2 til 3	M
10	Som et medlem vil jeg kunne koble til min Steam konto	2	2	L



25	Som et medlem vil jeg kunne slette kontoen min	2	4	S
32	Som et medlem vil jeg kunne hentet ut all dataen om meg selv	2	4	S
34	Som et medlem vil jeg kunne få romforslag basert på mine interesser	2	6	M
04	Som et medlem vil jeg kunne sette min lokasjon	3	4	XS
07	Som et medlem vil jeg kunne være admin i et rom	3	5	XS
17	Som et medlem vil jeg kunne endre min profilinfo	3	4	S
19	Som et medlem vil jeg kunne se en aktivitetslogg	3	6	M
20	Som et medlem vil jeg kunne rate andre medlemmer	3	6	XS
21	Som et medlem vil jeg kunne kontakte support	3	7	M
23	Som et medlem vil jeg kunne endre profilbilde	3	6	S
03	Som et medlem vil jeg kunne sette mitt kjønn	4	1	XS
13	Som et medlem vil jeg kunne søke etter andre medlemmer	4	8	M
18	Som et medlem vil jeg kunne følge andre medlemmer	4	8	S
22	Som et medlem vil jeg kunne blokkere andre medlemmer	4	9	S
24	Som et medlem vil jeg kunne kjøpe meg premium abonnement	4	10	XXL
26	Som et medlem vil jeg kunne lese nyheter på spill	4	9	M
27	Som et premium medlem vil jeg kunne opprette premium rom	4	10	S
28	Som et premium medlem vil jeg kunne bli med i premium rom	4	10	XXS
29	Som et premium medlem vil jeg kunne se hvem som er inne på min profil	4	10	S
30	Som et premium medlem vil jeg kunne ha unike emojis	4	11	XL
31	Som en bedrift vil jeg kunne legge ut nyheter til spill jeg eier	4	HOLD	XL

33	Som et medlem vil jeg kunne se Twitch-siden til et spill	4	HOLD	L
34	Som et medlem vil jeg kunne endre bakgrunnsbildet på min profil.	4	11	M

Tabell 3: Brukerhistorier

Vi har også laget et «use case» diagram for å gi en visuell beskrivelse. Du kan se diagrammet nedenfor.



## 4.2 Endringer

### 4.2.1 Endringer underveis

Noen av brukerhistoriene vi hadde planlagt innebar mulighet for å bli premium-bruker, altså en bruker som får flere muligheter eller fordeler hos Playfu gjennom betaling. Vi valgte å tilrettelegge muligheten for å implementere dette senere i designet, men avgjorde å ikke lage noe av funksjonaliteten knyttet til dette. Dette er hovedsakelig siden det å legge til en betalingsløsningen kan by på mye tidsbruk og utfordringer. Deriblant ville det kreve høy grad av sikkerhet og løsninger som mødfører kostnader. Ettersom Playfu ikke er et kommersielt prosjekt valgte vi å nedprioritere dette området i forhold til andre områder.

Planen var å utvikle Discord-boten i programmeringsspråket Rust. Istedenfor så ble denne utviklet i C#. Grunnen til dette var rett og slett tid og kompetanse. Planen var å lære seg Rust og utvikle Discord-boten i det programmeringsspråket. Dette var et ønske fra Sivert, han hadde til og med kjøpt et kurs og var klar for å skaffe seg kompetansen. På grunn av forskyvinger i fremdriftsplanen og større omfang av prosjektet enn antatt så ble dette endret. En annen grunn er også hosting av denne Discord-boten. Ved å kjøre Discord-boten på en helt egen tråd sammen med backend så slapp vi å hoste den som en egen applikasjon.

### 4.2.2 Ufullførte brukerhistorier

I tillegg til brukerhistoriene om premium-brukere som er nevnt tidligere, så har vi ikke lagt til muligheter for å legge ut nyheter på spill-sider og oppkobling til spill sine «Twitch-sider». Vi prioriterte heller å forbedre andre deler av applikasjonen.

Playfu er hovedsakelig et prosjekt for å vise hva vi kan og utfordre oss på nye områder. Intensjonen var aldri å drive denne applikasjonen kommersielt. Dersom vi skulle gjort det så ville vi heller fokusert mer på å tilpasse Playfu for å bli populær. Da ville områder som markedsføring og forretningsutvikling blitt veldig relevant. En vei for å oppnå mulighet for å bli populær er å tilpasse siden for deling via link eller ulike sosiale/gaming-nettverk. Funksjonaliteter for premium-brukere ville blitt mer aktuelt og betalingsløsning ville blitt etablert. Vi ville satt på plass rutiner for BI (Business Intelligence) for å analysere bruken av applikasjonen. Med bruk av BI for Playfu ville vi hatt mulighet for å forbedre brukeropplevelsen, økt bruken av applikasjonen og økt graden av brukere som velger å betale

for tilleggstjenestene. På sikt ville sannsynligvis målet blitt å utvikle applikasjonen sånn at Playfu ikke bare var et mellomledd for Discord, men en konkurrent med Discord.

Det er heller ikke koblet opp muligheten til å hente ut data om seg selv rett i applikasjonen. Det er derimot laget en funksjonalitet for det i backend, men det er ikke laget noe mulighet for å automatisk sende denne dataen til brukeren. Dette var noe vi ikke rakk, derfor har vi heller løst dette ved at bruker kan spørre om data over support. Data må da hentes ut manuelt ved hjelp av backend funksjonaliteten.

### 4.3 Utfordringer

Underveis i prosjektet har vi støtt på mange forskjellige utfordringer. Noen utfordringer har ikke hatt noen konsekvens, mens andre utfordringer har gått utover fremdriftsplanen. Under har vi laget en liste over utfordringer som har hatt konsekvens på utførelsen av prosjektet.

#### 4.3.1 Fravær

Gjennom prosjektet har det oppstått både kortsiktig og langsiktig fravær som har påvirket prosjektet. Kortsiktig fravær ble vurdert som svært sannsynlig under risikoanalysen og var derfor forventet, mens langsiktig fravær var vurdert som lite sannsynlig og kom ganske uforventet. I respons på kortsiktig fravær ble oppgavene satt på vent. Mens i respons på langsiktig fravær ble flere av oppgavene fordelt på andre i gruppen og gjennomført frem til fraværet var over.

Vi ble midlertidig 1 mindre på gruppen i en lengre periode på grunn av langsiktig fravær. Vi måtte da gjøre endringer i prosjektet som nevnt over. Midlertidig endring ble følgende:

Sivert: Ble tildelt rollen databaseadministrasjon

Ørjan: Fikk alt arbeid på frontend

Govert: Ble tildelt rollen diagrammer

### 4.3.2 Steam

En annen utfordring som oppstod var problemer innenfor utvikling av funksjoner hvor bruker kobler seg opp mot Steam og funksjonen «Lobby». Problemene relatert til dette var knyttet til kompetanse og estimert tidsbruk. Denne typen oppgave var ukjent for gruppen, og det fantes ikke mye ressurser eller alternative løsninger for å finne ut av det. Da vi estimerte tiden for oppgaver relatert til Steam og Lobby var vi klar over at det var usikkert og at det dermed kunne ta mye tid. Derfor ble disse oppgavene vurdert som store. Etter planen skulle disse oppgavene være ferdig tidlig i prosjektets utviklingsfase, men ble arbeidet med over størsteparten av utviklingsfasen. Disse oppgavene var av høy prioritet og det var derfor riktig å plassere dem på starten av utviklingens fremdriftsplan, men i etterkant kan vi se at planen burde gitt disse oppgavene lengre tid.

### 4.3.3 Steam sine API-er

Vi har brukt to API-er fra Steam. Steam sine API-er er litt spesielle, de er ikke som de fleste andre API-er. De er gamle og følger ikke gode design-prinsipper som er forventet i dag. Det første API-et deres er mer rettet mot Steam generelt. Dette benytter vi for å skaffe oss en liste over alle appene som Steam har. Problemet med dette er at man får tilbake alt som er en applikasjon på Steam, og det er ingen filtreringsmetode her. Dette betyr at i denne listen så ligger det DLC, Soundtracks, verktøy osv. Dette er informasjon vi ikke vil ha med i søkefunksjonaliteten. Det er heller ikke spesifisert hvilken type app som hver «Entity» har fra dette API-et.

Det andre API-et er mer rettet mot butikken til Steam. Det er dette som butikken til Steam benytter, og her kan du sende over en ID som gir deg tilbake all informasjonen som er relatert til den ID-en. Tingen med Steam er at de har en «rate limit» på 200 forespørsler hvert 5 minutt. Det er nå over 139.000 apper i den listen. Det vil si at det ville ta rundt 58 timer å fullføre en forespørsel for hver eneste app og filtrere ut de som ikke er spill. Løsningen vi kom frem til da var å lage en egen app som gjorde dette for oss. Denne arbeidet da i 58 timer, og når den var ferdig så lagret den informasjonen i databasen. Denne appen ble utviklet i .NET som en konsollapplikasjon og «dockerized» slik at Sivert kunne hoste den på sin egen server.

#### 4.3.4 CORS

*CORS (Cross-Origin Resource Sharing)* er en mekanisme som involverer *HTTP*-tittelens tilkobling til servere for å få tilgang til å bruke API-er. Behovet for *CORS* oppstår når en ikke klarer å ta i bruk API-er når det ikke er tillatt av nettleserens «CORS Policy». For å løse dette kan en laste inn et «CORS-bibliotek» og bruke det i koden for henting av API-et. (Mozilla, 2022) (Web Dev Simplified Blog, 2021)

På grunn av dette så fikk vi litt problemer med hva som var tillatt. Forståelsen vår angående *CORS* var veldig liten, og vi måtte bruke mye tid på å lære hva *CORS* gikk ut på. Det er et komplisert tema og det er mye som er utenfor hva vi har muligheten til å endre på da dette er innebygd i nettleseren. Det var hovedsakelig “*access-control-allow-origin*” som ga oss problemer. Dette er en header som sier hvem responsen kan bli delt med fra forespørselen sin «*Origin*». Tingen her er at dette er en server-basert header, den er bare tilgjengelig i en respons fra en server. Denne serveren må da spesifisere hvilke «*Origin*» som har lov til å få denne responsen. Det er her Discord og Steam kommer inn, da vi ikke kan endre serveren på deres side.

Etter en god del research så kom vi frem til problemet. *CORS* har en mekanisme som gjør at den ikke blir iverksatt på hver eneste forespørsel. Den blir bare utløst dersom det er brudd på et kriterium. Dersom front-end JavaScript kode tar i bruk «*XMLHttpRequest*» for å utføre en forespørsel så vil nettleseren sende en “preflight” forespørsel. I vårt tilfelle så har vi satt det opp slik at Angular sender disse forespørselene. Dette utløser mekanismen “preflight” som sjekker om “*access-control-allow-origin*” har riktig origin som forespørselen. For å komme rundt dette så tok vi i bruk en “a href” istedenfor en *POST* fra Angular. Dette gjorde at forespørselene gikk rett fra *HTML* og ikke igjennom Angular. Mekanismen “Preflight” blir da ikke utløst og *CORS* sier at alt er greit.

#### 4.3.5 Hosting

Prosjektet vårt tar i bruk mange forskjellige tjenester som må bli hostet et sted. Utfordringene her var prisen og hvor tjenestene skulle bli hostet. Heldigvis så tilbyr Azure 1000 kroner i kreditt dersom du er student. Dette var helt greit, og vi hadde mer enn nok til å kunne hoste i minst et par måneder etter sensur. Det var helt til vi skulle sette opp Meilisearch på Azure. Selve backend-en vår koster rundt 50-100 kroner i måneden, mens Meilisearch kostet opptil 300 kroner i måneden. Dette gjorde at vi måtte finne andre muligheter. Azure tar betalt i forhold til hvor mye ressurser som blir brukt, og da øker kostnadene eksponensielt. Heldigvis så har Github noe som heter Github Student Developer Pack og USN studenter er kvalifiserte for denne pakken. Dette ga oss 1000 kroner på Digital Ocean. Digital Ocean har en fast kostnad per måned på 50 kroner noe som var mer enn nok for det vi trengte.

#### 4.3.6 Implementasjon og feil

Vi glemte å oppdatere koden for produksjon. Mye ble forbedret i koden mot slutten av utviklingen, men da oppstod det nye feil som krevde en del arbeid for å løse. For å løse problemene brukte vi parprogrammering og kontinuerlig testing. Når vi tester nettsiden krever det at flere er aktive samtidig eller at flere brukere er innlogget av samme person. Vi brukte også test-brukere til å teste imens vi utviklet, det viste seg senere at når vi registrerte egne brukere ble det mer problemer.

#### 4.3.7 Server- og kode-effektivitet

I forkant av brukertestingene testet vi nettsiden selv og oppdaget at serveren krasjet ved litt bruk. For å løse dette oppgraderte vi til en bedre server slik at dette ikke skjedde igjen. I tillegg forbedret Sivert koden for å gjøre nettsiden mer effektiv og mindre krevende for serveren.

### 4.4 GDPR

Applikasjonen lagrer personlig informasjon om brukeren. Dette innebærer hovedsakelig epost, men applikasjonen har også funksjonalitet for bildeopplastning av profilbilde. Her kan brukeren laste opp bilde av en avatar, eller seg selv. Vi har derfor tatt forhåndsregler angående GDPR på grunnlag av dette. (Datatilsynet, u.d.)



Formålet med å lagre eposten er først og fremst for innlogging, men den brukes også for å kunne kontakte brukeren som tar i bruk kontaktskjema for support. Bildeopplastning er frivillig, og brukeren må selv ta forhåndsregler om hva de laster opp. Samtykke blir gitt ved at brukeren lager en konto, dette er spesifisert inne på registreringssiden. Brukeren har alltid muligheten til å slette kontoen sin, endre informasjon og sende en forspørsel om data vi har lagret som er knyttet til brukeren. (Datatilsynes, u.d.)

## 4.5 Utviklingsmetode

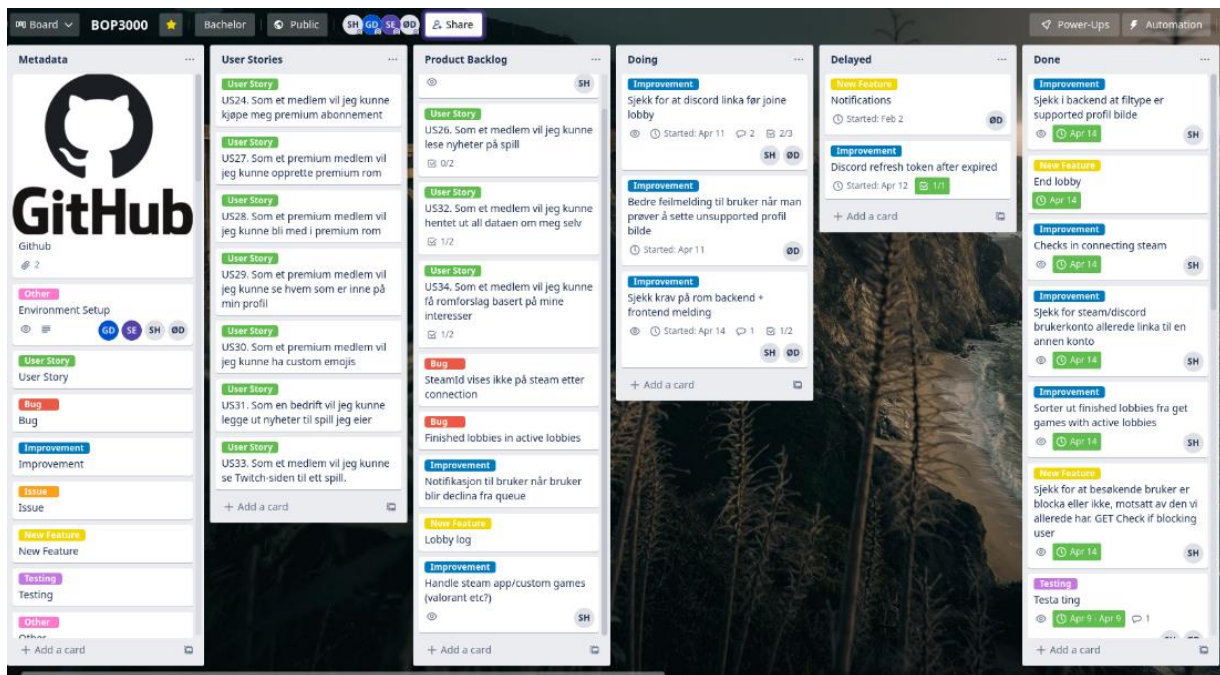
Utviklingsmetoden som vi har fulgt er smidig systemutviklingsmetode med en blanding av forskjellige hjelpemidler. Vi har tatt i bruk Scrum, Kanban-tavle og sprint-basert utvikling. Scrum og Kanban er begge smidige metoder som benytter seg av et pull-system. For oss var det helt naturlig å benytte denne utviklingsmetoden på grunn av erfaring fra tidligere emner. Vi lagde en kanban-tavle med ulike brukerhistorier i en product backlog. Deretter drev vi prosjektet fremover ved å bruke sprint-basert utvikling. For hver sprint så hadde vi planlagt hvilken brukerhistorie som skulle gjøres og flyttet disse rundt i kanban-tavlen.

### 4.5.1 Planleggingsfasen

Planleggingsfasen pågikk en god stund, faktisk hele januar ble for det meste brukt til planlegging og oppsett. Vi lagde mange brukerhistorier og lagde en fremdriftsplan for prosjektet. Deretter lagde vi en oversikt over alle sprintene og ulike milepæler, og lagde en plan for når brukerhistoriene skulle gjøres. Vi opplevde det som svært produktivt å ha en oversikt som sa hvilke oppgaver som skulle gjøres til en hver tid.

### 4.5.2 Kanban

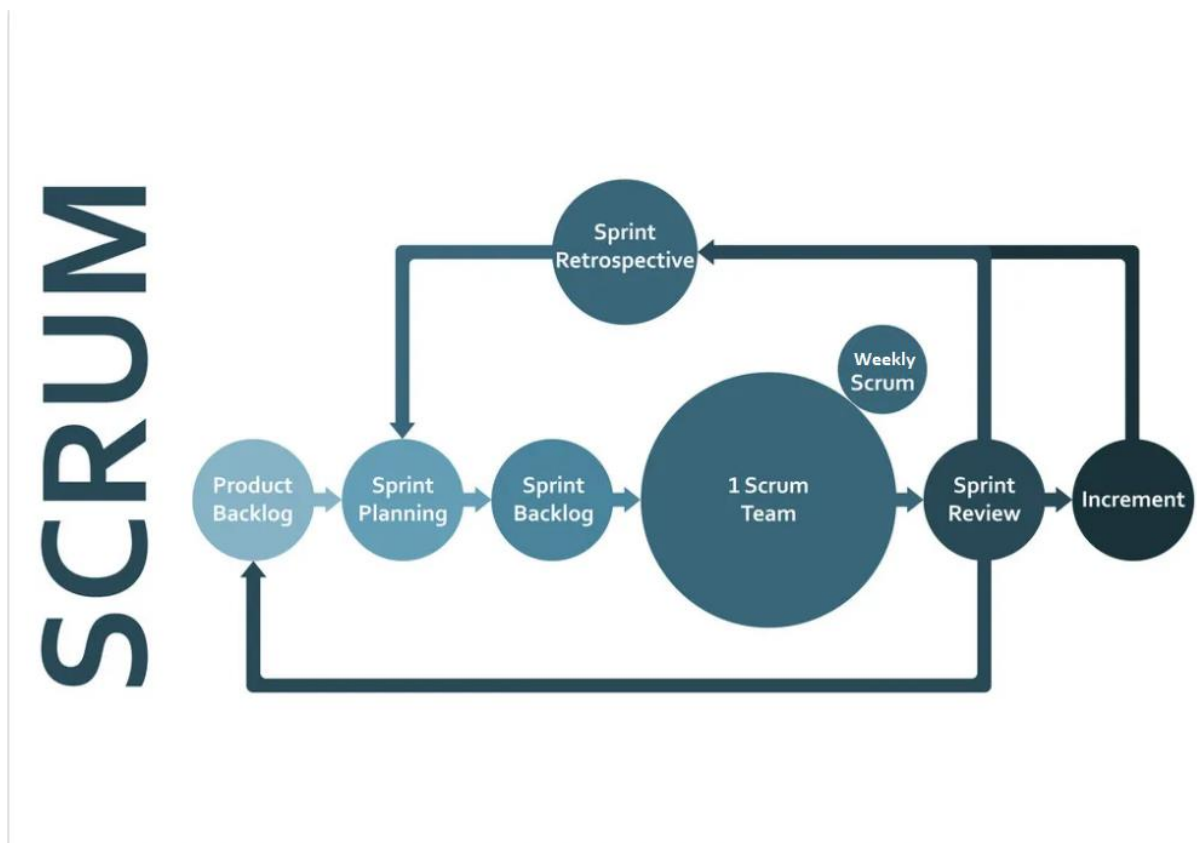
Vi benytter verktøyet Trello for å ha en Kanban-tavle. Dette verktøyet har vi brukt tidligere i andre prosjekter. Her har vi også lagt inn noen automatiseringer for å gjøre arbeidet litt lettere. For eksempel når en oppgave blir flyttet til kolonnen «Doing» vil kortet automatisk få dette tidspunktet som «Start Dato». Deretter om kortet blir flyttet til kolonnen «Done» så vil kortet få en slutt-dato satt til tiden det ble flyttet dit. Denne automatiseringen hadde vi ikke i starten og det ble ofte glemt å sette start/slutt dato. Vi hadde også integrert TeamGantt med Trello, men dessverre ville ikke den integrasjonen fungere som vi ønsket. Kanban ble brukt til å holde styr på hva som skal og har blitt gjort i hver sprint. I tillegg så ser vi hva hvert medlem jobber med. Under ligger et bilde av hvordan Kanban-tavlen vår så ut midt i april.



Figur 6: Trello

## 4.6 Scrum

Benyttelse av Scrum har gruppen hatt god erfaring med fra tidligere, og alle visste da hvordan vi skulle gå frem. Scrum er et rammeverk som hjelper teams å jobbe sammen. I Scrum så er det fastsatt hva som skal gjøres i hver sprint, og den skal ikke endres dersom avvik skjer. Vi valgte å være litt mer åpne her, på grunn av kompetansen vi måtte skaffe oss underveis. Men vi holder oss uansett innenfor sprintene vi har på en uke. Dette betyr at dersom noe har blitt forsinket og ikke gjort ferdig i forhold til planen, så vil det bli overført til neste sprint.



Figur 7: Flytdiagram av Scrum metoden vi har tatt i bruk

#### 4.6.1.1 Metoder

##### - ***Sprint retrospektiv***

Etter hver sprint så hadde vi en «Scrum Retrospektiv». Det var her vi gikk over hva som hadde blitt gjort. Alle gikk over oppgavene sine selv og forklarte i detalj hva de hadde utført i sprinten. Vi gikk også over hva som manglet og diskuterte arbeidet. Her viste vi også frem nye funksjoner i applikasjonen.

##### - ***Sprint planlegging***

Etter retrospektiv så tok vi planlegging av neste sprint. Her gikk vi ut fra planlagte oppgaver fra fremdriftsplanen, men tok også en vurdering av uløste problemer og endringer som måtte bli gjort. Dersom noe ikke ble ferdig i forrige sprint, ble det overført til neste sprint.

- ***Scrum Master***

Scrum Master har ansvaret for å opprettholde Scrum igjennom utviklingen og sikrer at alt utføres i henhold til metodikken. Vi valgte å ha en Scrum Master i prosjektet ettersom det gir en trygghet for at Scrum blir opprettholdt hele veien. Istedenfor å sette en fast person, så valgte vi å endre over visse perioder slik at alle på teamet vårt fikk forsøkt å være Scrum Master.

- ***Scrum Poker***

Scrum Poker blir brukt for å kunne estimere forskjellige grader, i vårt tilfelle vanskelighetsgraden på hver brukerhistorie. Dette gjorde vi fordi vi ønsket en viss estimering på hvor lang tid hver brukerhistorie ville ta. For å gjøre metoden litt mer «moro» så gjorde vi T-skjorte versjonen. Dette gikk veldig fint og resultatet ble bra, dette gjorde av vi fikk et bedre overblikk over prosjektet sitt omfang.

- ***Parprogrammering***

Av og til kom vi over utfordringer som vi ikke klarte å løse hver for oss. For å løse disse utfordringene så brukte vi ofte parprogrammering. Dette betyr at du sitter sammen med noen andre og skriver kode. Det er da en person som skriver kode, mens den andre personen observerer. Vi brukte også parprogrammering når vi skulle utvikle noe nytt som vi ikke helt visste hvordan skulle fungere. Et eksempel er lobbysystemet i applikasjonen. Dette systemet ble for det meste skrevet ved hjelp av parprogrammering.

#### 4.6.2 Utviklingsverktøy

Igjennom prosjektet har vi tatt i bruk en rekke verktøy. Noen verktøy har gjort arbeidet lettere, mens andre har vært helt nødvendig for utførelsen. Mange av disse verktøyene har gruppen hatt erfaring med før, mens noen er helt nye for oss. Vi har laget to tabeller som har en kort beskrivelse av hva de går ut på. Den første tabellen viser planlagte verktøy, som vi planla i planleggingsfasen. Den andre tabellen viser nye verktøy som vi har tatt i bruk under utviklingen.

Navn	Beskrivelse
Visual Studio Code	Koderedigeringsprogram som er en fullverdig IDE. Den har integrerte verktøy som gjør det lettere for utvikling.
Trello	Webbasert program som gjør det mulig å dele et prosjekt inn i kolonner og samarbeide med andre. Blir ofte brukt i sammenheng med Kanban.
Github	Leverer tjenester for programvareutviklingsprosessen som tar i bruk versjonskontrollprosess ved bruk av GIT.
Git	Et versjonskontrollsystem som er designet til å kunne håndtere alt fra små til store prosjekter. Brukes ofte sammen med Github, men det finnes også andre leverandører som kan brukes istedenfor.
Azure	En «Cloud computing platform» som er laget av Microsoft. Leverer tjenester via internettet, for eksempel SaaS, Paas og IaaS.
Microsoft Teams	En kommunikasjonsplattform for bedrifter. Støtter tjenester som chat, videomøter, fil-lagring og andre applikasjonsintegrasjoner.
Discord	Gratis «VoIP-program» som har fokus på å være en digital distribusjonsplattform som er designet for Gaming-felleskapet. Denne

	plattformen er spesialisert på kommunikasjon mellom brukere.
Postman	En «API Platform» som gjør det mulig å bygge og bruke APIer. Den gjør det lettere for utviklere å teste API eller opprette bedre APIer.
Figma	Figma er et webbasert gratis verktøy for design av blant annet prototype design. (Wikipedia, 2022)
MySQL Workbench	Verktøy for databasedesign som integrerer funksjoner for arbeid med SQL. Kan brukes til blant annet å utvikle, administrere, designe, bygge og vedlikeholde SQL.
Visual Paradigm	Et «UML case» verktøy som støtter «UML 2», «SysML» og «Business Process Modeling Notation». Tilbyr også andre funksjonaliteter som reverse engineer, code generation og rapportgenerering mm.
TeamGantt	Et planleggingsverktøy for prosjekter som gjør det mulig å opprette «Gantt charts» på web. Har mulighet for teams og klienter som kan samarbeide på prosjekt.

Tabell 4: Planlagte utviklingsverktøy

Navn	Beskrivelse
Securityheaders.com	Verktøy til å scanne nettsider for «security headers» som sendes over HTTP/HTTPS.
Burp Suite	Inneholder et sett med verktøy for å kjøre penetrasjonstester i en webapplikasjon.
Sonar Qube	Verktøy for å inspisere kodekvalitet. Brukes til å oppdage bugs, unødig kode, gjentakende kode og anbefaler sikkerhetstiltak.
DBeaver	SQL-klient og databaseadministrator som tilbyr en rekke funksjoner. For eksempel «code completion», «syntax highlighting», modellering mm.
Freshdesk	Skybasert programvare for kundeservice som er koblet til kontaktskjema i web-applikasjonen. Veldig rik på funksjonalitet med livechat, email, telefon og sosial media.

Tabell 5: Nye utviklingsverktøy

### 4.6.3 Utviklingsbibliotek

Under utviklingen så har vi tatt i bruk en god del bibliotek som har hjulpet oss mye med utviklingen av løsningen. «NuGet» og «NPM» er pakkeleverandørene som vi har tatt i bruk i prosjektet. Se vedlagt fil «Bibliotek» for en oversikt over alle «NPM-pakker» og «NuGet-pakker» som vi har brukt og beskrivelse av disse.

## 4.7 Brukergrensesnitt

For utvikling av brukergrensesnittet har vi hatt tre personer med hvert sitt ansvar. Govert lagde prototype-design og Ørjan hadde hovedansvar for frontend sammen med Sigve. Prototypen ble lagd med verktøyet Figma. Figma er et gratis og lettvtint verktøy som det finnes mange gode kurs om på YouTube. Hensikten med prototypedesignet var å skape et bilde på kort tid av hvordan produktet vårt kunne se ut. Prototypen kunne da brukes til inspirasjon for videre utvikling av design og funksjonalitet.

### 4.7.1 Logoen



*Figur 8: Logo V.1*



*Figur 9.1: Logo V2, V3.1, V3.2, V3.3*

Tidlig i designet av prototypen oppstod logoen. Den første versjonen ble inspirert av designet i et kurs hvor en plasserte tilleggsinformasjon på toppen av overskrifter, på samme måte som «Gaming for everyone» står over Playfu logo V.1. For å gi logoen et preg av tilhørighet til spillverdenen er det valgt en passende font for teksten i logoen, og brukt et symbol av en spillkontroller. Dette symbolet kom fra Figma-utvidelsen Iconify (Iconify, n.d.). Deretter ble det naturlig å lage en mindre versjon av logoen, dette resulterte i den runde og simple logoen vi bruker i dag. Senere ble det også lagd nye versjoner av logoen i forskjellige farger for å kunne tilpasse oss andre valg av farger i designet.



## 4.7.2 Arbeidet med prototype-designet



Figur 10: Oversikt over prototypen i Figma

Arbeidet og kursing innen prototype-designet startet tidlig i januar. Govert hadde ikke noe erfaring med prototype-design fra før, derfor krevde det en del læring gjennom kurs og praktiske øvelser.

Under arbeidet av prototypen ble det fokusert mye på å gjøre applikasjonen så enkel som mulig å bruke, og med et minimalistisk design. Tanken var også at versjoner på mobil og nettbrett ville se ganske like ut, samtidig som applikasjonen kan ha en del til felles med enkelte spill og dermed føles gøy å bruke. Dette resulterte i runde kanter og former brukt i designet med store knapper og tekst. Valget av lilla som hovedfargen på dette designet var basert på et ønske om å ha en unik farge som er mindre brukt for å kunne skille oss ut. Denne fargen skal også sørge for at nettsiden ser morsom og uformell ut, da Playfu er en nettside for fritidsaktiviteter.

## 4.7.3 Arbeidet med utvikling av Design

Det endelige designet er inspirert av en kombinasjon av prototype-designet og eksisterende webapplikasjoner som Faceit og Esportal. Vi har tatt med egne og personlige erfaringer med liknende applikasjoner, forbedret det og gjort det til vårt eget på Playfu.

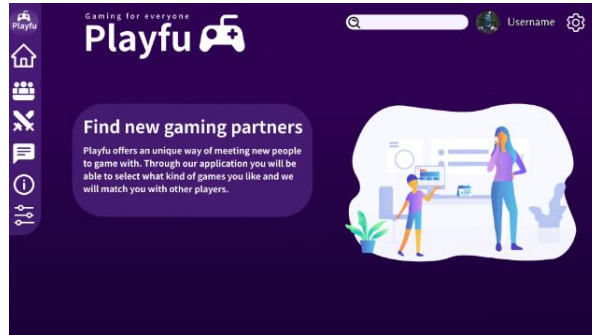
Designet har vært gjennom flere endringer gjennom hele prosjektet. Brukervennligheten har forbedret seg mye underveis basert på resultater fra brukertestene. Det er store forskjeller mellom prototype og endelig resultat på grunn av dette.

Et eksempel er da vi la til «Gå til lobby»-knappen som er festet til navigasjonsbaren. Vi fikk en tilbakemelding om at det kunne være vanskelig å finne tilbake til lobbyen du var i. Grunnen til dette var at det kunne være mange aktive lobbyer en måtte bla gjennom.

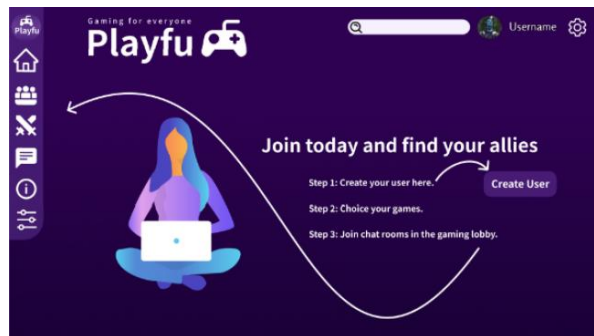
Vi har også to navigasjonsbarer, en på toppen og en på siden. Grunnen til dette er at det vil gjøre det lettere å utvikle utvidelser senere. I starten hadde vi heller ikke full kontroll på hvor mye plass de forskjellige elementene ville ta i navigasjonsbaren. Derfor tenkte vi det ville bli lite plass om alt var i en tradisjonell navigasjonsbar på toppen av siden. En ser også at navigasjonsbaren på siden blir mer og mer brukt hos store nettsteder.

#### 4.7.4 Eksempler på prototype og endelig brukergrensesnitt

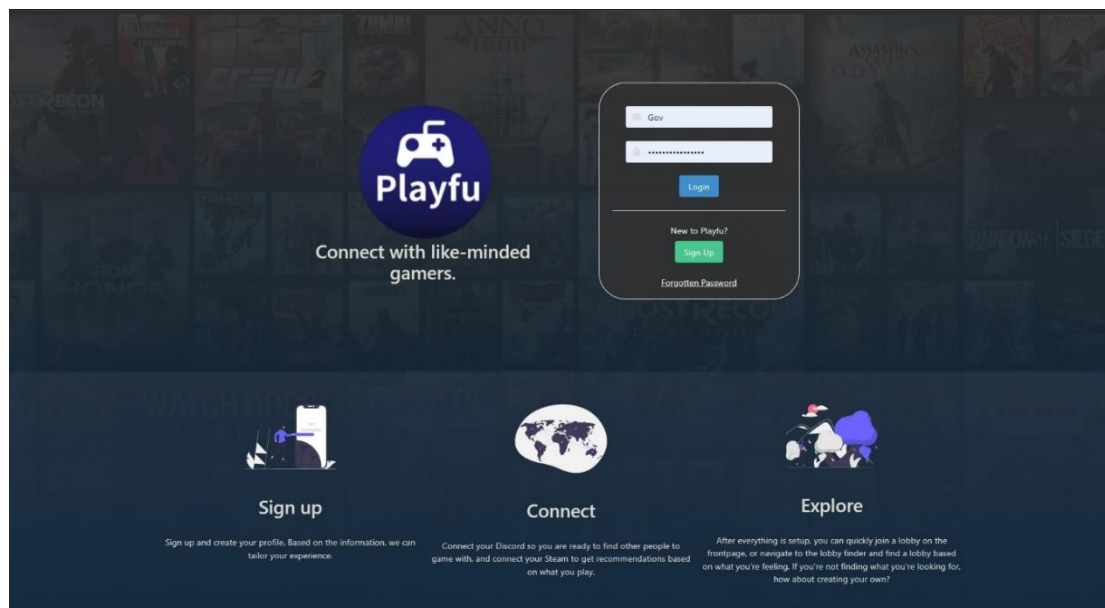
### Fremside



Figur 11: Skjerm bilde av fremsiden til prototypen



Figur 12: Skjerm bilde av fremsiden til prototypen ved ned-scrolling

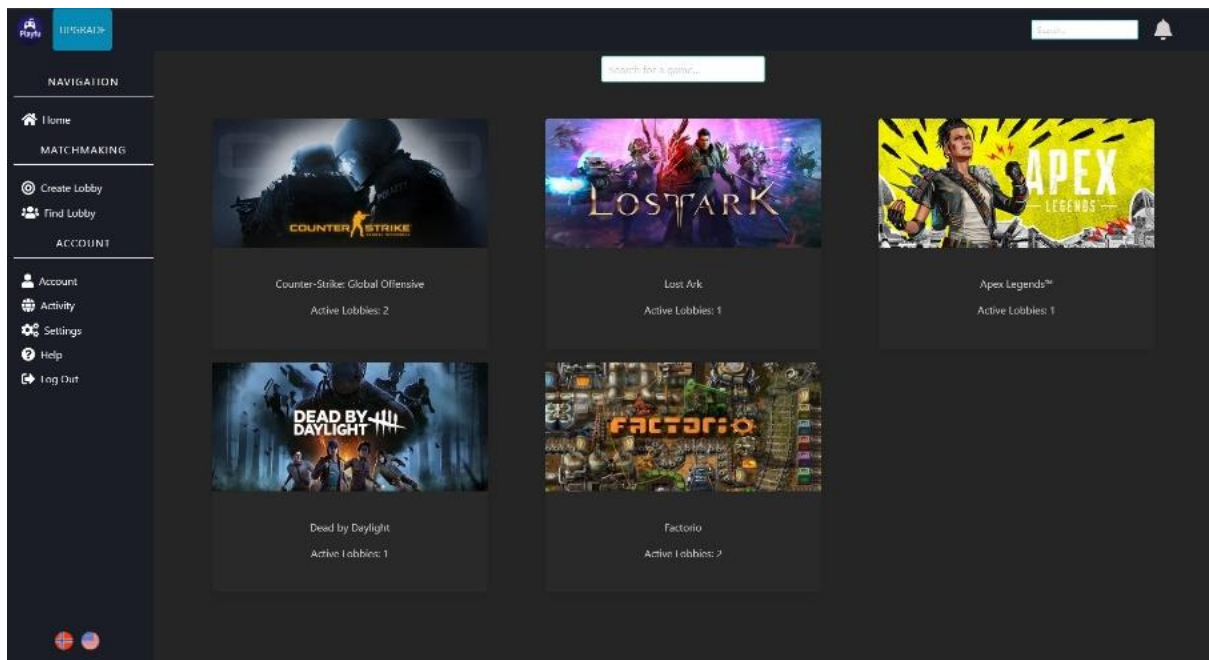


Figur 13: Skjerm bilde av fremsiden

## Spill Oversikt



Figur 14: Skjerm bilde av «Game Lobby» i prototypen

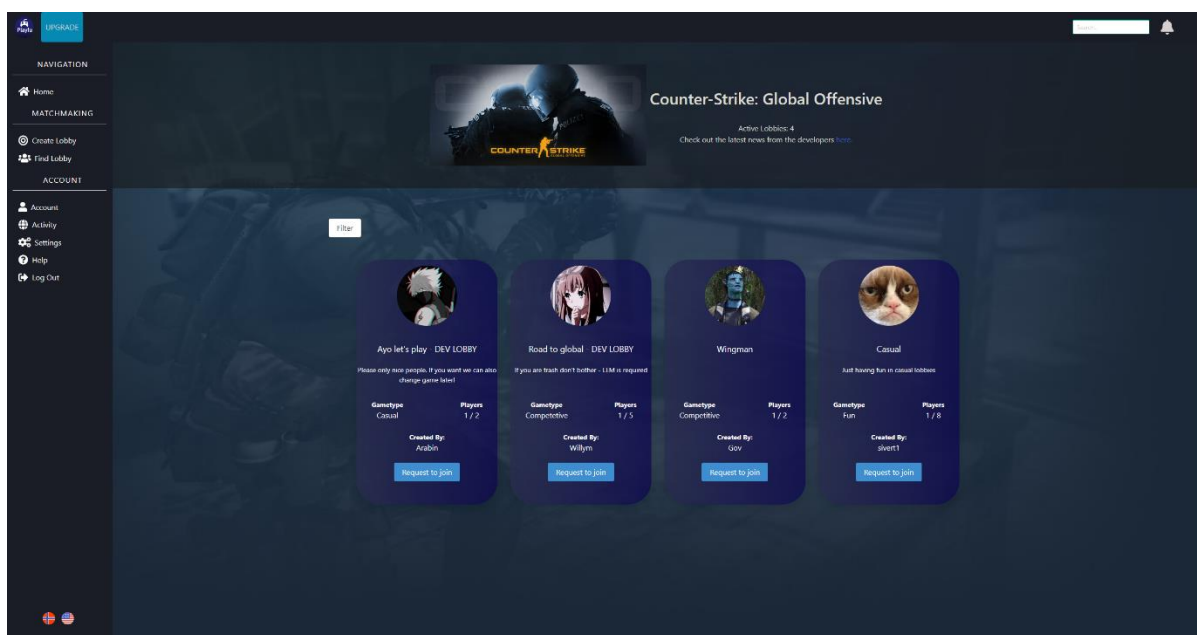


Figur 15: Skjerm bilde av Spill-oversikten

## Lobby oversikt



Figur 16: Skjerm bilde av «Lobby Rooms» i prototypen

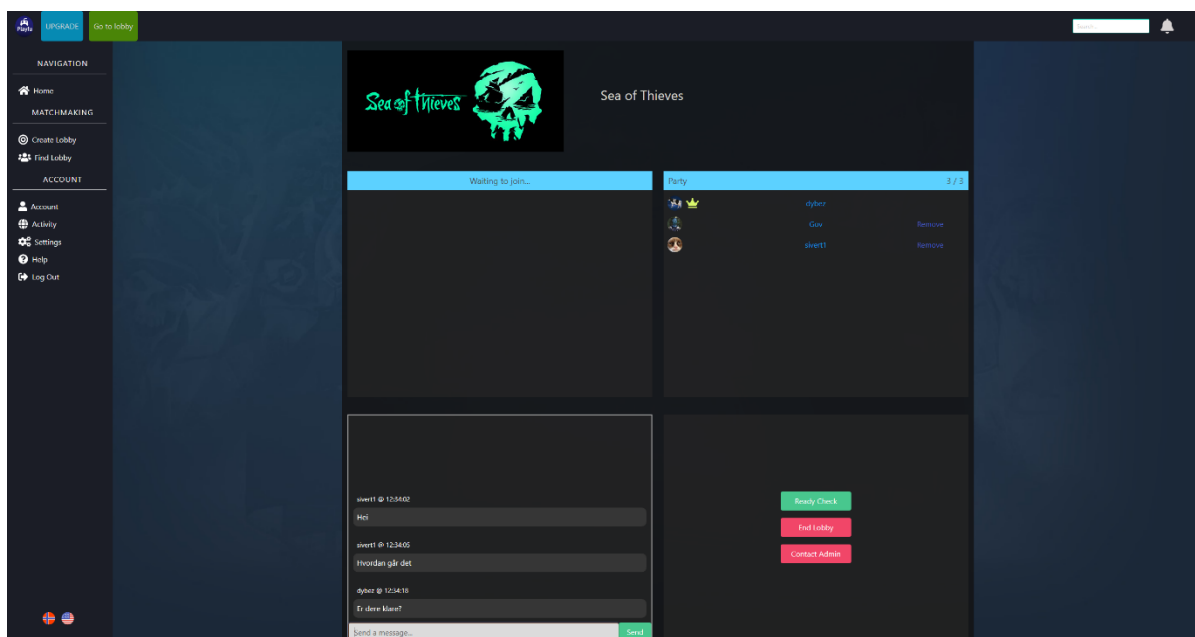


Figur 17: Skjerm bilde av Lobby-oversikten

## Chat Lobby



Figur 18: Skjerm bilde av Chat Lobby i prototypen

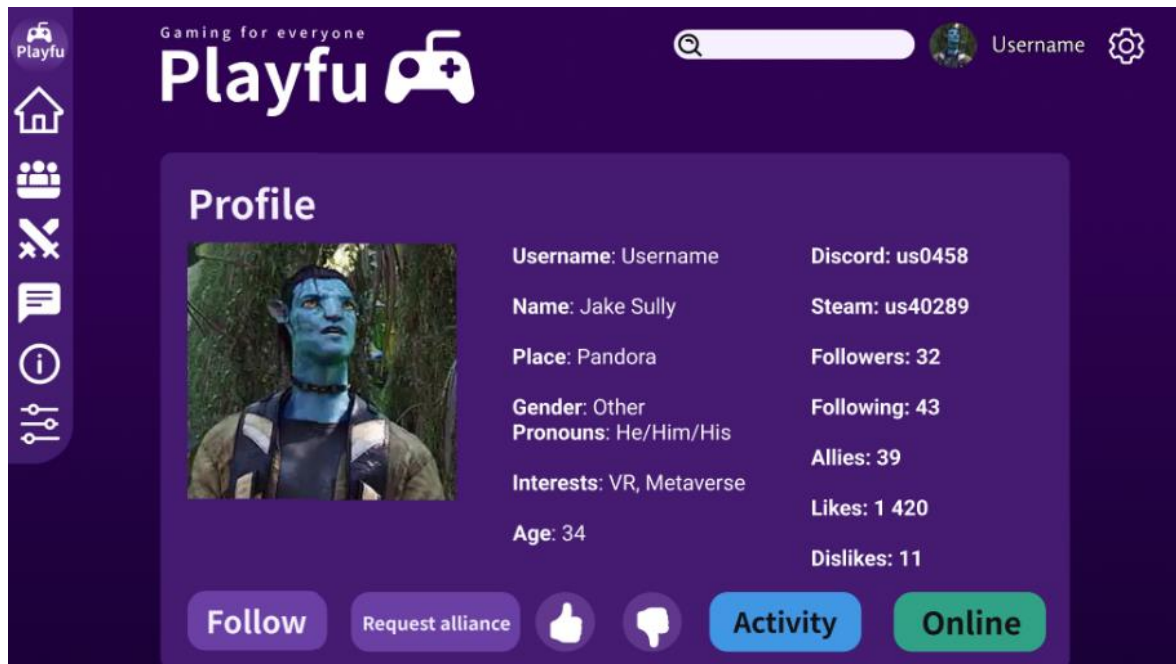


Figur 19: Skjerm bilde av Chat Lobby

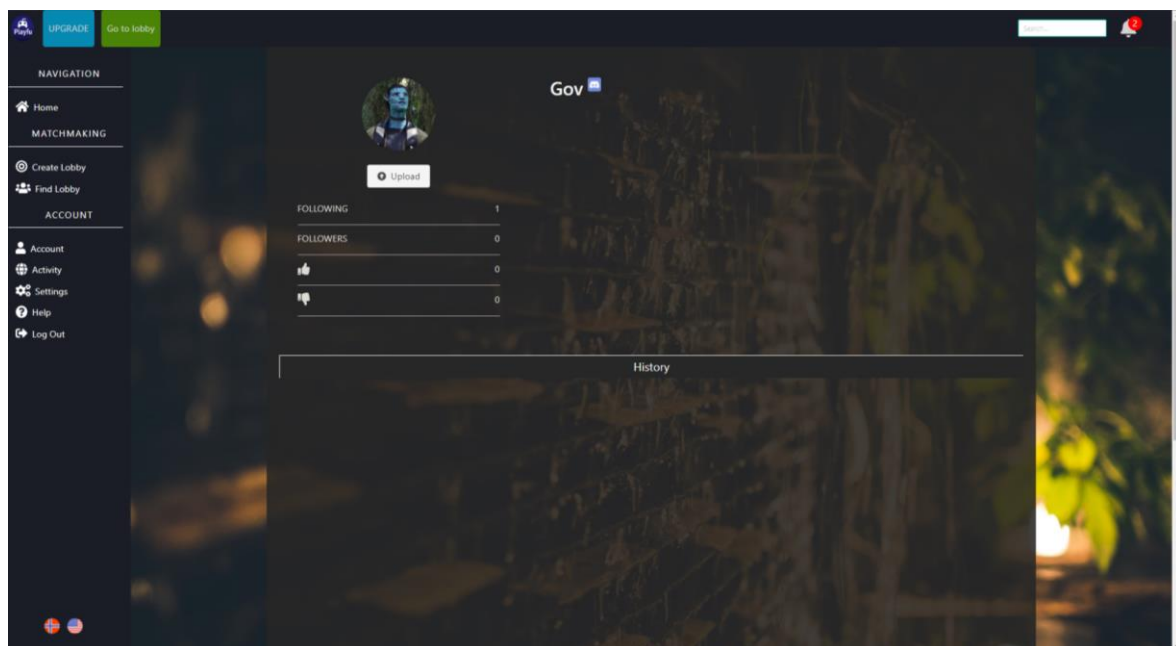
*Kommentar: Prototypen ble lagd i begynnelsen av prosjektet og medregnet derfor ikke oppkoblingen til Discord. Lobbyen inneholder spillere og en chat i begge, mens det endelige resultatet har mulighet til å knytte sammen gruppen på Discord.*



## Kontoside

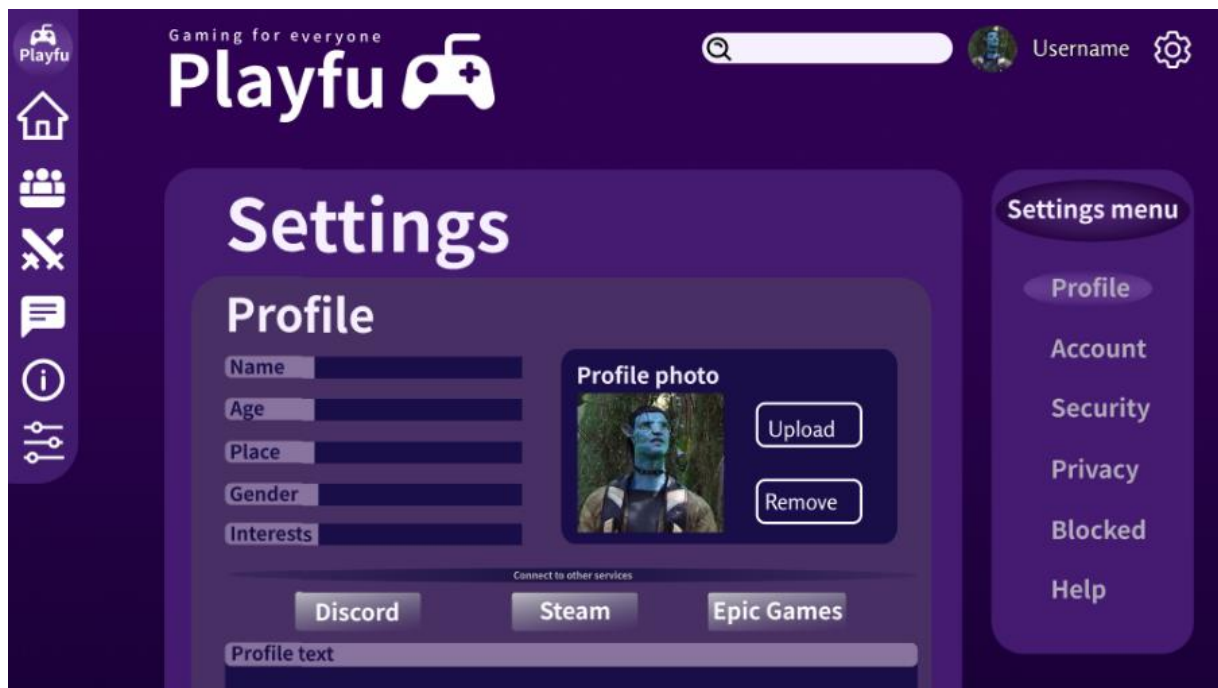


Figur 20: Kontosiden i prototypen

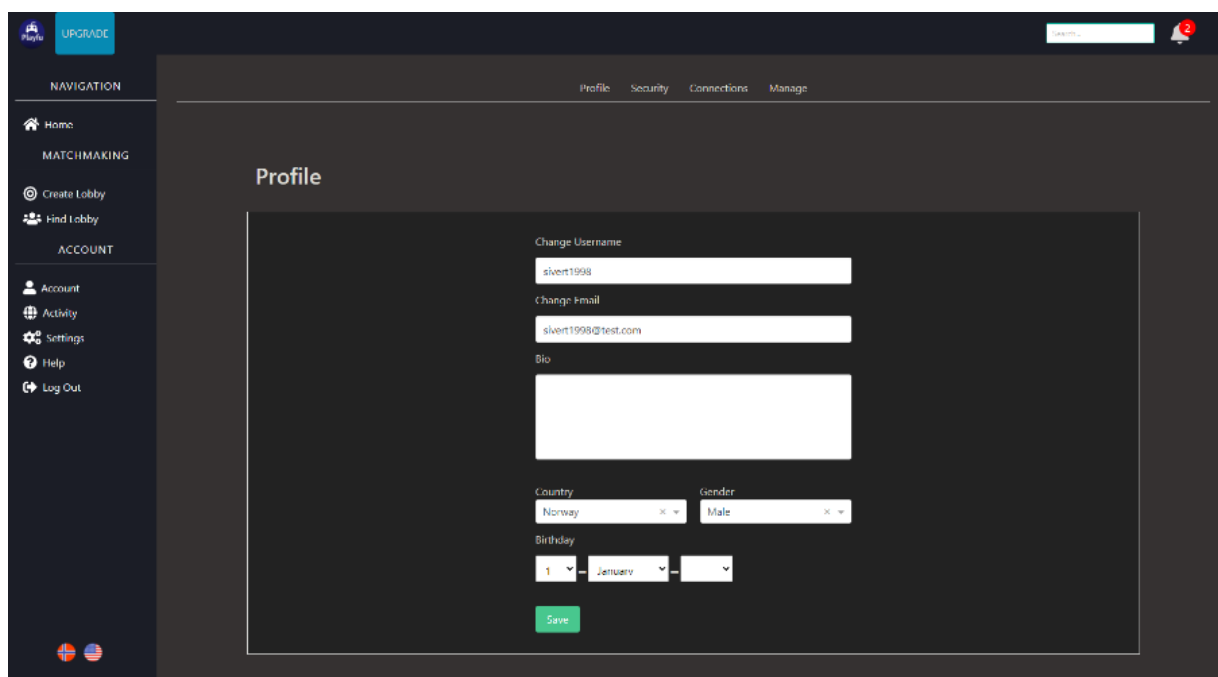


Figur 21: Kontosiden

## Innstillinger



Figur 22: Skjerm bilde av profilinnstillinger i prototypen



Figur 23: Skjerm bilde av profilinnstillinger



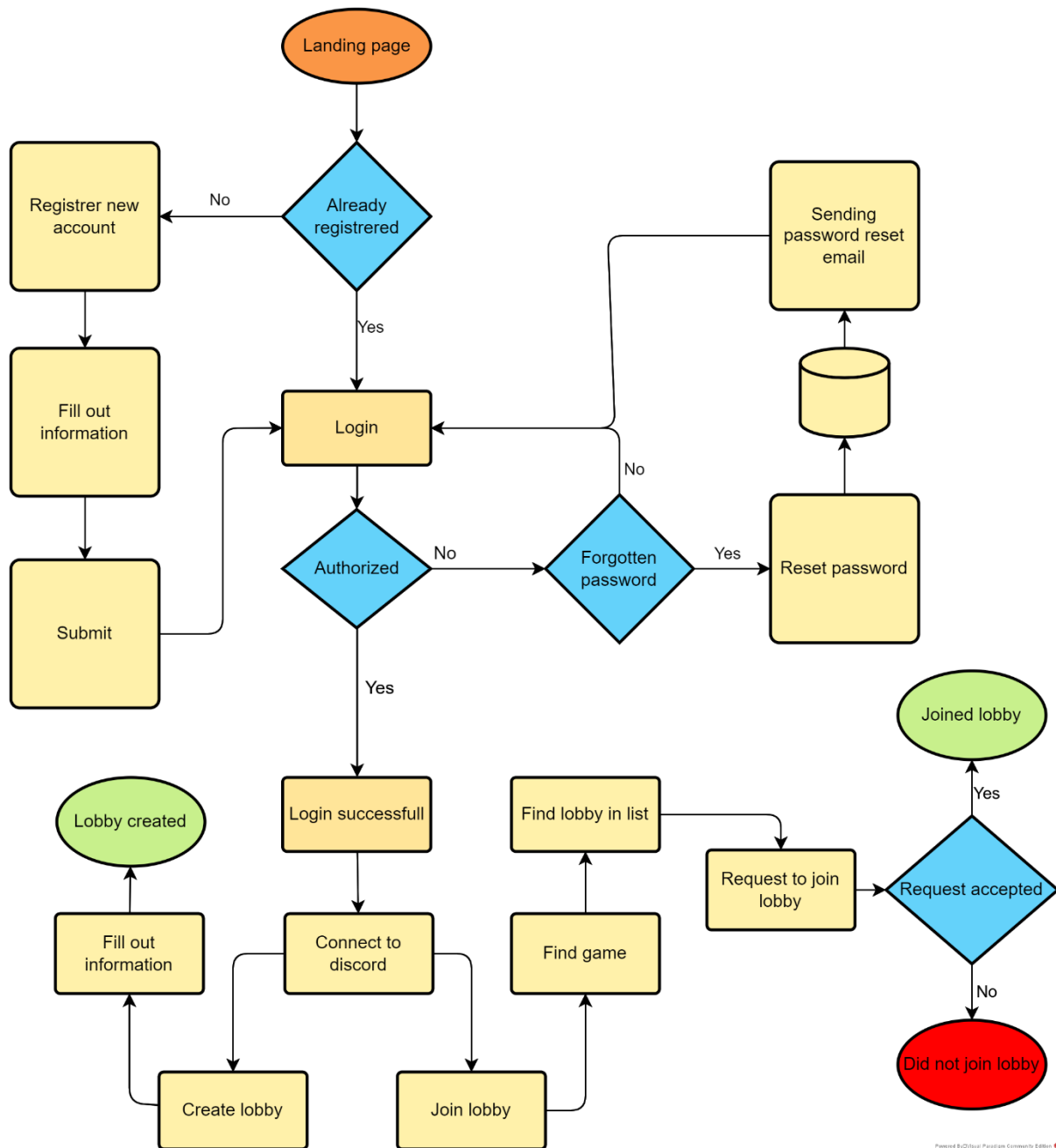
## 5 Systemdokumentasjon

### 5.1 Overordnet beskrivelse

Vi begynte veldig tidlig med å fremstille en plan for hvordan løsningen skulle bli bygd opp. Løsningen fungerer slik at vi har et API som klienten sender forespørsler til. Det er da «Controller» som tar imot HTTP metoder, og backend gjør da nødvendige operasjoner og returnerer en respons til klienten. For eksempel så vil klienten sende en forespørsel for å skaffe informasjon om brukeren, backend henter da denne informasjonen og sender det tilbake som en respons. Løsningen tar også i bruk «websocket» for å gjøre det mulig for sanntidskommunikasjon mellom klient og backend etter at nettsiden er prosessert. Det er hovedsakelig lobby som bruker denne kommunikasjonsmetoden.

Vi har også en Discord-bot og en søkemotor som heter Meilisearch. Med en bot mener vi et program som kan gjøre spesifikke oppgaver automatisk. Discord-boten blir «hostet» på samme backend som API-et ligger på. Discord-boten tar seg av opprettelse av kanaler og rettigheter tilknyttet Discord. Si at en lobby er klar for utførelse, så vil Discord-boten få en beskjed om å utføre nødvendige operasjoner for at brukeren skal kunne bli med i en kanal. Meilisearch tar seg av søkemuligheter i store mengder data med god responstid. Vi har erfaring med søkefelt laget fra bunn av tidligere. Dette krever masse tid og kunnskap for å utvikle fra bunn av med god responstid. Vi endte derfor opp med Meilisearch som et godt alternativ.

For å visualisere et overblikk over hvordan applikasjonen fungerer i sin helhet, så har vi laget et enkelt flytdiagram nedenfor. Dette flytdiagrammet viser prosessen fra når en først besøker nettstedet og til slutt når en blir med i en lobby.



Figur 24: Flytdiagram - Bli med i lobby

## 5.2 Frontend

Webapplikasjonen er bygget med frontend-rammeverket Angular. Dette er et *TypeScript*-basert rammeverk laget av Google. Det er et av de mest populære frontend-rammeverkene på markedet akkurat nå, sammen med React og Vue. Angular er brukt til å lage «Single-Page Applications» (SPA), som vil si at den ikke laster inn en helt ny side for hver gang klienten trykker, men bytter ut dataen dynamisk.

*TypeScript* er mye likt *JavaScript*, men kan sees på som en oppgradering fra *JavaScript*. *JavaScript* er standarden i alle de største nettlesere i dag. *TypeScript* som vi bruker vil da bli kompilert til *JavaScript* slik at nettleseren kan lese det, og kan da forstå hva den skal gjøre.

CSS & HTML brukes for å bygge nettsiden, der HTML står for strukturen og CSS for det visuelle. Disse to er helt nødvendig for å bygge nettsider, som kan leses av de fleste populære nettleserne vi bruker i dag.

## 5.3 Backend

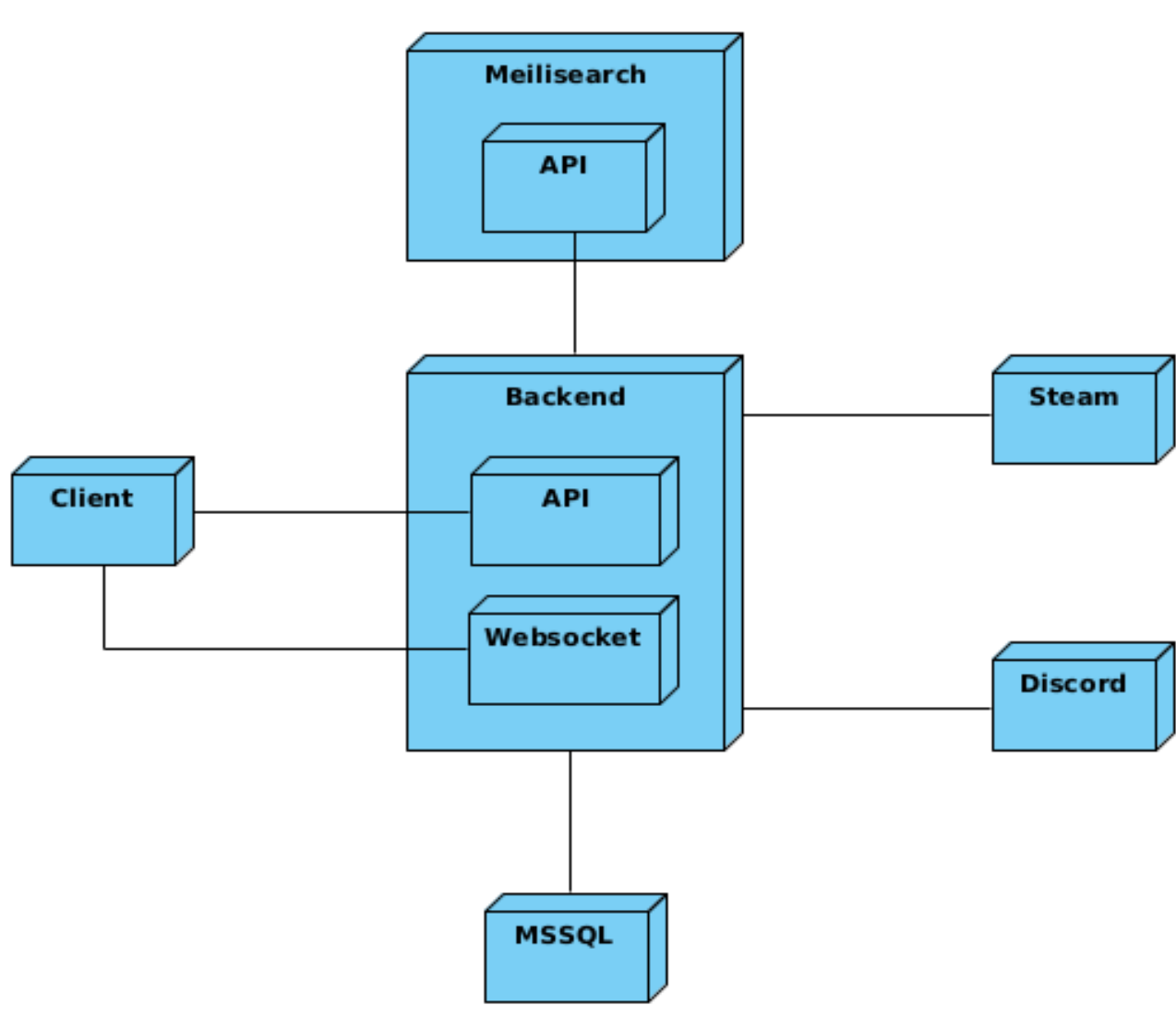
Applikasjonens backend er skrevet i C#. Det er et objekt-orientert programmeringsspråk som ble utviklet av Microsoft. Grunnen til at vi valgte å bruke C# til å utvikle prosjektet sin backend var fordi vi hadde lyst til å ta i bruk .NET, som også er utviklet av Microsoft og er et rammeverk som har stor etterspørsel i arbeidslivet. Det er også mye dokumentasjon og svært mye ressurser å benytte seg av i .NET sitt økosystem. Det er åpen kildekode som betyr at alle kan utvikle og utvide rammeverket ved å lage bibliotek som andre kan ta i bruk.

### 5.3.1 API

Vi tar i bruk en rekke API-er i prosjektet. Først og fremst så har vi vår eget API. Det er her alle forespørslene fra klienten havner. Deretter tar backend og videresender til eventuelle andre API-er. Et av disse er Discord sitt API. Dette gjør det mulig for oss å kunne bruke Discord sine funksjoner som en Discord-bot tilbyr. Her tar vi i bruk et bibliotek for .NET som håndterer forespørslene til API-et. Vi har også laget en egen service som håndterer forespørslene tilknyttet brukeren. Dette brukes til for eksempel å skaffe en bruker sin Discord-ID på vegne av brukeren sin Discord-token.

Steam tilbyr også en rekke API-er, hvor vi benytter to av disse. Den første henter inn informasjon om alle apper som eksisterer hos Steam. Den andre henter inn informasjon om en spesifikk app. Vi sender da en ID som vi får fra første API, og bruker den for å skaffe mer detaljert informasjon fra det andre API-et.

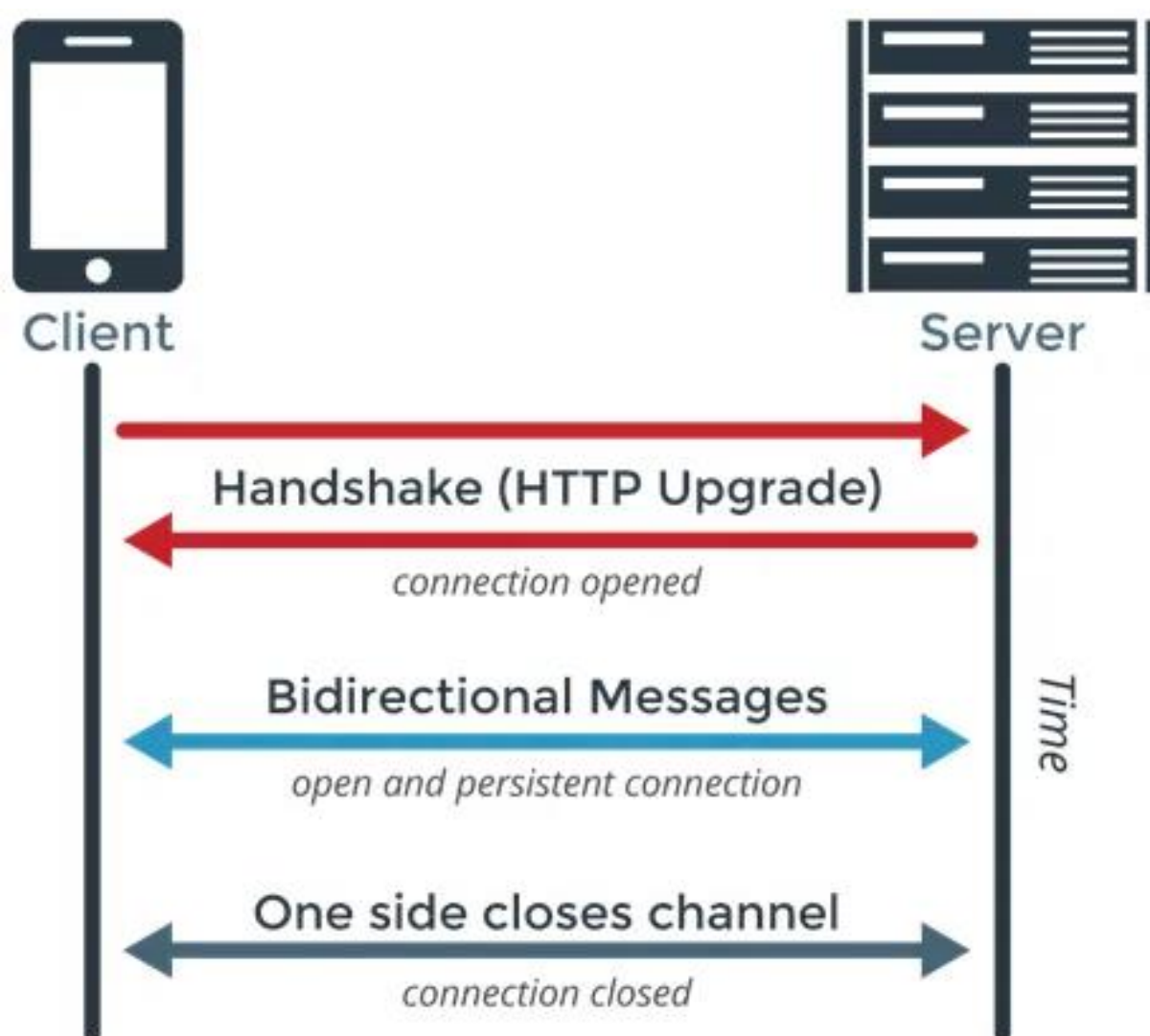
For å kunne integrere en søkemulighet for å søke gjennom mange tusen apper som vi får fra Steam (139.000 for å være eksakt), så trengte vi en måte å gjøre dette effektivt på. Det er her vi tar i bruk Meiliseach. Meiliseach er åpen kildekode som er en kraftig søkemotor vi bruker i prosjektet for at brukeren skal kunne søke på en app eller bruker. Denne aksesseres via et API som Meiliseach oppretter. Nedenfor ligger en veldig enkel modell som viser systemet i sin helhet.



Figur 25: Et enkelt utplasseringsdiagram av applikasjonen.

### 5.3.2 Websocket

For å kunne få mest mulig ut av prosjektet så trengte vi sanntidskommunikasjon mellom klient og tjener. Vi har vært borti websocket tidligere fra andre prosjekt og valgte da å ta i bruk det her også. «Websocket» er en kommunikasjonsprotokoll over en singel TCP kobling. Om vi ser på en HTTP-kobling så snart informasjonen er ferdig levert, så vil HTTP-forbindelsen bli lukket. «Websocket» opprettholder en kontinuerlig kobling som da gir oss muligheten for toveiskommunikasjon. Dette betegnes ofte som «full-duplex», som da sier toveiskommunikasjon over en enkel kanal.



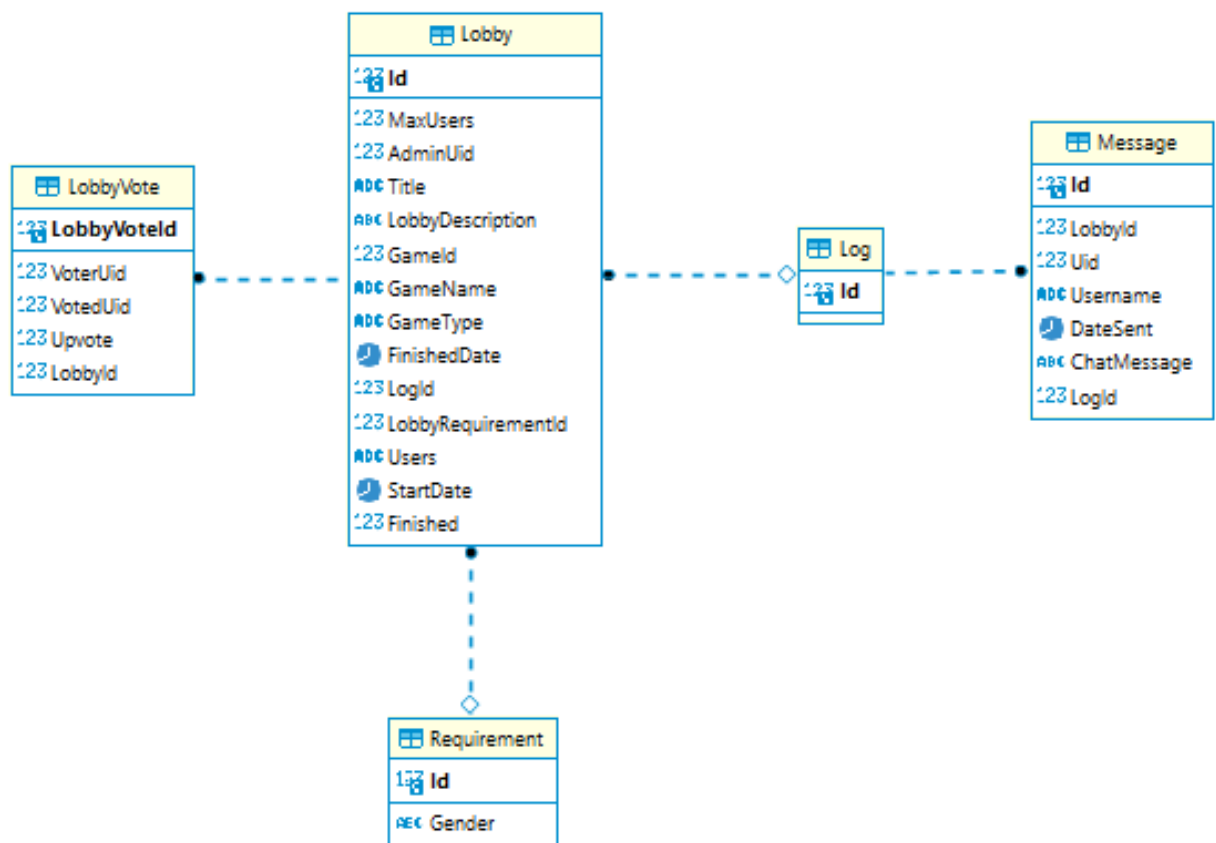
Figur 26: Modellen viser en visuell beskrivelse av Full-dupleks

Vi benytter da SignalR som er et bibliotek fra *.NET* som håndterer opprettelser av «websocket» og andre operasjoner tilknyttet dette. Dette gjorde at vi bare trengte å tenke på utførelsen av funksjonene vi skulle ha. Vi oppretter 2 koblinger, en av disse er for «Lobby» når brukeren først logger inn på nettsiden. Den andre er chatten som er inne i en «Lobby», og kobles bare til når brukeren får tilgang.

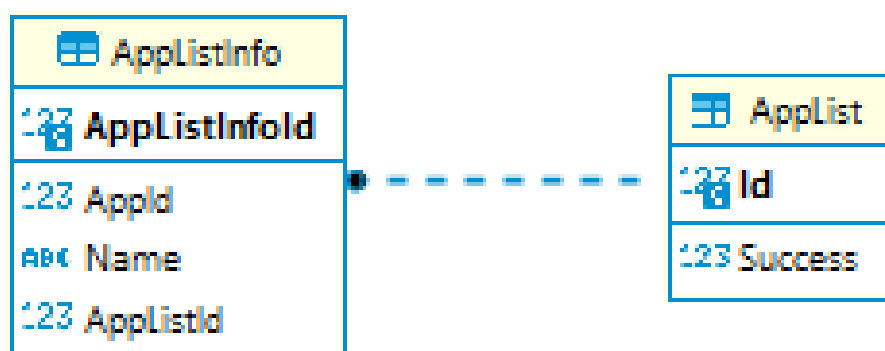
## 5.4 Database

Vi benytter oss av *.NET Entity Framework* som er en moderne objekt-database «mapper» for *.NET*. Vi tenkte en *MSSQL* database ville være best her, siden denne databasetypen er godt integrert med denne modulen. Vi tok først i bruk *MySQL*, men på grunn av hosting hos *Azure* så byttet vi over til *MSSQL* da det var bedre support for denne. Databasen jobber bare direkte med vårt eget API. *Entity Framework* gjør det mulig for oss å heller fokusere på koding, enn SQL (Structured Query Language) delen av databasen. Den støtter LINQ (Language-Integrated Query) spørringer, som kort fortalt gjør om kode til SQL spørringer. For eksempel si at vi vil ha en «Lobby» i databasen. Istedenfor at vi må skrive SQL spørringer som oppretter tabeller, relasjoner og andre attributter, så gjør *Entity Framework* dette for oss. Selvfølgelig så må det blir gjort litt konfigurasjon, men dette gjøres i C# istedenfor direkte mot databasen. Nedenfor kan du se ER diagrammer av databasen.



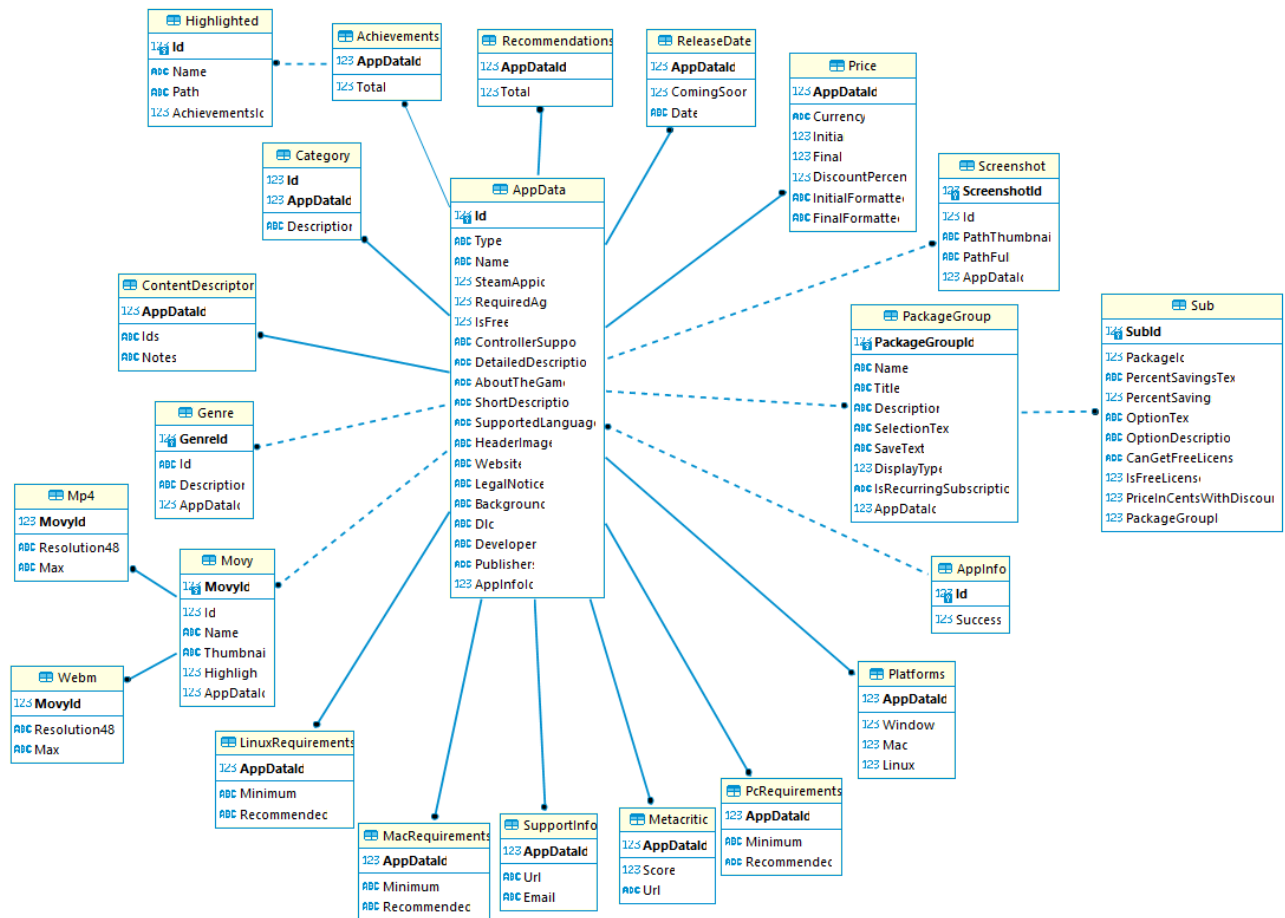


Figur 28: ER Diagram: Lobby



Figur 29: ER Diagram: AppList





Figur 30: ER Diagram: SteamApp

## 5.5 Kodestruktur

Prosjektet følger *Model-View-Controller (MVC)* som kodestruktur. *View* benytter Angular som brukeren kommuniserer med. Presentasjonslaget blir bygget opp med data fra *Controller*. *Controller* utfører nødvendige databehandlinger. Typiske operasjoner her er sortering av data, datamanipulering og beregning av verdier som presentasjonslaget skal benytte. *Controller* benytter seg av *Model*-delen som har alle klassene med tilhørende attributter i forhold til databasen. (Mozilla, 2022)

Grunnen til at vi benyttet *MVC* er at dersom vi gjør endring i enkelte deler, så vil det ikke påvirke de andre delene. Unødvendig kompleksitet er noe man vil unngå i prosjektutvikling. Det er spesielt kode som er avhengig av hverandre som gjør prosjektet mer kompleks og vanskelig å vedlikeholde. Ved bruk av *MVC* så blir hoveddelene (*Model*, *View*, *Controller*) separert fra hverandre. Et eksempel er hvis *View* ikke var separert fra *Model*, så ville det bety at alle endringer i *View* ville ha påvirket *Model* og motsatt vei.

## 5.6 Sikkerhet


Det var svært viktig for oss at sikkerheten på web-applikasjonen overholdt standarden som er satt i dag for web-applikasjoner. Applikasjonen kjører på *HTTPS* og tar i bruk *TLS* (Transport Layer Security), som er den teknologien som har overtatt for *SSL* (Secure Sockets Layer). Dette gjør at alle forespørsler på klientsiden skjer kryptert. For å håndtere bruker-autentisering så benytter vi *JWT* (JSON Web Token). Det er en åpen standard som brukes til å dele sensitiv informasjon mellom to parter, i dette tilfelle en klient og en server. *JWT* benytter seg av en kryptografisk algoritme som sikrer at kravene ikke kan endres etter at “token” er utsendt. *JWT* er satt opp ved hjelp av en pakke som heter *.NET Identity*. Her kunne vi også tatt i bruk en kombinasjon av *JWT* og «session» for å gjøre det sikrere, men vår løsning vil være tilstrekkelig for vårt prosjekt. Vi har også sett på «security headers». Dette er «headers» som beskytter mot forskjellige angrep, som nettsider som regel blir utsatt for. Dette kan være angrep som blant annet XSS, kode injeksjon og klikkjackning.

### 5.6.1 Sikkerhetstester


Vi ønsket å gjøre nettsiden så sikker som mulig etter vår beste evne, derfor tok vi i bruk en del verktøy for å analysere og forbedre sikkerheten.

#### 5.6.1.1 Security Headers

*Security Headers* (Helme, 2022) ble brukt til å analysere HTTP-Respons «headers» til web-applikasjonen. Dette var et veldig bra verktøy da dette ga oss en enkel tilbakemelding på hva vi måtte gjøre for å gjennomføre forbedringer. Under vil du se analysen før vi begynte å implementere disse sikkerhetstiltakene, og etter vi innførte disse tiltakene.

Security Report Summary	
	Site: <a href="https://bop3000.azurewebsites.net/">https://bop3000.azurewebsites.net/</a>
	IP Address: 20.50.2.86
	Report Time: 31 Mar 2022 12:53:13 UTC
	Headers: <span>✖ Strict-Transport-Security</span> <span>✖ Content-Security-Policy</span> <span>✖ X-Frame-Options</span> <span>✖ X-Content-Type-Options</span> <span>✖ Referrer-Policy</span> <span>✖ Permissions-Policy</span>

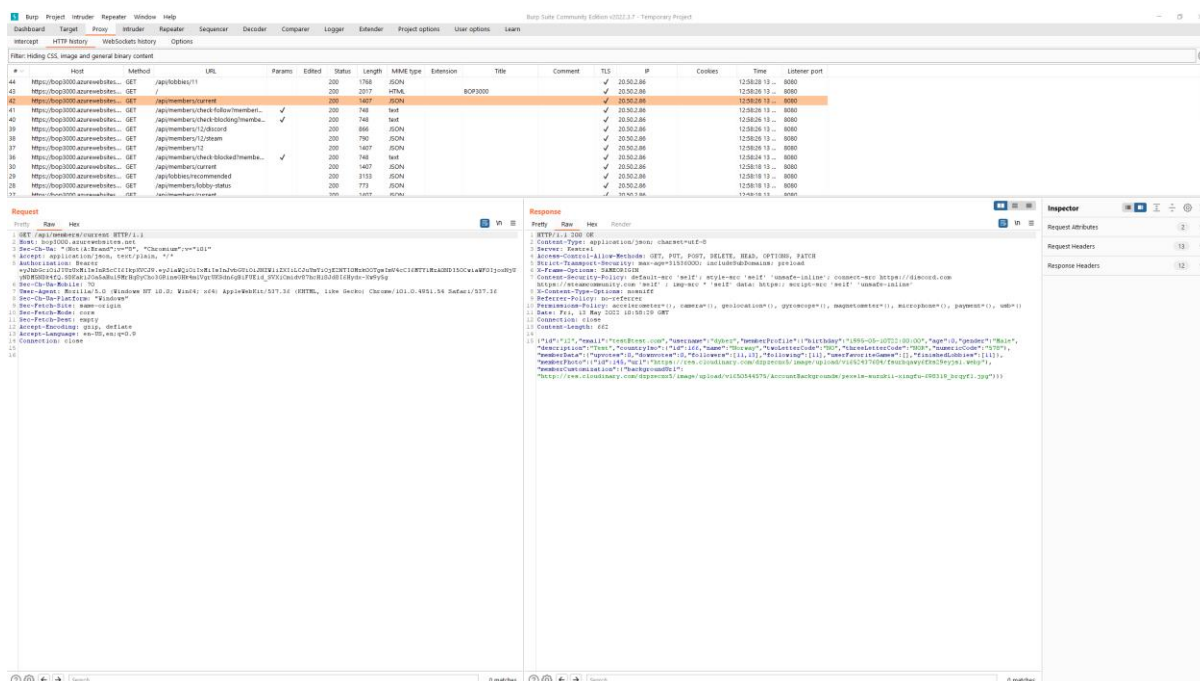
Figur 31: Resultatet før sikkerhetstiltak

Security Report Summary	
	Site: <a href="https://bop3000.azurewebsites.net/">https://bop3000.azurewebsites.net/</a>
	IP Address: 20.50.2.86
	Report Time: 16 Apr 2022 21:02:51 UTC
	Headers: <span>✔ Strict-Transport-Security</span> <span>✔ X-Frame-Options</span> <span>✔ Content-Security-Policy</span> <span>✔ X-Content-Type-Options</span> <span>✔ Referrer-Policy</span> <span>✔ Permissions-Policy</span>
	Warning: Grade capped at A, please see warnings below.

Figur 32: Resultatet etter sikkerhetstiltak

### 5.6.1.2 Burp Suite

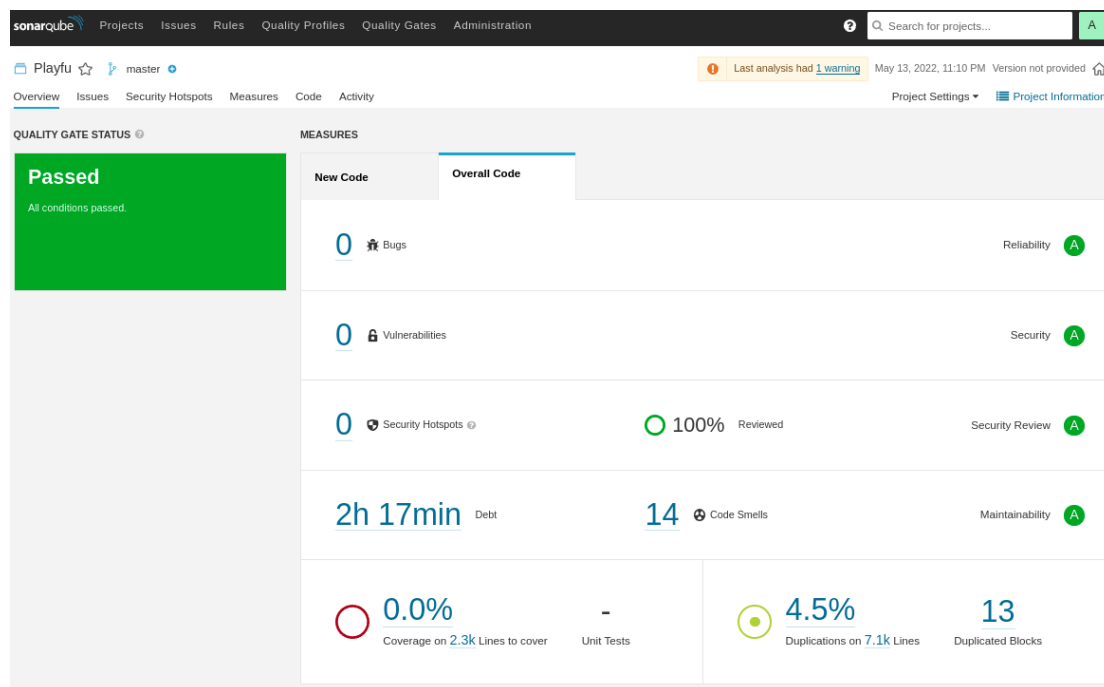
Dette verktøyet ble brukt hovedsakelig med «Proxy». Den fungerer som en nettpoxy-server mellom nettleseren og backend. Slik kunne vi avskjære trafikken mellom disse, og deretter inspisere og endre på trafikken eller dataen som passerer. Vi brukte også «Repeater», som blir brukt til å endre og manipulere spesifikke requests. Med dette kunne vi teste å få oversikt over data vi egentlig ikke skulle få tilgang til. På bildet under ser du hvordan vi kunne inspisere hver enkelt request mellom backend og klient, og se hva slags data den sendte eller mottok.



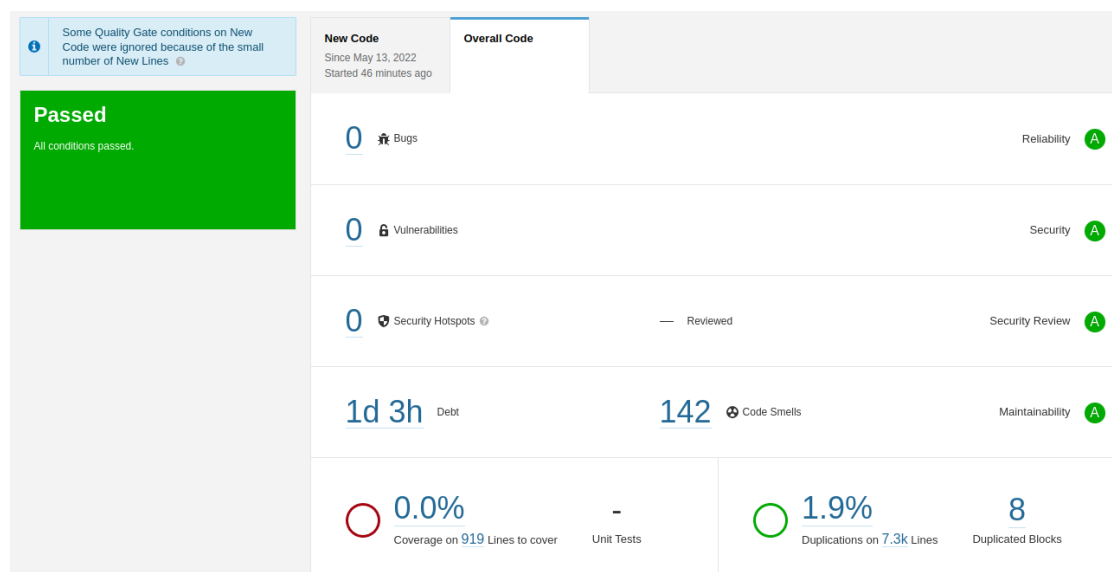
Figur 33: Burp Suite Test

### 5.6.1.3 SonarQube

Vi brukte dette verktøyet for å analysere koden vår for bugs, duplikater og sårbarheter. På bildet nedenfor vil du se hvordan scoren er etter at vi har ordnet opp i bugs, og utgitt vår første og fungerende versjon. Vi scoret veldig godt på det meste, både frontend og backend. Vi hadde noe høye verdier på «code smells», men dette utgjør ingen sikkerhetstrussel eller betydning for ytelsen.



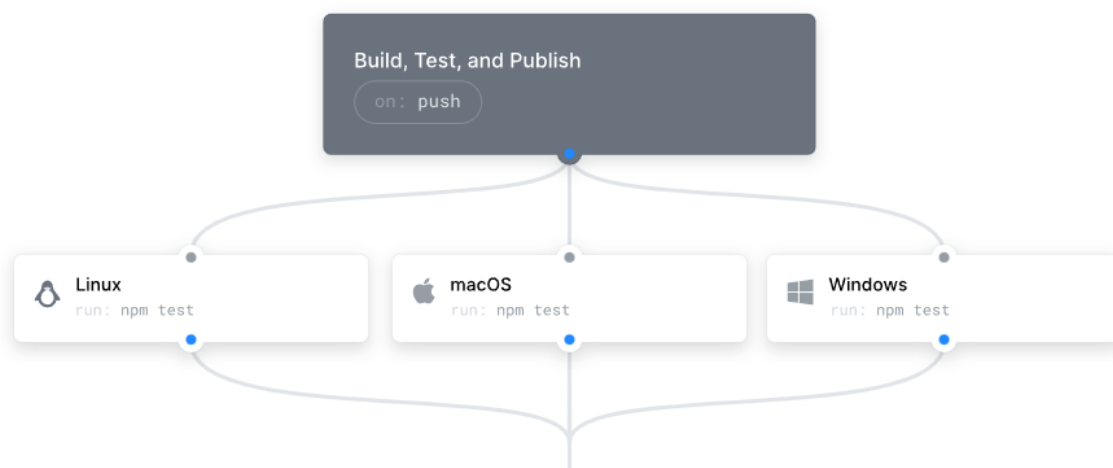
Figur 34: Resultat av backend



Figur 35: Resultat av frontend

## 5.7 Utplassering

Vi har benyttet *Github Actions* for all utplassering. *Github Actions* er en lett måte å automatisere arbeidsflyter for programvare.

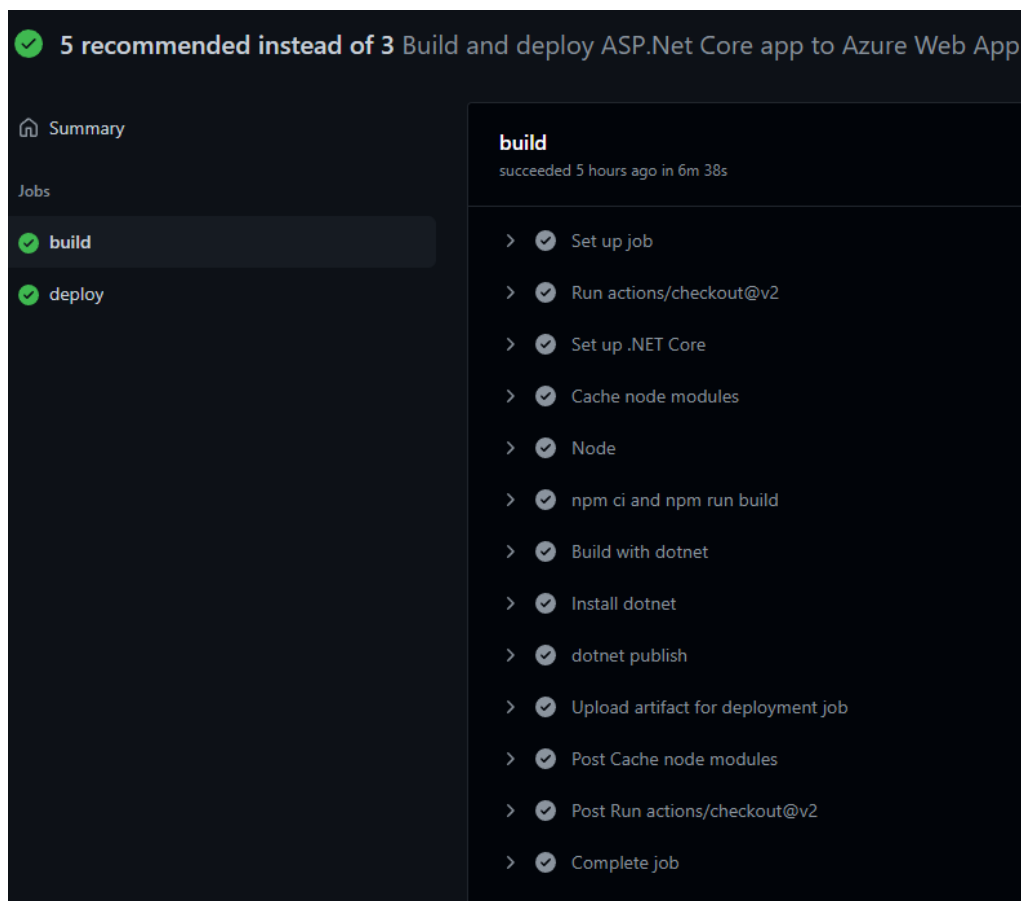


Figur 36: Github Actions Overblikk

Man kan blant annet bygge, teste og utgi applikasjoner direkte i *Github Actions*, noe som gjør det mulig å automatisere utplasseringen av applikasjoner. Dette har spart oss for mye tid som vi kan bruke på utvikling istedenfor.

### 5.7.1 Backend

Utviklingen av prosjektet har pågått i en branch som heter «dev». Når applikasjonen har blitt testet og alt fungerer som det skal så har «dev» blitt «merget» inn i branch «main». Her er det satt opp en «action» som kjøres når det skjer en «commit» på branch «main».



Figur 37: Github Actions Deployment

Her ser man at det er mye som skal gjøres før applikasjonen er klar for utplassering. Først må riktige verktøy skaffes, deretter må applikasjonen bygges ved hjelp av disse verktøyene. Når da alt har bygget ferdig og alt er i orden, så blir applikasjonen lastet opp til Azure som vil starte applikasjonen på nytt og benytte seg av den nye versjonen.

### 5.7.2 API Dokumentasjon

Her har vi benyttet noe som heter Slate (Lord, 2022). Det er et slags hjelpebibliotek som gjør det enkelt å sette opp responsiv API dokumentasjon sammen med *Github Pages*. *Github Pages* brukes til å hoste statiske nettsider hos Github. Slate oppretter alle actions som trengs for utplassering til *Github Pages*. Den benytter 2 branches: «slate» og «gh-pages». Vi skriver dokumentasjonen i branch «slate», mens nettsiden hostes i branch «gh-pages». API-dokumentasjonen skrives i «markdown» som Slate da konverterer til HTML ved hjelp av *Github Actions* og blir da lastet opp til *Github Pages*.

## 6 Brukertesting

### 6.1 Om testplanen

For å oppdage feil og få tilbakemelding på produktet har vi gjennomført brukertester. Det er også gjennomført tester av sikkerheten som ble beskrevet i forrige del (5.7 Sikkerhet).

Vedlegget «Testplan Playfu» er et skjema som beskriver planen for brukertesting.

Skjemaet bruker den samme malen som Helse Vest IKT bruker for sine tester. Her inngår det beskrivelse av testkriteriene våre og hvordan resultatene skal være for at testen blir godkjent.

Videre i denne delen av rapporten beskriver vi fremgangsmåten for brukertesting, spørsmålene, besvarelsene og resultatene fra brukertestene vi har gjennomført.

#### 6.1.1 Fremgangsmåte

Brukerne mottar en lenke til et Google-skjema som forklarer prosedyren for gjennomgangen av siden. Dette innebærer å opprette en bruker og teste Playfus funksjoner. Siden Playfu er en sosial nettside med chat og lobby-funksjonalitet så er vi selv pålogget for å gjøre testingen så fullstending som mulig.

Vi fikk ikke tak i mange testere, totalt testet sju personer siden. Grunnen til at det ikke er så mange er siden testingen er tidkrevende og testerne bør være kjent med Discord. I tillegg var det nødvendig å avtale tidspunkt for testingen siden vi måtte være til stede når dette er en sosial applikasjon. Ulempen med få testere er at enkelte resultat kan bli påvirket mye mer av et individs preferanser, enn med mange testere hvor en får lavere standardavvik som er nærmere gjennomsnittet. I tillegg er testerne personer vi kjenner godt, og det kan medføre at disse relasjonene hindrer at testerne er nøytrale. På den andre siden er oppdagelse av feil høyest prioritert, og de testerne vi har valgt er personer med kompetanse innen IT som vi vet er nøye når det kommer til teknologi.

## 6.2 Brukertesting fase 1.

### 6.2.1 Brukertesting instruks

Forutsetninger:

- Discord bruker

Gjennom testen ønsker vi at du:

Steg 1. Gå til <http://bop3000.azurewebsites.net/>

Steg 2. Opprett en bruker og logg inn.

Steg 3. Koble til din Discord bruker.

Steg 4. Oppdater din konto informasjon.

Steg 5. Sett en ny avatar.

Steg 6. Opprett en Lobby i et spill som ikke finnes i listen over Lobbyer.

Steg 7. Avslutt lobby.

Steg 8. Endre passord.

Steg 9. Finn en "Counter Strike: Global Offensive" lobby og bli med.

Steg 10. Chat med andre i lobbyen.

Steg 11. Spør host om å starte lobbyen (Voice Channel).

Steg 12. Prat med andre i Discord.

Steg 13. Forlat Discord og dra tilbake til nettsiden.

Steg 14. Gi en stemme (opp/ned) på en bruker.



- Steg 15. Gå inn på en bruker.
- Steg 16. Følg brukeren.
- Steg 17. Gå til "Activities".
- Steg 18. Velg en bruker.
- Steg 19. Blokker brukeren.
- Steg 20. Logg ut og fyll ut spørreskjemaet.

Tabell 6: Brukertesting instruksjer

### 6.2.2 Spørsmål i spørreundersøkelsen

Nr.	Spørsmål	Besvarelsesform
1.	What browser are you using?	Alternativer: <ul style="list-style-type: none"> <li>• Firefox</li> <li>• Google Chrome</li> <li>• Microsoft Edge</li> <li>• Apple Safari</li> <li>• Opera</li> <li>• Brave</li> <li>• Annet (tekstsvar)</li> </ul>
2.	On a scale of 1 to 10, how was your first impression?	Lineær skala 1 - 10
3.	Additional comments to first impression:	Tekstbesvarelse
4.	On a scale of 1 to 10, how was your overall impression?	Lineær skala 1 - 10

5.	Additional comments to overall impression:	Tekstbesvarelse
6.	On a scale from 1 to 10, what do you think about the design?	Lineær skala 1 - 10
7.	Additional comments about the design:	Tekstbesvarelse
8.	On a scale of 1 to 10, how user friendly was the page?	Lineær skala 1 - 10
9.	Additional comments about the user friendliness:	Tekstbesvarelse
10.	On a scale of 1 to 10, how flawless was the web application?	Lineær skala 1 - 10
11.	Describe any errors/problems:	Tekstbesvarelse
12.	On a scale of 1 to 10, what do you think of the Lobby feature?	Lineær skala 1 - 10
13.	What were you most pleased with? (Select up to 3 areas)	<p>Alternativer:</p> <ul style="list-style-type: none"> <li>• Design</li> <li>• User-friendly</li> <li>• Lobby Chat</li> <li>• Connection to Discord / Steam</li> <li>• Lobby room</li> <li>• Follow functionality</li> <li>• Up-vote / Down-vote</li> <li>• Activity overview</li> <li>• Profiles</li> </ul>

		<ul style="list-style-type: none"> <li>• FAQ (Frequently Asked Questions)</li> <li>• Settings</li> <li>• Discord integration</li> <li>• Annet...</li> </ul>
14.	What were you least pleased with? (Select up to 3 areas)	<p>Alternativer:</p> <ul style="list-style-type: none"> <li>• Design</li> <li>• User-friendly</li> <li>• Lobby Chat</li> <li>• Connection to Discord / Steam</li> <li>• Lobby room</li> <li>• Follow functionality</li> <li>• Up-vote / Down-vote</li> <li>• Activity overview</li> <li>• Profiles</li> <li>• FAQ (Frequently Asked Questions)</li> <li>• Settings</li> <li>• Discord integration</li> <li>• Annet...</li> </ul>
15.	On a scale of 1 to 10, how relevant would it be for you to use such a page?	Lineær skala 1 - 10
16.	Any comment on your needs and interests as a user:	Tekstbesvarelse

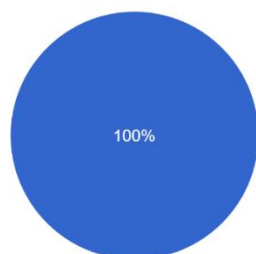
Tabell 7: Spørreundersøkelsens spørsmål

### 6.2.3 Testresultat fase 1.

#### 1. What browser are you using?

What browser are you using?

3 svar

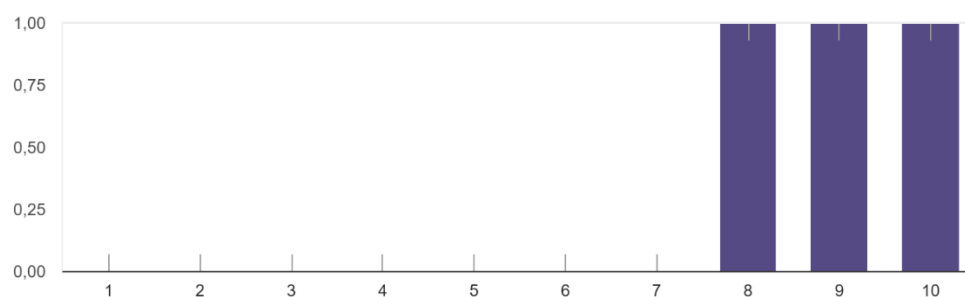


- Firefox
- Google Chrome
- Microsoft Edge
- Apple Safari
- Opera
- Brave

#### 2. How was your first impression?

On a scale of 1 to 10, how was your first impression?

3 svar



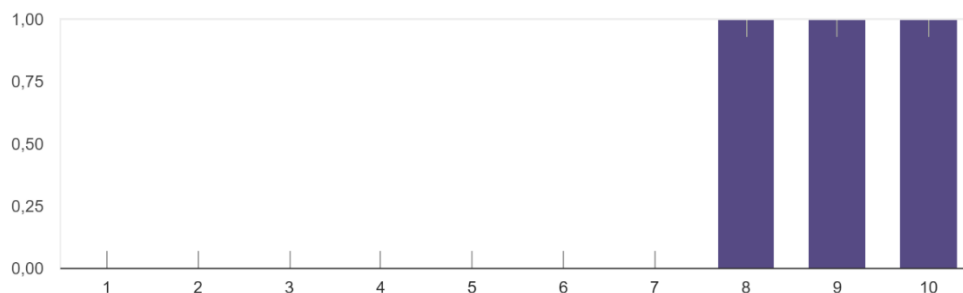
#### 3. Additional comments to first impression:

*Ingen besvarelse.*

#### 4. How was your overall impression?

On a scale of 1 to 10, how was your overall impression?

3 svar



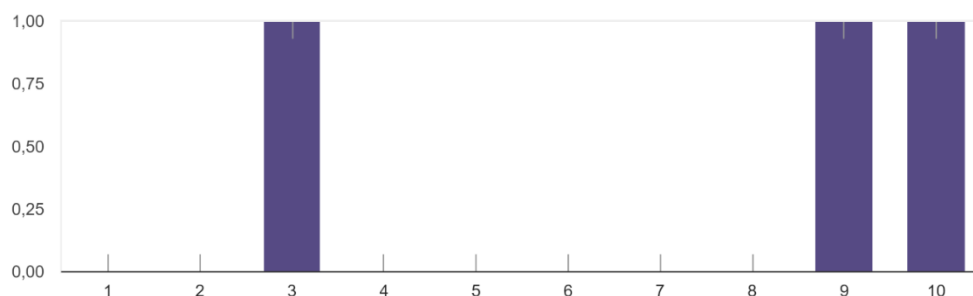
5. Additional comments to overall impression:

*Ingen besvarelse*

6. What do you think about the design?

On a scale from 1 to 10, what do you think about the design?

3 svar



Kommentar angående resultat fra spørsmål 6: For den første testeren vår var det noen feil i spørreskjemaet. Feilen var at 1 stod som «Very good» og 10 stod som «Very bad». Derfor regner vi med denne testeren gav 7 eller 8 i score på dette spørsmålet.

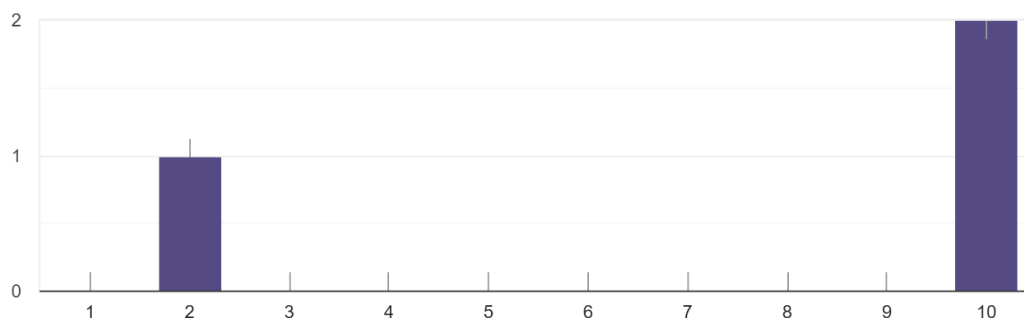
7. Additional comments about the design:

- *hvorfor er det ikke en egen knapp i siden til å linke discord vis det er en av hoved tingene med siden*

8. How user friendly was the page?

On a scale of 1 to 10, how user friendly was the page?

3 svar



Kommentar angående resultat fra spørsmål 8: Igjen her var det feil for betydningen til skalaen. Vi tror denne besvarelsen var gav en score på 9.

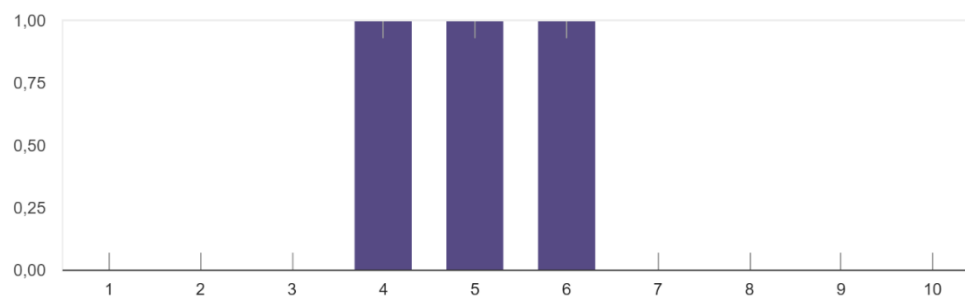
9. Additional comments about the user friendliness:

*Ingen besvarelse.*

10. How flawless was the web application?

On a scale of 1 to 10, how flawless was the web application?

3 svar



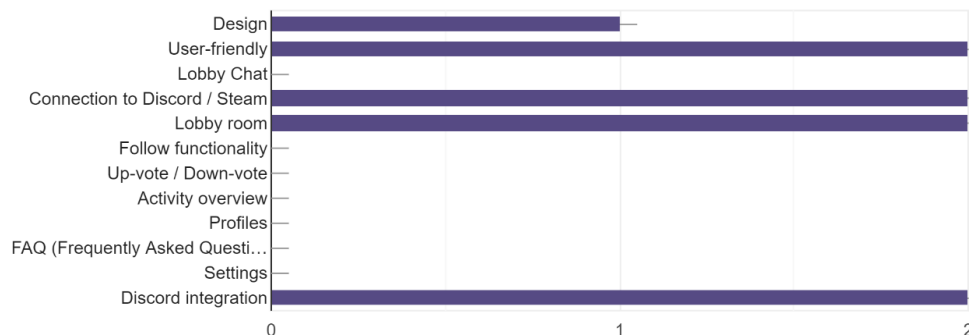
11. Describe any errors/problems:

- chatten fungerer ikke, redirect etter end lobby, og bytte av passord var blocket
- Activities fungerte ikke
- Endring av konto informasjon fungerte ikke

## 12. What were you most pleased with?

What were you most pleased with? (Select up to 3 areas)

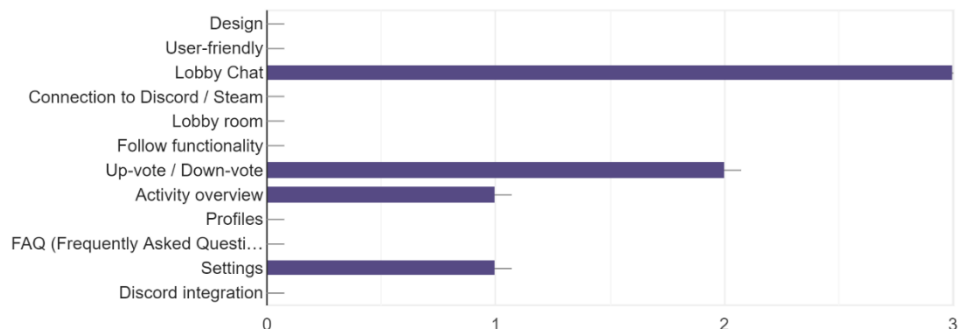
3 svar



## 13. What were you least pleased with?

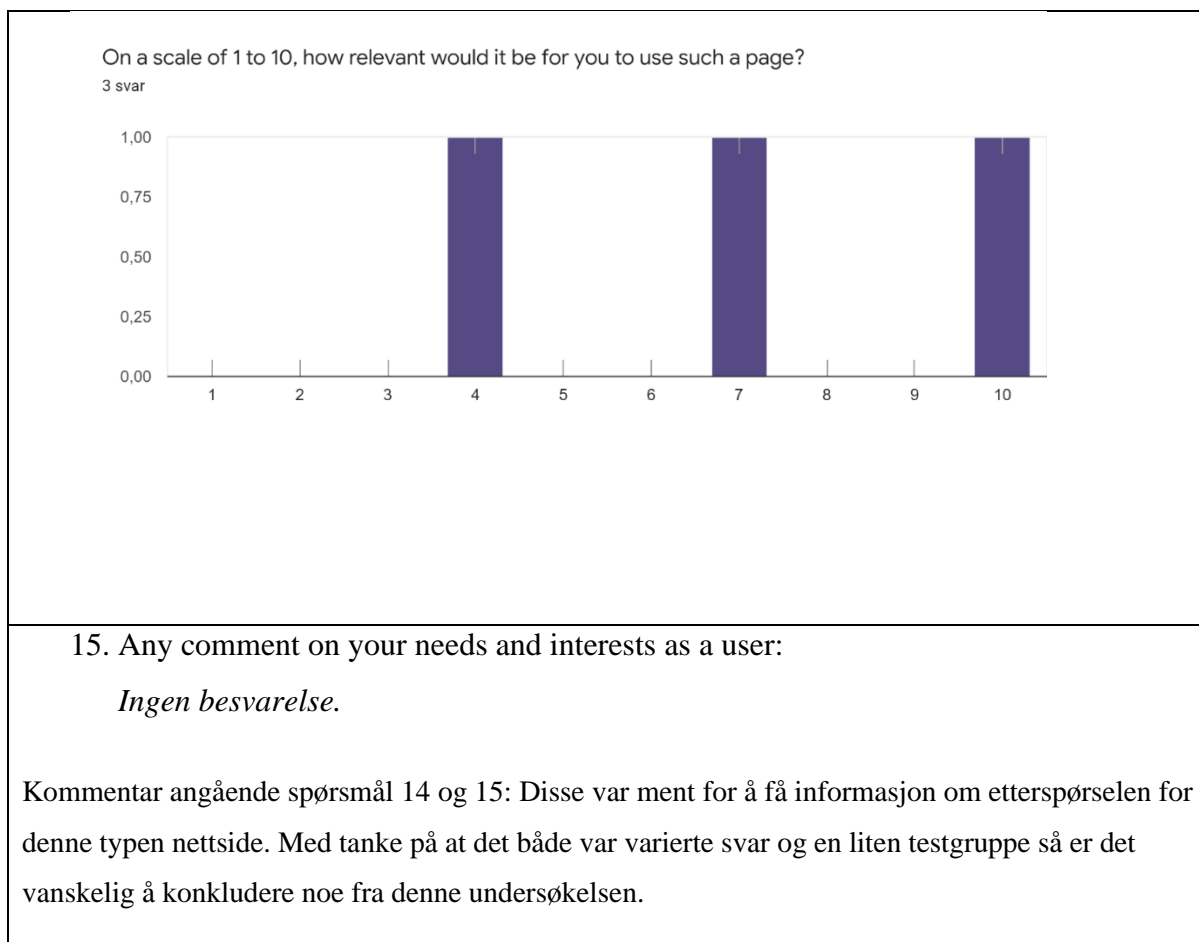
What were you least pleased with? (Select up to 3 areas)

3 svar



Kommentar: Lobby chatten hadde problemer under testing fase 1.

## 14. How relevant would it be for you to use such a page?



Tabell 8: Testresultat fase 1

#### 6.2.4 Resultater brukertesting fase 1.

Gjennomføringen av brukertestene gikk bra. Målet vårt i denne testrunden var hovedsakelig å få andre perspektiv på Playfu og få full oversikt over eventuelle feil. Den største feilen som oppstod var at chatten i Lobbyen ikke fungerer. Denne feilen vurderer vi som en B-feil (alvorlige feil). I tillegg var det C-feil som gjorde at en ikke ble sendt videre automatisk og at en ikke kunne bytte passord.

Kravene i testplanen for en godkjent test er at det ikke er noen utestående A-feil og maks 5 B-feil, så brukertesten er godkjent. I etterkant av denne testen ble feilene rettet opp og vi ønsket enda en kort brukertest til å bekrefte at forbedringer er gjennomført.



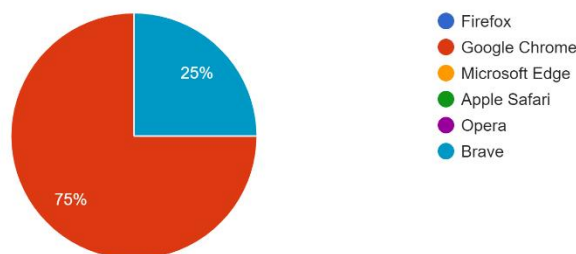
## 6.3 Brukertesting fase 2

Brukertesting fase 2 ble utført 15 og 16 mai. Dette var for å forsikre oss om at feil som er korrigert etter fase 1 fungerer som de skal og at det ikke eksisterer andre feil. Disse testene ble gjennomført med de samme spørsmålene som fase 1 med samhandling med oss og uten hjelp til å navigere siden.

### 6.3.1 Testresultater fase 2

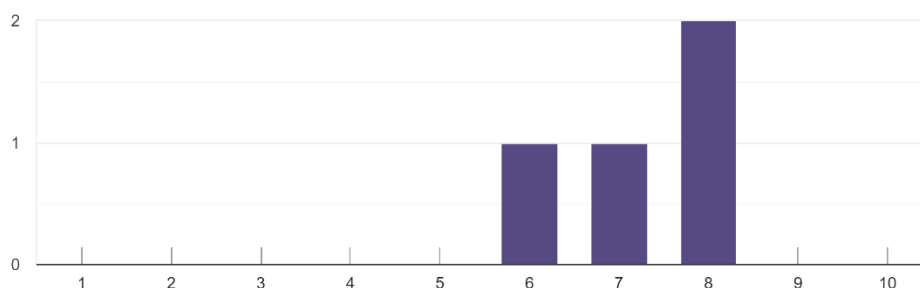
#### 1. What browser are you using?

What browser are you using?  
4 svar



#### 2. How was your first impression?

On a scale of 1 to 10, how was your first impression?  
4 svar



#### 3. Additional comments to first impression:

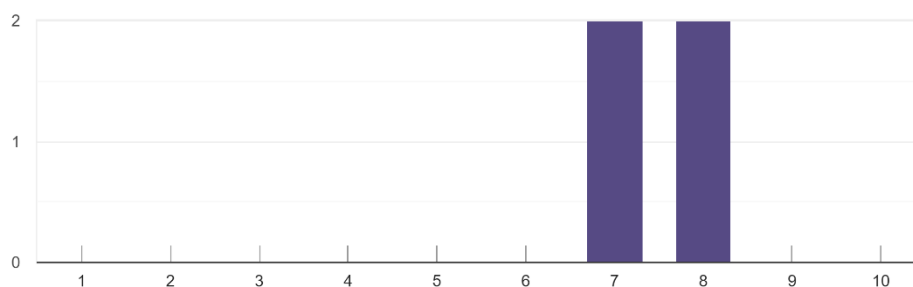
- *"Sign up" funkete ikke i starten. Måtte refreshe for å få knappen til å funke.*
- *Very good website, good idea for lonely people. Sometimes just a bit unclear on where to find certain settings.*

- *alltid fresht å se et forsøk på å samle og evt lage nye samfunn uansett hvilke navn, medium eller interesser det er under eller orientert rundt,*
- *Rent og ryddig*

#### 4. How was your overall impression?

On a scale of 1 to 10, how was your overall impression?

4 svar



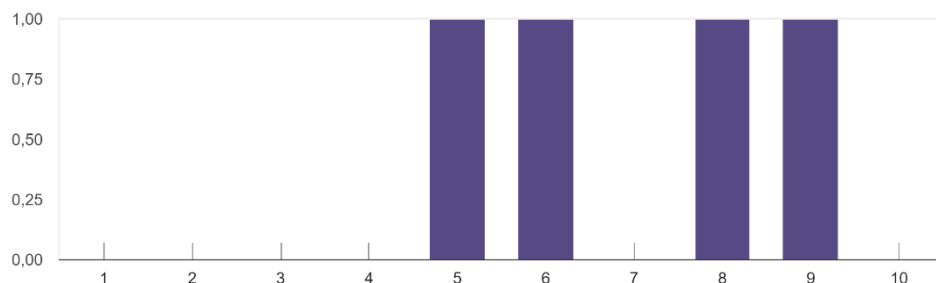
#### 5. Additional comments to overall impression:

- *Ikke for vanskelig og fint med feilmeldinger og suksessmeldinger. Noen bugs som trekker ned.*
- *doesnt have league of legends*

#### 6. What do you think about the design?

On a scale from 1 to 10, what do you think about the design?

4 svar



#### 7. Additional comments about the design:

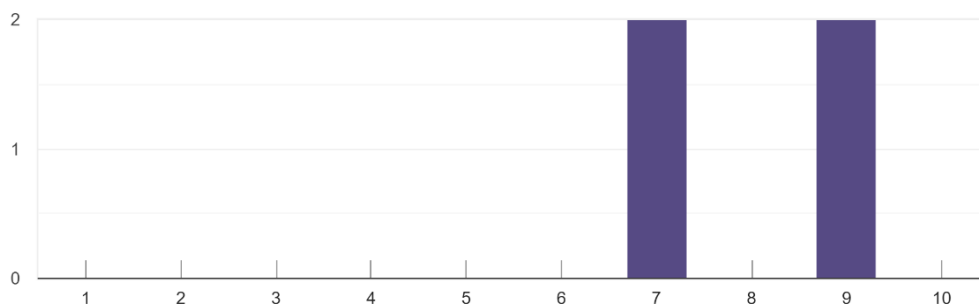
- *Veldig "gamer-aktig" design.*

- *looks clean plus dark so I dont get blinded*
- *De kjenner sluttbrukerne sine da det er slik design som er typisk å se ved diverse lignende tjenester så det å navigere faller ganske automatisk på plass, man føler seg ikke ved ukjent farvann, legg til noen flere "filter" funksjoner så kunne jeg likefort navigert server-browseren til battlefield 4  
<https://battlelog.battlefield.com/bf4/servers/>  
bortsett fra at denne tjennesten er litt mer allsidig orientert iforhold.*
- *Likte godt Spill listen i det du lager lobby*

#### 8. How user friendly was the page?

On a scale of 1 to 10, how user friendly was the page?

4 svar



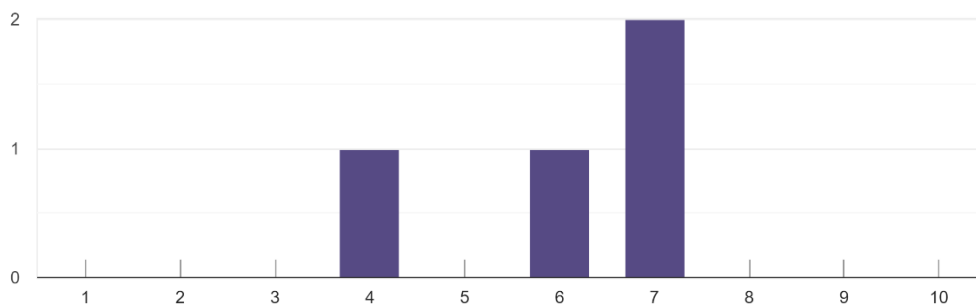
#### 9. Additional comments about the user friendliness:

- *Testet ikke med skjermleser, så vet ikke helt hvor brukervennlig det var på det området, men vil si det var greit å forstå. Det eneste som var litt forvirrende var at man ikke kunne endre på kontoinformasjon under account, men at man måtte inn på innstillinger.*
- *Little unclear at times but takes a second or two to figure out.*

#### 10. How flawless was the web application?

On a scale of 1 to 10, how flawless was the web application?

4 svar



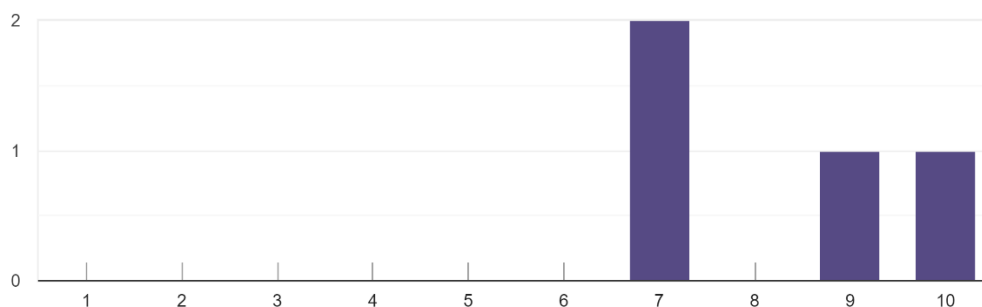
11. Describe any errors/problems:

- *Fikk feilmeldinger etter opprettelse av lobby selv om den ble opprettet. Måtte refreshe for at siden skulle funke igjen. Lå litt random tekst under "sign up"-skjemaet. "Activity"-sida var tom. "Sign up"-knappen funkete ikke i starten.*
- *Overall good, did have to refresh the page a couple of times for it to work.*
- *"aktivitet" panelet var ikke medvillig de første par gangene, men fungerte til slutt,*
- *Kræsja med end lobby*

12. What did you think about the Lobby feature?

On a scale of 1 to 10, what do you think of the Lobby feature?

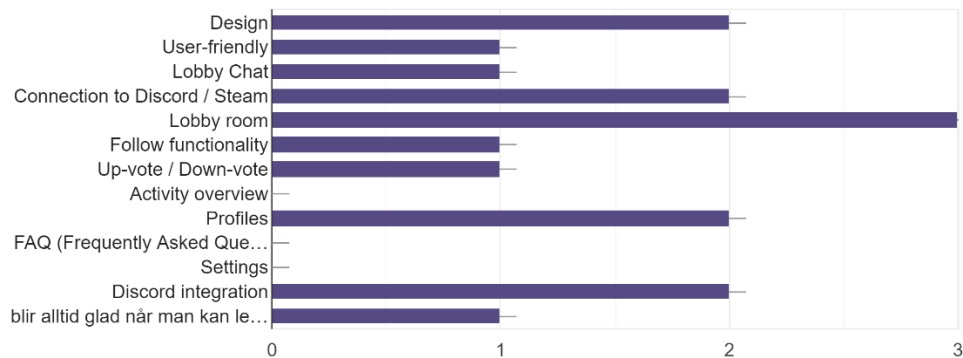
4 svar



13. What were you most pleased with?

What were you most pleased with? (Select up to 3 areas)

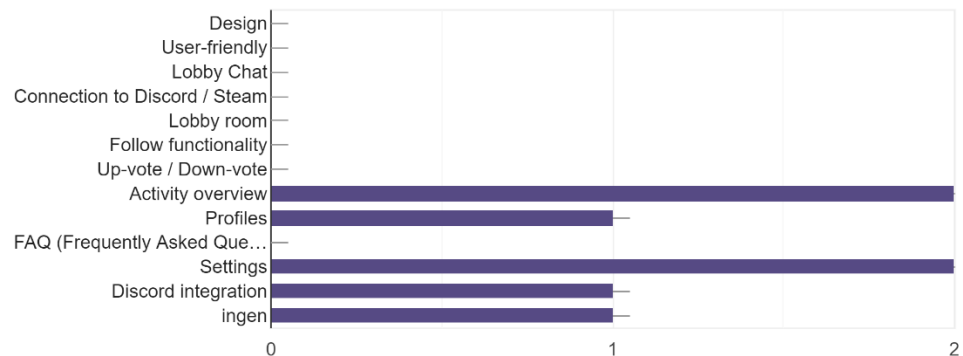
4 svar



#### 14. What were you least pleased with?

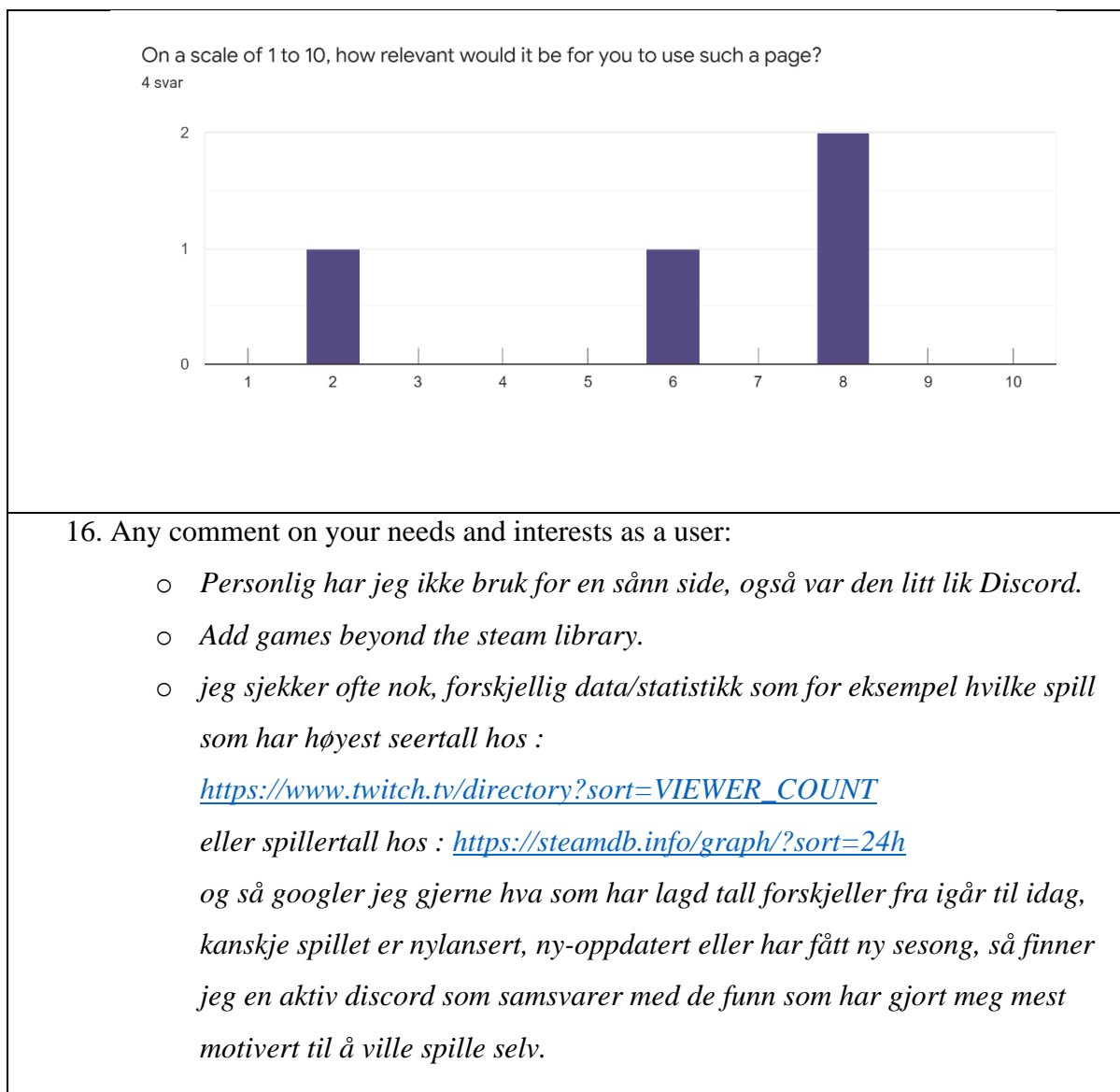
What were you least pleased with? (Select up to 3 areas)

4 svar



Kommentar: Det oppstod nye problemer med Activity-siden.

#### 15. How relevant would it be for you to use such a page?



Tabell 9: Testresultater fase 2

### 6.3.2 Resultater fra brukertesting fase 2

Forventningene for disse testene var at Playfu skulle være så feilfri som mulig, men det viste seg å ikke være tilfellet. Det oppstod og ble oppdaget nye feil under testingen. Mellom noen av testene ble enkelte feil og mangler fikset. Å gjennomføre to test-runder på forskjellige tidspunkt har vært veldig positivt for prosjektet. Med dette har vi oppdaget flere feil og oppnådd å gjøre Playfu så funksjonell som mulig før innlevering av prosjektet.

## 7 Konklusjon

Formålet med prosjektet vårt var å skape en webapplikasjon som fungerte som et møtested for gamere. Dette ga mulighet for registrering av brukere, valg av spill, opprettelse av lobby, lobbychat og integrasjon med Discord og Steam. Dette har vi i stor grad oppnådd med Playfu. Vi fikk inn det meste av det vi ønsket, til tross for enkelte utfordringer.

For å avgjøre hvilke funksjonaliteter vi ønsket å benytte i Playfu lagde vi brukerhistorier. Disse brukerhistoriene dannet grunnlaget for utviklingen av prosjektet. I utviklingen benyttet vi oss av smidig metode ved hjelp av Scrum, Kanban-tavle og sprintbasert utvikling.

Til tross for uforutsette utfordringer med at vi mistet et teammedlem i en periode, så har vi gjennomført et fullverdig prosjekt. Det fraværende teammedlemmet kom sterkt tilbake og tok til seg arbeid igjen, og fortsatte med utviklingen av Playfu.

Gjennom arbeidet med prosjektet har vi tilegnet oss mye nye kunnskaper og ferdigheter, både faglig og personlig. Vi har arbeidet med nye programmeringsspråk, verktøy, API, design, marked og regelverk. I tillegg har vi fått mer erfaring med å jobbe i team og samarbeide. Vi har fordelt roller og oppgaver, og hatt ukentlige møter. På møtene har vi i fellesskap gått gjennom hva hver enkelt har gjort, og delt ut nye oppgaver vi skal arbeide videre med. Oppgavene går ut fra brukerhistoriene som ble satt opp til hver enkelt sprint. Vi har også fått mer erfaring med problemløsning og å håndtere utfordringer. Til tross for uforutsette hendelser som har påvirket gruppen så har alle tilpasset seg og jobbet sammen for å finne nye løsninger.

Vi gjennomførte to runder med testing for å få tilbakemeldinger på Playfu og få oversikt over feil og områder med forbedringspotensiale. I begge testrundene ble det oppdaget flere feil som vi fikk rettet på, og fikk dermed utbedret disse.

Basert på spørreundersøkelsen vår kom det frem at det er varierende interesse av å benytte seg av Playfu. Videre drift ville ikke vært umulig, men krever større innsats for å sikre relevans. For eksempel kunne driften vært mer spesialisert, som å rette seg mot kun Norge eller bestemte spill-arrangementer. Playfu måtte også fungert like bra eller bedre enn andre konkurrenter, og kunne tilby noe unikt.

## 8 Referanser

Angular. (2022, Februar 28). *What is Angular?* Hentet fra Angular.io:

<https://angular.io/guide/what-is-angular>

Datatilsynet. (u.d.). *Datatilsynet*. Hentet Februar 21, 2022 fra Datatilsynet.no:

<https://www.datatilsynet.no/rettigheter-og-plikter/den-registrertes-rettigheter/>

Datatilsynet. (2019, Juli 17). *Hva er en personopplysning?* Hentet Mai 16, 2022 fra

Datatilsynet: <https://www.datatilsynet.no/rettigheter-og-plikter/personopplysninger/>

Datatilsynet. (u.d.). *Datatilsynet*. Hentet Februar 17, 2022 fra Datatilsynet.no:

<https://www.datatilsynet.no/rettigheter-og-plikter/virksomhetenes-plikter/>

Discord. (u.d.). *Discord*. Hentet fra <https://discord.com/developers/docs/intro>

Fusion Media Limited. (u.d.). *Investing*. Hentet Mars 15, 2022 fra

<https://www.investing.com/equities/wegames-financial-summary>

gergerrg. (2022, Januar 03). *gerger*. Hentet fra ergfrg: [www.google.com](http://www.google.com)

Granevang, M. (2020, Juli 31). *Store norske leksikon*. Hentet Mai 14, 2022 fra

<https://snl.no/backend>

Helme, S. (2022, Mars 31). *Scan your site now*. Hentet fra Security Headers:

<https://securityheaders.com/>

Iconify. (u.d.). *Iconify*. Hentet fra <https://docs.iconify.design/design/figma/>

Jøsang, A. (2021). *Informasjonssikkerhet - Teori og praksis*. Universitetsforlaget.

Lord, R. (2022, Mars). *Slatedocs*. Hentet fra Github: <https://github.com/slatedocs/slate>

Microsoft. (2022, Mars 11). *Microsoft*. Hentet Mai 14, 2022 fra

<https://docs.microsoft.com/en-us/dotnet/core/introduction>

Microsoft. (u.d.). *Github Pages*. Hentet Mai 14, 2022 fra <https://pages.github.com/>



Mozilla. (2022, April 30). *Mozilla*. Hentet April 18, 2022 fra

<https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>

Mozilla. (2022, Februar 18). *MVC*. Hentet Mai 14, 2022 fra Mozilla:

<https://developer.mozilla.org/en-US/docs/Glossary/MVC>

no.education-wiki. (u.d.). *no.education-wiki*. Hentet April 19, 2022 fra <https://no.education-wiki.com/1964463-what-is-websocket>

Planning Poker Online. (u.d.). *Planning Poker Online*. Hentet Januar 28, 2022 fra

<https://planningpokeronline.com/>

SonarQube. (u.d.). *SonarQube Documentation*. Hentet fra SonarQube:

<https://docs.sonarqube.org/latest/>

Udemy. (u.d.). *Udemy*. Hentet fra <https://www.udemy.com/>

Unblnd. (2020, Desember 27). *Unblnd*. Hentet April 01, 2022 fra [unblnd.com](https://unblnd.com):

<https://unblnd.com/blog/5-best-apps-to-find-gaming-friends-instantly>

UXPin. (u.d.). *UXPin*. Hentet Mars 8, 2022 fra <https://www.uxpin.com/studio/blog/what-is-a-prototype-a-guide-to-functional-ux/>

Web Dev Simplified Blog. (2021, 5 3). *Web Dev Simplified Blog*. Hentet April 18, 2022 fra

<https://blog.webdevsimplified.com/2021-05/cors/>

Wikipedia. (2022, 04 14). *Wikipedia*. Hentet April 19, 2022 fra

<https://en.wikipedia.org/wiki/WebSocket>

Wikipedia. (2022, Mai 8). *Wikipedia*. Hentet Mai 10, 2022 fra

<https://en.wikipedia.org/wiki/DBeaver>

## 9 Oversikt over tabeller og figurer

### 9.1 Tabeller

Tabell 1: Oversikt over kildekode og viktig dokumentasjon

Tabell 2: Oversikt over ressurser

Tabell 3: Brukerhistorier

Tabell 4: Planlagte utviklingsverktøy

Tabell 5: Nye utviklingsverktøy

Tabell 6: Brukertesting instruksjer

Tabell 7: Spørreundersøkelsens spørsmål

Tabell 8: Testresultat fase 1

Tabell 9: Testresultater fase 2

### 9.2 Figurer

Figur 1: Fremdriftplan for prosjektets faser

Figur 2: Gantt Chart del 1/3

Figur 3: Gantt Chart del 2/3

Figur 4: Gantt Chart del 3/3

Figur 5: Use case diagram

Figur 6: Trello

Figur 7: Flytdiagram av Scrum metoden vi har tatt i bruk

Figur 8: Logo V.1

Figur 10: Oversikt over prototypen i Figma

Figur 11: Skjerm bilde av fremsiden til prototypen

Figur 12: Skjerm bilde av fremsiden til prototypen ved ned-scrolling

**Feil! Fant ikke referansekilden.**

Figur 14: Skjerm bilde av «Game Lobby» i prototypen

Figur 15: Skjerm bilde av Spill-oversikten

Figur 16: Skjerm bilde av «Lobby Rooms» i prototypen

Figur 17: Skjerm bilde av Lobby-oversikten

Figur 18: Skjerm bilde av Chat Lobby i prototypen

Figur 19: Skjerm bilde av Chat Lobby

Figur 20: Kontosiden i prototypen

Figur 21: Kontosiden

Figur 22: Skjerm bilde av profilinnstillinger i prototypen

Figur 23: Skjerm bilde av profilinnstillinger

Figur 24: Flytdiagram - Bli med i lobby

Figur 25: Et enkelt utplasseringsdiagram av applikasjonen.

Figur 26: Modellen viser en visuell beskrivelse av Full-dupleks

Figur 27: ER Diagram: Bruker

Figur 28: ER Diagram: Lobby

Figur 29: ER Diagram: AppList

Figur 30: ER Diagram: SteamApp

Figur 31: Resultatet før sikkerhetstiltak

Figur 32: Resultatet etter sikkerhetstiltak

Figur 33: Burp Suite Test

Figur 34: Resultat av backend

Figur 35: Resultat av frontend

Figur 36: Github Actions Overblikk

Figur 37: Github Actions Deployment