

Prosjektoppgave IN2000

FROLF

Gruppe 7:

Celine Varmann Aamodt

Daniel Kongshaug

Eirin Lishaugen Nyfløt

Nora Skjæveland Utseth

Oscar Berget Dahl

Peder Frivold Skotte ¹

Veiledere:

Sergey Jakobsen

Synne Gåseidnes Thorsen

Emnenavn: Software Engineering med prosjektarbeid

Emnerekode: IN2000

Dato: 20. mai 2022



¹celineva, daniekon, eirinln, norasut, oscarbd, pederfs

Forord

Denne rapporten er svar på prosjektarbeidet i emnet “Software Engineering med prosjektarbeid” ved Universitetet i Oslo. Emnet er et erfaringsbasert emne hvor studentene blir satt sammen i ulike grupper som skal svare på problemstillinger knyttet til software engineering. Rapporten er skrevet av Celine Varmann Aamodt, Daniel Kongshaug, Eirin Lishaugen Nyfløt, Nora Skjæveland Utseth, Oscar Berget Dahl og Peder Frivold Skotte gjennom våren 2022. Formålet med systemutviklingsprosjekt i gruppen har vært å utvikle en applikasjon på Android plattform med programmeringsgrensesnitt fra Meteorologisk institutt, og få trening i å bruke moderne metoder, teknikker og verktøy innen “Software Engineering”. Denne rapporten ønsker å formidle løsning på valgt caseoppgave, med relevant dokumentasjon av hovedaktivitetene fra systemutviklingsprosessen. Dette inkluderer planlegging, kravhåndtering, design, programmering, testing og vedlikehold. Det bør merkes at designprosessen er separert fra prosessdokumentasjonen og er satt som en egen del, navngitt “Designprosess”. Vi ønsker å uttrykke takknemlighet til studentassistentene Sergey Jakobsen og Synne Gåseidnes Thorsen for rådgivning.

Innhold

1 Innledning	1
1.1 Faget og prosjektet	1
1.2 Oppgaven	1
1.3 Gruppen	2
2 Brukerdokumentasjon	3
2.1 Målgruppe og tilgjengelighet	3
2.2 WCAG 2.1 og universell utforming	3
2.3 Design	5
2.4 Applikasjonens funksjonalitet	5
3 Kravspesifikasjon og modellering	8
3.1 Kravspesifikasjon	8
3.2 UML-modellering	10
3.2.1 Use case 1	11
3.2.2 Use case 2	14
3.3 Ikke-funksjonelle krav	16
3.4 Designmønster	17
4 Designprosess	18
4.1 Spørreundersøkelse og dybdeintervju	18
4.1.1 Gjennomføring av spørreundersøkelse	18
4.1.2 Gjennomføring av dybdeintervju	20
4.1.3 Funn fra spørreundersøkelse og dybdeintervju	21
4.2 Designendringer	22
4.3 Brukertesting	23
4.3.1 Gjennomføring av brukertesting	24
4.3.2 Funn fra brukertesting	25
5 Produktdokumentasjon	25
5.1 Teknologi	25
5.2 Valg av API-nivå	26
5.3 Design og arkitektur	26
5.4 HTTP-biblioteker for deserialisering	27
5.5 Egenutviklet frisbeegolfbane-API	28
5.6 API fra Meteorologisk institutt	28
5.7 API fra Google Maps	29
5.8 Utfordringer med API	29
5.9 Kvalitetsegenskaper ved applikasjonen	29
6 Testdokumentasjon	30

6.1	Testprinsipper	30
6.2	Generelt om testing i prosjektet	30
6.3	Enhetstesting	31
6.4	Integrasjonstesting	32
7	Prosessdokumentasjon	33
7.1	Planlegging	33
7.1.1	Utviklingsmetode	33
7.1.2	Verktøy	35
7.1.3	Ideutvikling og produktvisjon	35
7.2	Kravhåndtering	36
7.2.1	Kravspesifikasjon	37
7.3	Design	39
7.4	Programmering	39
7.4.1	Fragmenter	39
7.4.2	Kart	40
7.4.3	Banedata	40
7.4.4	Regelside	41
7.4.5	Værdata	41
7.5	Testing	42
7.6	Vedlikehold og videreutvikling	42
8	Refleksjoner	43
8.1	Personlig refleksjon	43
8.2	Retrospektiv	45
Referanser		47
Vedlegg		i
A Teamavtale		i
B Universell utforming		ii
C Spørreundersøkelse		vi
D Resultater fra kvalitativ spørreundersøkelse		ix
D.1	Generell oversikt	ix
D.2	Andre applikasjoner	x
D.3	Været	xii
D.4	Regler	xiii
D.5	Layout	xiv
D.6	Andre tilleggsfunksjoner	xv

E Intervjuplan	xvii
F Samtykkeskjema	xx
G Affinity Diagram	xxii
H Brukertesting	xxiii
I Kravspesifikasjon	xxiv

1 Innledning

1.1 Faget og prosjektet

Emnet *Software Engineering med prosjektarbeid* er et obligatorisk fag for enkelte bachelorprogram ved Institutt for Informatikk. Målet med faget er å gjøre studentene bedre rustet til fremtidig arbeid innen software engineering. Emnet har spesielt fokus på gruppearbeid og gruppene blir satt sammen av studenter med ulik faglig bakgrunn. Studentene skal utvikle en prosjektidé innenfor emnets rammer, og må lære seg hvordan de kan bidra med sin kompetanse i prosjektarbeidet. I tillegg er det også stort fokus på bruk av moderne metoder, teknikker og verktøy innen software engineering.

Studentene i emnet ble delt opp i grupper på fem til seks personer, og skal sammen utvikle et prosjekt basert på relevant og nøye utvalgt problemstilling. I samarbeid med Meteorologisk institutt skal gruppene utvikle sin egen applikasjon med bruk av Metrologisk institutt sin åpne data og programmeringsgrensesnitt (API) som grunnlag. Det ble gitt seks ulike caseoppgaver hvor gruppen valgte case 6: Åpent case. Dette caset går ut på at studentene kan selv fritt velge tema for applikasjon, så lenge kursledelsen godkjenner forslaget.

1.2 Oppgaven

Gruppen hadde et ønske om å utvikle en original applikasjon med utgangspunkt i en fritidsaktivitet. Basert på egne interesser og globale trender landet gruppen på fritidsaktiviteten frisbeegolf. De siste årene har frisbeegolf, også kalt discgolf, blitt en populær sport både i Norge og på verdensbasis. Bare fra 2019 til 2020 var det en global dobbling av aktive discgolfere (Tverga, 2022). Frisbeegolf ligner vanlig golf, og går ut på å kaste en frisbee i en metallkurv ved så få kast som mulig. Det er en sosial og fysisk sport som krever lite utstyr. I prinsippet krever et spill kun en frisbee. De fleste banene er også gratis å spille på. Sporten er egnet for alle aldre og funksjonsnivåer, og målgruppen for frisbeegolf-applikasjonen er derfor stor. I dag eksisterer det flere alternativer ute på markedet. Felles for disse er at de inneholder svært mange funksjonaliteter. De oppfattes som avanserte og når kun ut til mer erfarte spillere. Gruppen oppdaget et behov for en enklere applikasjon som er mer egnet for nybegynnere.

Løsningen på valgt case er frisbeegolf-applikasjonen *FROLF*. *FROLF* er en applikasjon med kun det nødvendige en spiller trenger under en runde. Dette innebærer en detaljert oversikt over frisbeegolfbaner, kart, regelside og poengkort. For å avgrense oppgaven er kun baner i Osloregionen inkludert. Frisbeegolfbanene er vanligvis plassert i park- og friområder utendørs, og værdata er svært relevant for applikasjonen. Applikasjonen henter derfor ut sanntidsdata for temperatur, nedbør og vindforhold på de ulike banene. Bruker kan enkelt navigere seg frem til ønsket bane for å finne nyttig informasjon. Potensialet for å utvide applikasjonen er stor.

Det ble utviklet en produktvisjon med formål om å veilede arbeidet fremover. *FROLF er en frisbeegolf-applikasjon for nye spillere som mangler en enkel applikasjon til å starte med, som tilbyr rask oversikt over baner, værdata, regler og poengkort. I motsetning til eksisterende frisbeegolf-applikasjoner, tilbyr vårt produkt et enkelt brukergrensesnitt med kun de nødvendige funksjonene for nybegynnere.* Produktvisjonen er en enkel setning som definerer essensen av produktet som skal

utvikles, og er brukt som et grunnlag for mer detaljerte beskrivelser av funksjonalitet (Sommerville, 2019, s. 7).

1.3 Gruppen

Den mangfoldige gruppen består av medlemmer med ulike erfaringer og egenskaper. Gruppen er satt sammen av personer fra ulike studieprogrammer fra Institutt for Informatikk (IFI). Fagfeltene til gruppa er “Informatikk: Design, bruk, interaksjon”, “Informatikk: Digital økonomi og ledelse” og “Informatikk: Programmering og systemarkitektur”. Figur 1 er et gruppebilde av gjengen.



Figur 1: Gruppe 7 tester ut frisbeegolf sammen.

Celine og Peder har bakgrunn i “Design, bruk, interaksjon” og har tilegnet seg kunnskap om hvordan IT-løsninger designes og brukes (UiO, 2021a). Celine sine styrker er blant annet å finne glede i å lage visuelle oversikter og skape struktur. En av hennes svakheter er å prokrastinere i frykt for å mislykkes med arbeidet sitt. Peder liker stress og trives godt når han kan jobbe sammen med andre. En av hans svakheter er at han kan fort bli utålmodig og mislikter ukonkrete arbeidsoppgaver.

Eirin og Nora har bakgrunn i “Digital økonomi og ledelse” som dekker fagfeltene informatikk- og forretningsadministrative fag (UiO, 2021b). Eirin sine styrker er at hun er pålitelig og inkluderende. En svakhet er at hun ønsker å se resultater raskt og kan derfor bli utålmodig ved lite fremdrift. Nora sin styrke er at hun er omgjengelig og mottakelig for andres humør. Hun kan til tider være passiv og anser dette som en av hennes svakheter.

Daniel og Oscar har bakgrunn i “Programmering og systemarkitektur”, og har kunnskaper om oppbygning, utvikling og bruk av datasystemer (UiO, 2021c). Daniel sin styrke er at han jobber effektivt og godt under arbeidsøktene sine. En av hans svakheter er at når han først står fast med

arbeidet sitt kan han bli demotivert. Oscar er gruppens store frisbeegolfspiller og en av hans styrker er at han er løsningsorientert. I likhet med Celine, er en av hans svakheter å prokastinere.

Til tross for at medlemmene kommer fra lignende studieprogram, har fagvalg og tidligere erfaringer gjort at gruppen har en tilnærmet tverrfaglig bredde inn mot oppgaven. I starten av prosjektet utarbeidet gruppen en samarbeidsavtale som fastsetter grunnleggende premisser for relasjonen mellom gruppemedlemmene og danner grunnlag for samarbeidet mellom partene. Samarbeidsavtalen er lagt ved i vedlegg A.

2 Brukerdokumentasjon

I brukerdokumentasjonen er applikasjonens målgruppe og tilgjengelighet diskutert. Tilgjengelighet omfatter hvor brukeren kan få tak i applikasjonen, samt hvilke retningslinjer gruppen har fulgt for å sikre universell utforming. Til slutt inneholder kapittelet en gjennomgang av applikasjonens design og funksjonalitet.

2.1 Målgruppe og tilgjengelighet

Den primære målgruppen er nybegynnere og middels gode frisbeegolfspillere som nettopp har begynt eller ikke spiller på et profesjonelt nivå. Dette forutsetter ingen spesifikk aldersgruppe, men heller at brukeren har tilgang til en smarttelefon, samt mulighet og funksjonsevne til å utøve aktiviteten. Applikasjonen ønsker å dekke så bred målgruppe som mulig. Derfor er enkelhet og brukervennlighet satt i fokus under designprosessen.

Tidlig i prosjektet ble det holdt en spørreundersøkelse hvor deltakerne var alt fra omtrent 12 til over 55 år gamle, med flest deltagere i intervallet 18-25 år. Hadde man vært nødt til å spesifisere en aldersgruppe er det individer i aldersspennet 12-60 år som har vist interesse for temaet. I og med at utviklerne er studenter med begrenset tid, gjenstår det fortsatt mye for å sikre universell brukbarhet. På et vis holdes svaksynte utsiktet utenfor applikasjonens brukergruppe.

Applikasjonen er tilgjengelig for brukere med tilgang til UiO sin GitHub. Kildekoden får en tilgang til ved å søke opp Frisbee på GitHub, eller ved å kontakte en av gruppemedlemmene nevnt i rapporten. Applikasjonen er forbeholdt Android-telefoner. Merk at for å kunne kjøre applikasjonen må API-nøkkelen til Google Maps legges inn i "local.properties" i prosjektfilen. Denne nøkkelen er følgene: "MAPS_API_KEY=AIzaSyCEWb1TU0Y24ng-RXfmZN7ZH1vkU_vUo2E". I *Prosess-dokumentasjonen* vil det bli utdypet grundigere hvordan applikasjonen skal aksesseres i framtiden for at den skal være mer tilgjengelig.

2.2 WCAG 2.1 og universell utforming

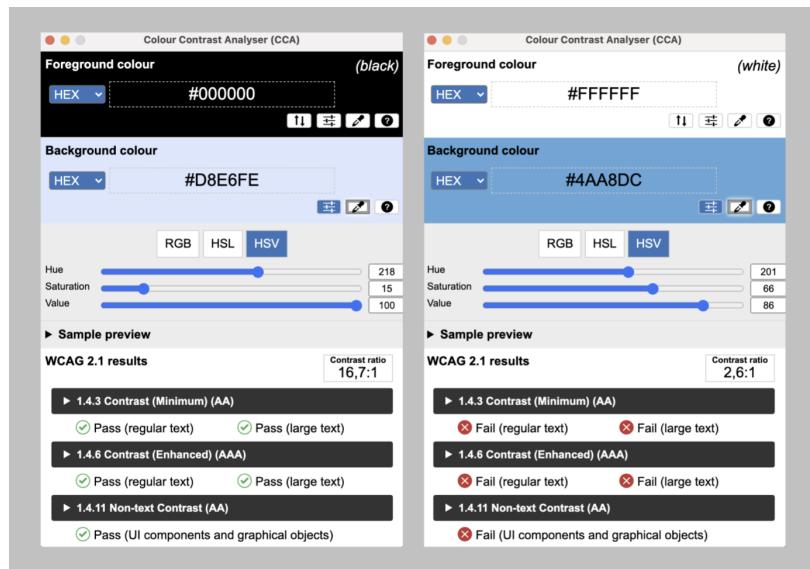
Som utgangspunkt hadde gruppen valgt seg ut 14 WCAG retningslinjer fra 2.1-modellen som skulle oppfylles. Etterhvert som arbeidet med applikasjonens layout utfoldet seg ble det tydelig at selv 14 punkter var en utfordring å sikre. Vedlegg B viser oversikt over graden av de ulike retningslinjene ble dekket. Begrunnelse er utdypet i prosess. Dette innebærer at applikasjonen ikke egner seg for

svaksynte.

Det er ikke enda utarbeidet egne WCAG retningslinjer for applikasjoner, men kravene i regelverket gjelder likevel for nettløsninger. Applikasjoner som trenger internettforbindelse for å kunne brukes etter at de er lastet ned, skal derfor følge kravene til universell utforming av IKT. Hvilke krav som gjelder i praksis, varierer etter applikasjonens funksjonalitet og kompleksitet. Kravene i WCAG gjelder innholdstype, og dersom denne innholdstypen finnes i applikasjonen, skal kravet følges (UUtilsynet, 2022a).

Som utgangspunkt hadde gruppen valgt seg ut 14 WCAG retningslinjer fra 2.1-modellen som skulle oppfylles. Etterhvert som arbeidet med applikasjonens layout utfoldet seg ble det tydelig at selv 14 punkter var en utfordring å sikre. Spesielt syntes gruppen det var vanskelig å gi brukeren mulighet til å endre skriftstørrelse uten at det gikk ut over brukbarheten til applikasjonen. Regelsiden, kartet og banelisten var brukbart leselige i 200 prosent større skriftstørrelse, men baneoversikten og poengkortet mistet all funksjonalitet.

Verktøyet Color Contrast Analyzer (CCA) ble brukt for å måle kontrastforholdene i designet, da dette var anbefalt av Uutilsynet. Figur 2 illustrerer fargetest i CCA av *FROLF*. Forholdet mellom tekst og bakgrunn i applikasjonen er godkjent når det er over 4,5:1 i følge WCAG 1.4.3 (UUtilsynet, 2022b). Etter en gjennomgang med CCA ble det fastslått at applikasjonens kontrastforhold stort sett var godkjent. Der det ikke var godkjent ble det gjort fargeendringer, da gruppen ville prioritere å få på plass de retningslinjene det var tid og kunnskap til å sikre.

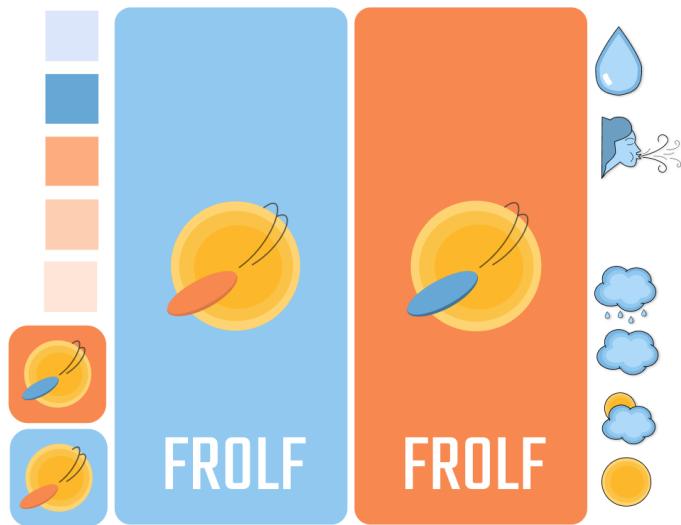


Figur 2: Fargetesting i color contrast analyser (CCA).

Generelt i applikasjonen burde det vært flere muligheter til å tilpasse designet. Dersom man hadde hatt mer tid til å utvikle applikasjonen hadde en av de store prioriteringene vært å lage et design som holder struktur under endring av skriftstørrelse, og som har tekstalternativer til alt av ikke-tekstlig innhold.

2.3 Design

Den etablerte visjonen stod sterkt i fokus ved valg av design og farger. Figur 3 illustrerer symboler og fargevalg brukt i *FROLF*. De to primærfargene som går igjen i designet er ferskenfarget oransje og himmelblå. Ikke bare er disse fargene gode kontrastfarger (Owren, 2019), men de representerer også hver av de to hovedfunksjonalitetene i applikasjonen, nemlig banedata og værdata. Oransjefargen kan minne om en av de mange sterke fargene som ofte brukes på frisbeeplatene. Blåfargen kan minne om nettopp himmelen og været.

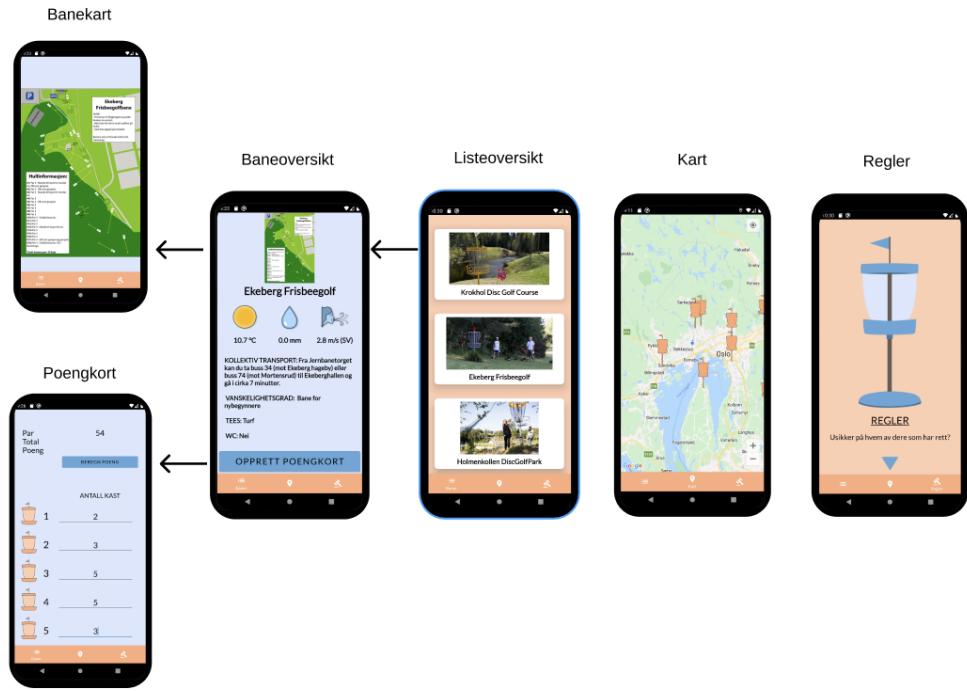


Figur 3: Visuell presentasjon av *FROLF*.

I visjonen står også enkelhet i fokus. Derfor har *FROLF* en stil med enkle linjer og mindre detaljer. Navigasjonsbaren benytter seg av tre symboler fra Android sitt bibliotek. Til resten av applikasjonen er det utarbeidet egne symboler og illustrasjoner. Dette er med på å danne en helhetlig og oversiktlig følelse. I tillegg til dette ligger det meste av informasjon sortert bak overskrifter og forsider. På denne måten unngår man at brukeren blir presentert med mengder av informasjon, før de velger å oppsøke den selv.

2.4 Applikasjonens funksjonalitet

Under bruk av *FROLF* kan brukeren navigere seg gjennom flere sider med ulike funksjonaliteter. Figur 4 illustrerer applikasjonens struktur og hvordan de ulike sidene er koblet sammen.



Figur 4: Skjermbilder av applikasjonens sider. Pilene viser sammenhengen mellom sidene.

En bruker som åpner applikasjonen for første gang vil bli møtt av en introskjerm, med en *FROLF*-logo i applikasjonens hovedfarger. Når alt er lastet inn, vil en forespørsel vises på midten av skjermen. Her vil brukeren bli bedt om å gi tilgang til sin posisjon. Dersom brukeren godtar, vil posisjonen vises på kartet gjennom den klassiske “blue dot”-markøren til Google Maps. Kartet vil også sentreres rundt dette punktet, og frisbeegolfbaner i området vil være markert med egne kurv-markører laget kun for denne applikasjonen. Dersom brukeren ikke godtar, kan applikasjonen fortsatt benyttes, men med begrenset funksjonalitet.

Ved behov for adresse eller veibeskrivelse til en bestemt bane kan bruker trykke på den respektive banemarkøren. Da vil kartet sentreres rundt dette punktet, og to ikoner vil vises nede i høyre hjørne. Ved å trykke på “veibeskrivelse”-ikonet (blå diamant med pil) vil bruker navigeres direkte til veibeskrivelse i Google Maps. Dersom det er ønskelig å kun åpne lokasjonen i Google Maps, kan det trykkes på Google Maps-ikonet ved siden av.

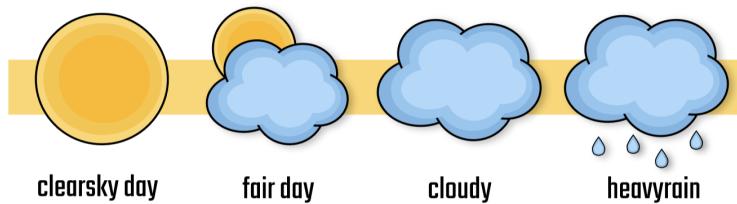
Nederst i applikasjonen vil det til enhver tid være en fersken-farget navigasjonsbar med hvit skrift. Denne navigasjonsbaren viser tre ikoner; et liste-ikon for “Baner”, et markør-ikon for “Kart” og et hammer-ikon for “Regler”. Det ikonet med skrift under vil være den siden bruker befinner seg på.

Dersom bruker har lyst til å undersøke forholdene på de ulike banene, trykkes det på liste-ikonet. Klikket fører brukeren til en liste av baner, hvor hver bane er representert med navn og bilde. Bruker kan finne ønsket bane ved å bla gjennom listen, og trykke på ønsket bane fører bruker til en side med utdypende informasjon om banen. Det er på denne siden av applikasjonen at banedata møter værdata. Ved valgt bane vil brukeren få opp en baneoversikt. Øverst i oversikten ligger et banekart, som brukeren kan se nærmere på ved å trykke på bildet. Da vil det fylle hele skjermen. Under

banekartet finner man navnet på banen.

All værdata ligger under banenavnet og er representert ved tre ulike ikoner. Helt til høyre finner man vindguden Aiolos (SNL, 2019), og under dette ikonet oppdateres informasjon om vindhastighet og retning. Vinden blir målt i meter i sekundet, og retningen er representert gjennom henholdsvis N for nord, S for sør, Ø for øst og V for vest. Til venstre for dette ligger informasjon om nedbør. Dette er signalisert gjennom et vanndråpe-ikon, hvor nedbør målt i millimeter oppdateres under ikonet.

Helt til venstre finner man den dynamiske delen av baneoversikten. Under et værsymbol vil temperatur stå oppført i Celsius. I tillegg til dette vil værsymbolet endre seg ut i fra skydekket på banen. I *FROLF* er det utarbeidet fire ulike symboler for skydekke; clearsky day for når det er skyfritt, fair day for de dagene med litt skyer, cloudy for når det er overskyet, og heavyrain når det regner. Værsymbolene er illustrert i figur 5. Under værdataen ligger en enkel oversikt over kollektiv transport, vanskelighetsgrad, tees og wc.



Figur 5: *FROLF* sine egne værsymboler

Helt nederst i baneoversikten har brukeren mulighet til å opprette et poengkort via en knapp. Trykker brukeren på knappen vil det opprettes en oversikt med atten hull som utgangspunkt. Hvert hull er representert med en frisbeegolfkurv og hullnummer ved siden av. Bruker kan fylle inn hvor mange kast som ble brukt på å treffen kurven, i et skriffelt til høyre for tilhørende hullnummer.

Helt øverst på poengkortet vil det til en hver tid være en poengberegnere. Øverst i poengberegneren vil totalpar for denne banen ligge, til høyre for "par". For å regne brukerens poengsum er man nødt til å legge sammen antall kast, og deretter sjekke hvor mange kast dette er over eller under par. Poengsummen vil altså være et positivt eller negativt tall som representerer antall kast brukt over (positivt tall) eller under (negativt tall) par. Målet med spillet er å ha et så lavt tall som mulig. Når "beregn poeng"-knappen trykkes på, vil *FROLF* gjøre denne beregningen automatisk, og både antall kast og den foreløpige poengsummen vil vises til høyre for "total" og "poeng". På denne måten kan brukeren enkelt se hvordan de ligger an, uten å måtte regne ut selv, eller søke opp par på en annen måte.

Sist men ikke minst tilbyr *FROLF* en egen regelside, som raskt kan aksesseres gjennom navigasjonsbaren. Hvis brukeren trykker på hammerikonet helt til høyre, vises en enkel forside bestående av en blå frisbeegolfkurv på en oransje bakgrunn. En liten blå pil nederst på skjermen vil indikere at brukeren kan bla nedover for å komme til regeloversikten. Som utgangspunkt vil applikasjonen vise de enkleste reglene først. Dersom brukeren har spørsmål om noe spesifikt, eller bare vil lese seg opp

på andre regler, kan det velges mellom flere regelkategorier i en spinner. Regelsiden kan aksesseres selv om brukeren har begynt å føre poeng på banen. Navigerer bruker tilbake gjennom liste-ikonet vil bruker komme direkte tilbake til poengkortet.

3 Kravspesifikasjon og modellering

Modellbasert systemutvikling innebærer objektorientert analyse og design. Analyse er knyttet til å identifisere hva systemet skal gjøre og bestemme krav til funksjonalitet. Dette var en viktig aktivitet i prosjektet. Kravene utgjør basen for design og implementasjon av systemet. Grundig gjennomført kravhåndtering er spesielt viktig i systemutviklingsprosjekter. Det er spesielt kostbart å rette feil etter systemleveranse (Maus, 2010). I design-delen bestemmes hvordan kravene skal implementeres i systemets helhet. Modellering kan gi en visuell representasjon, og gi en ide på egnet designmønster for videre implementasjon av kode og testing

3.1 Kravspesifikasjon

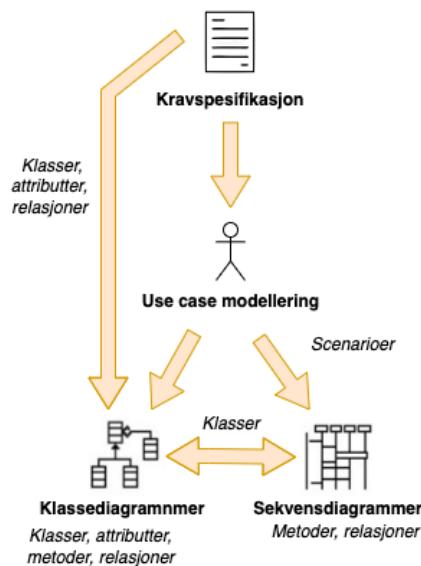
Kravhåndtering var en sentral del i utviklingsprosessen av *FROLF*. Under kravhåndteringsprosessen ble de funksjonelle kravene bestemt. Denne prosessen innebar å samle inn, analysere, validere, samt endre og organisere forslag. Idémyldring innad i gruppen, ønsker fra brukerundersøkelsen og dybdeintervju ga et bredt grunnlag for hva som kunne inkluderes. Basert på svar fra brukerundersøkelsen ble kravene videre analysert og validert. Det var nødvendig å tilpasse forslagene etter hva som var teknisk mulig å få til. Til slutt ble kravene organisert etter standard format ved å gruppere de etter må, kan og bør. Må-kravene er absolutte. Bør-kravene er ønsket funksjonalitet, som kan prioriteres etter at må-kravene er dekket. Kan-kravene har lavest prioritering og kan implementeres etter at både må og bør-kravene er implementert. Tabell 1 viser resultatet av kravhåndteringsprosessen. Nummereringen er tilfeldig, og implementasjonen foregikk etter hva som egnet seg best under prosessen.

Tabell 1: Følgende funksjonell krav ble tatt hensyn til under utviklingsprosessen av applikasjonen. Kravene beskriver hva systemet skal kunne gjøre. Prioritert rekkefølge fra høy-middels-lav er henholdsvis må-bør-kan.

Prioritet	Nr	Kravspesifikasjon
Må		
	1.0	Oversikt over banene (recycler)
	1.1	Liste over alle baner i Oslo, med tilhørende banekart
	1.2	Vise lokal værmelding for valgt område (temperatur, vindstryke, vindretning, nedbør og skydekke)
	1.3	Vise en beskrivelse for hver bane (Kollektiv, vanskelighetsgrad, tees og WC)
	1.4	Regelside med de viktigste reglene
	1.5	Føre poeng/kast til de ulike hullene
	1.6	Følge utvalgte WCAG-prinsipper
	1.7	Introside - Logo når bruker åpner applikasjon
	1.8	Scoreboard som tillater å bytte fragment uten at dataen forsvinner
Bør		
	2.0	Vise brukerens posisjon i kartet
	2.1	Nullstill zoom inn/ut på kart
	2.2	Utvidet regelside med kategorier (f.eks. folkeskikk og mando)
	2.3	Lagre poeng for hver runde
	2.4	Introside til applikasjonen (når man åpner applikasjonen første gang)
	2.5	Vise vanskelighetsgrad på de ulike banene
	2.6	Vise rating på de ulike banene
	2.7	Vise spilletid på hver bane
	2.8	Vise soloppgang og solnedgang
	2.9	Nedlastningsskjerm med snurrende frisbee
Kan		
	3.0	Tilby dark mode og light mode
	3.1	Tilby søk i fritekst blant baner
	3.2	Opprette profil
	3.3	Scoreboard med plass til flere personer
	3.4	Lagre favorittbane
	3.5	Slette favorittbane
	3.6	Instillinger-side (eks. clear history, dark mode, profilinstillinger)
	3.7	Legge igjen egen review på banen
	3.8	Vise pågang på banen
	3.9	Vise parkeringsinformasjon
	3.10	Vise om banen har ly for vinden
	3.11	Sortere listen med baner etter avstand, pågang, rating, kollektiv, vær
	3.12	Legge til flere baner i Norge
	3.13	Vise avstand til banene fra "min posisjon"
	3.14	Tilby satellitt/terreng-kart
	3.15	Måle avstand på kast
	3.16	Tilby banekart i Google Earth (polygon)
	3.17	Legge til en funksjon som viser UV-varsel for en gitt lokasjon og bane
	3.18	Vise nærmeste kiosk/badevann i forhold til banen
	3.19	Anbefale pakkeliste basert på værforhold
	3.20	Tilby videoer med teknikker og "tips og triks"

3.2 UML-modellering

Under prosjektet var det viktig at alle i gruppen hadde en felles forståelse for arkitekturen av systemet. Det var derfor hensiktsmessig å lage abstrakte modeller av applikasjonen for å illustrere ulike bruker- og systemperspektiv. Unified Modeling Language (UML) diagram er en industristandard for objekt-orientert modellering, og kan benyttes for å uttrykke krav på en kort og konsis måte. Ved å modellere flere ulike UML-diagram kan ulike aspekter ved applikasjonen illustreres (Sundaramoorthy, 2022, s. 2). De viktigste funksjonelle kravene ble kartlagt i to brukstilfeller (use case) med tilhørende brukerhistorie (user story), navngitt use case 1 og use case 2. For å få en logisk framstilling av applikasjonen, ble det modellert både et sekvensdiagram og et klassediagram for hvert use case. Hendelsesflyten fra et use case kan enkelt detaljeres ut i et sekvensdiagram, og videre utvides til et klassediagram med systemklasser (Lindsjørn, 2022b). Figur 6 viser skjematiske fremstilling av modellingsprosessen.

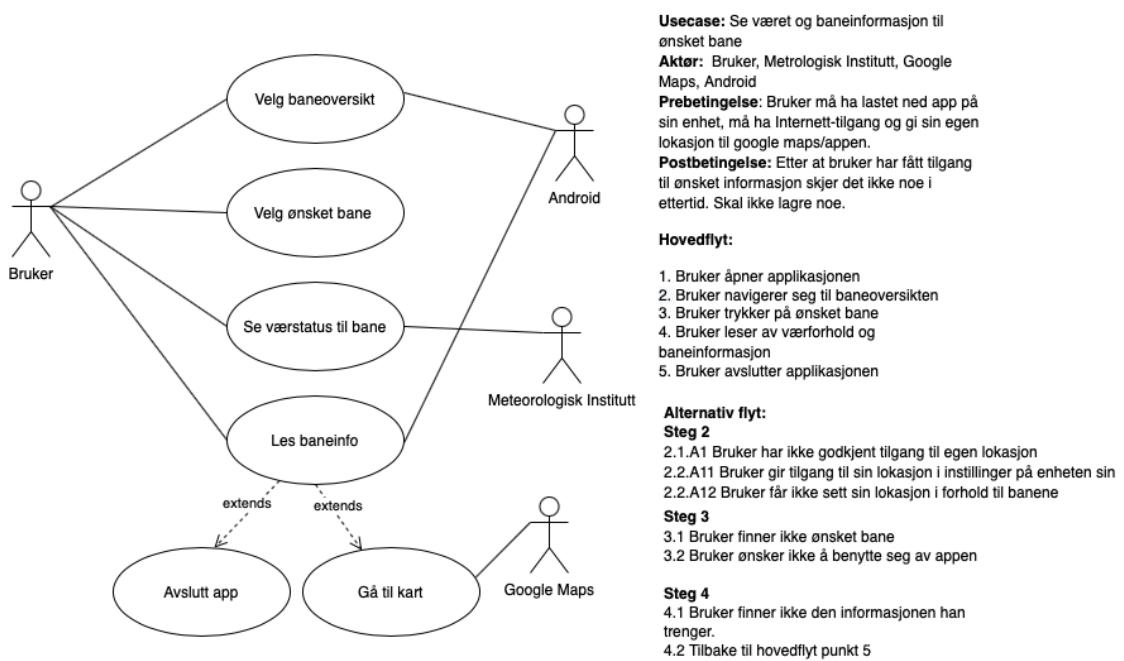


Figur 6: Use case i design (Lindsjørn, 2022b).

Overordnet viser et use case systemets funksjonalitet, og samspillet mellom systemet og omgivelsene. Hensikten med å modellere use case av systemet er for å illustrere kontekst der systemet tas i bruk, og hva brukeren faktisk har behov for. Et sekvensdiagram viser samspill mellom system og omgivelser, og mellom de forskjellige delene av systemet. Stegene (sekvensene) i systemet illustreres som meldinger som kan sendes mellom objektene ved å kalles på objektenes metoder, hvor interaksjonen mellom objektene er tegnet i rekkefølge fra topp til bunn av figuren. Modellering av sekvensdiagram er nyttig for å se hva som skal skje ved kjøretid, og viser den dynamiske oppførselen til applikasjonen. Et klassediagram viser objektklassene i systemet og assosiasjonene mellom disse klassene. Modellen er nyttig for å kartlegge hvilke objekter som bør tas hensyn til under utviklingen, og gir en visuell illustrasjon på hvordan relasjonene er mellom objektene (Papajorgji & Pardalos, 2014, s. 53–57), (Lindsjørn, 2022a). UML-modellene for systemet ble løpende modifisert under utviklingsprosessen. De presenterte modellene presenterer de siste versjonene av systemet.

3.2.1 Use case 1

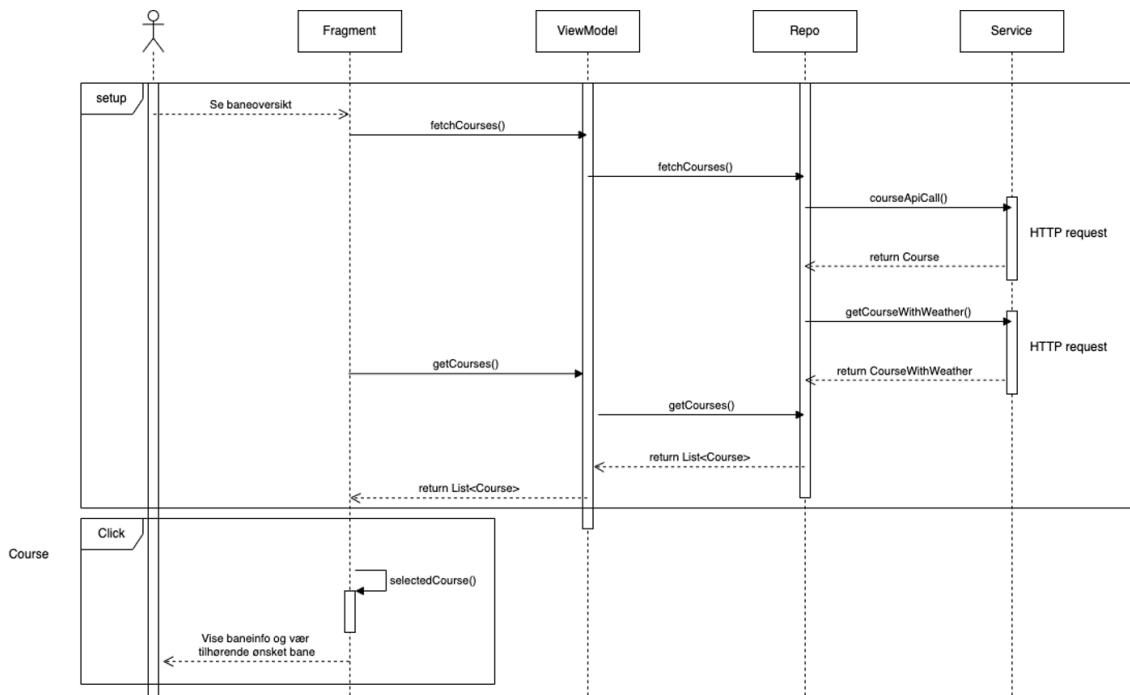
Use case 1 innebærer at bruker velger bane med tilhørende værforhold og baneinformasjon. Figur 7 illustrerer use case 1 med tilhørende tekstlig beskrivelse. Aktørene i use caset representerer ulike roller som mennesker eller et annet system. I dette tilfellet er brukeren en primæraktør, mens Android, Metrologisk institutt og Google Maps utgjør sekundæraktører. Use caset inneholder exclude relasjoner som beskriver tilleggsoppførsel som kan utføres under alternativ flyt (Lindsjørn, 2022a).



Figur 7: Use Case 1, med tilhørende brukerhistorie, av bruker som navigerer seg frem på applikasjonen for å finne værforhold og informasjon om ønsket frisbeegolfbane. Strekfigurer representerer ulike aktører, ovalene indikerer ulike use case, heltrukne streker er relasjoner, og striplet linje er exclude relasjoner.

Sekvensdiagram

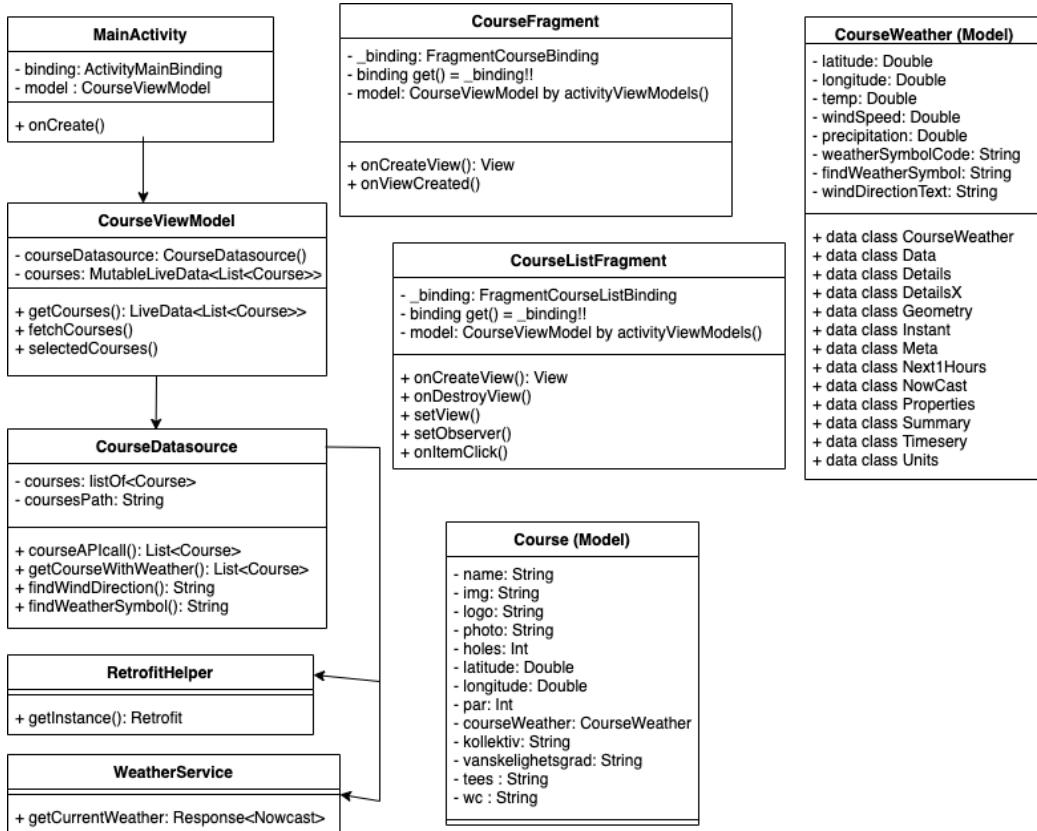
Figur 8 illustrerer et sekvensdiagram basert på use case 1. Illustrasjonen viser både hva som skjer ved oppstart (setup) av applikasjonen og hva som skjer når bruker trykker på en gitt bane (course) i applikasjonen. Objektene i sekvensdiagrammet er forenklet for å gi en bedre fremstilling over hva som skjer i systemet. Objektene i systemet er kategorisert som Fragment, ViewModel, Repo (repository) og Service.



Figur 8: Sekvensdiagram basert på use case 1. Strekfigur representerer brukeren. Fragment, ViewModel, Repo og Servic representerer ulike objekter. Sekvensene representerer objektenes metoder. Heltrukne piler indikerer metodekall, mens striplet piler indikerer retur.

Klassediagram

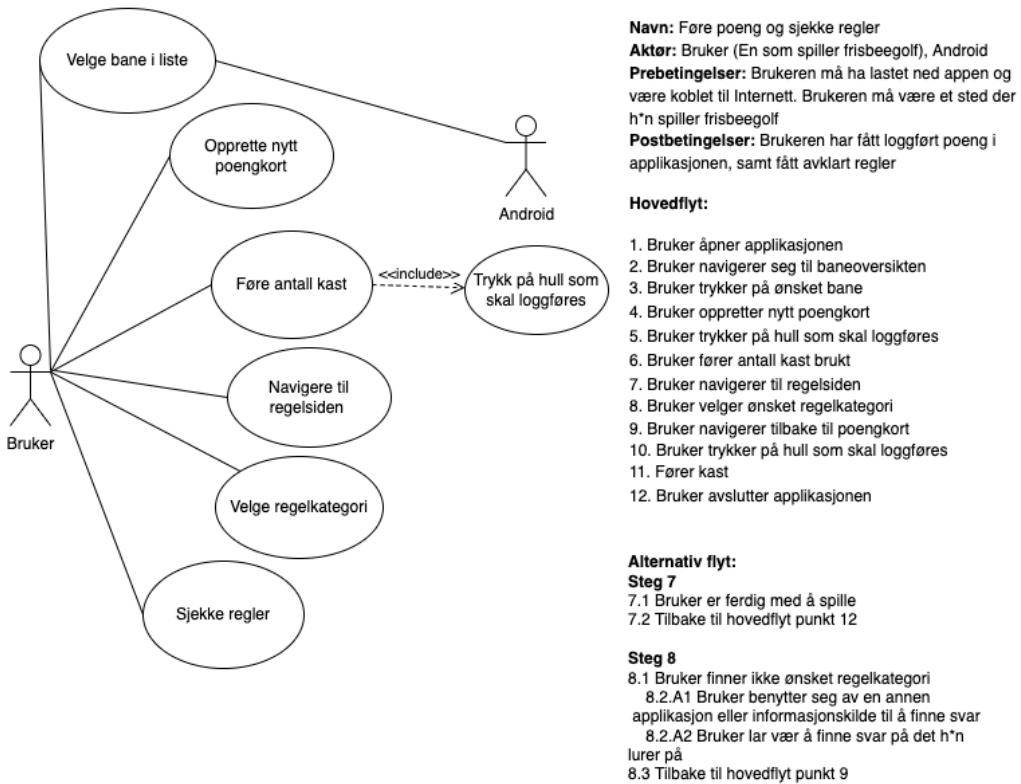
Figur 9 illustrerer et klassediagram basert på use case 1. Illustrasjonen viser hvilke klasser som er sentrale i systemet.



Figur 9: Klassediagram basert på use case 1. Boksene illustrerer ulike klasser. Tegnene (-) og (+) representerer henholdsvis tilhørende attributter og operasjoner i klassen. Heltrukken pil representerer enveis assosiasjon.

3.2.2 Use case 2

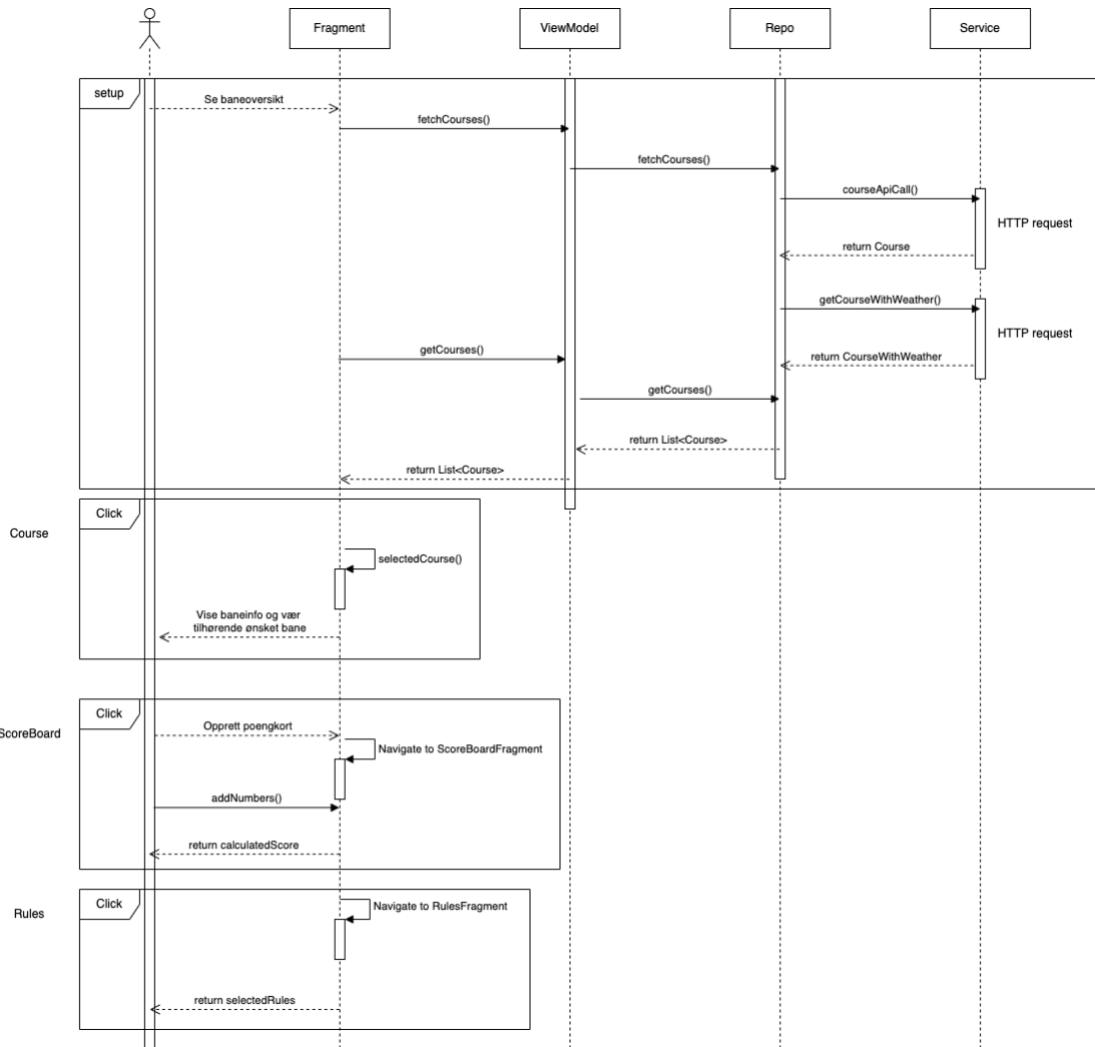
Use case 2 innebærer at bruker fører antall poeng i poengkortet, og sjekker regler i applikasjonen underveis for korrekt poengføring. Figur 10 illustrerer use case 2 med tilhørende tekstlig beskrivelse. Bruker og Android er henholdsvis primæraktør og sekundæraktør i use caset. Use caset inneholder include-relasjoner som indikerer use case som kan være en del av ett eller flere use cases (Lindsjørn, 2022a).



Figur 10: Use case 2, med tilhørende brukerhistorie, av bruker som fører antall poeng i poengkortet og sjekker regler i applikasjonen underveis for korrekt poengføring. Strekfigurer representerer ulike aktører, ovalene indikerer ulike use case, heltrukne streker er relasjoner, og striplet linje er include relasjoner.

Sekvensdiagram

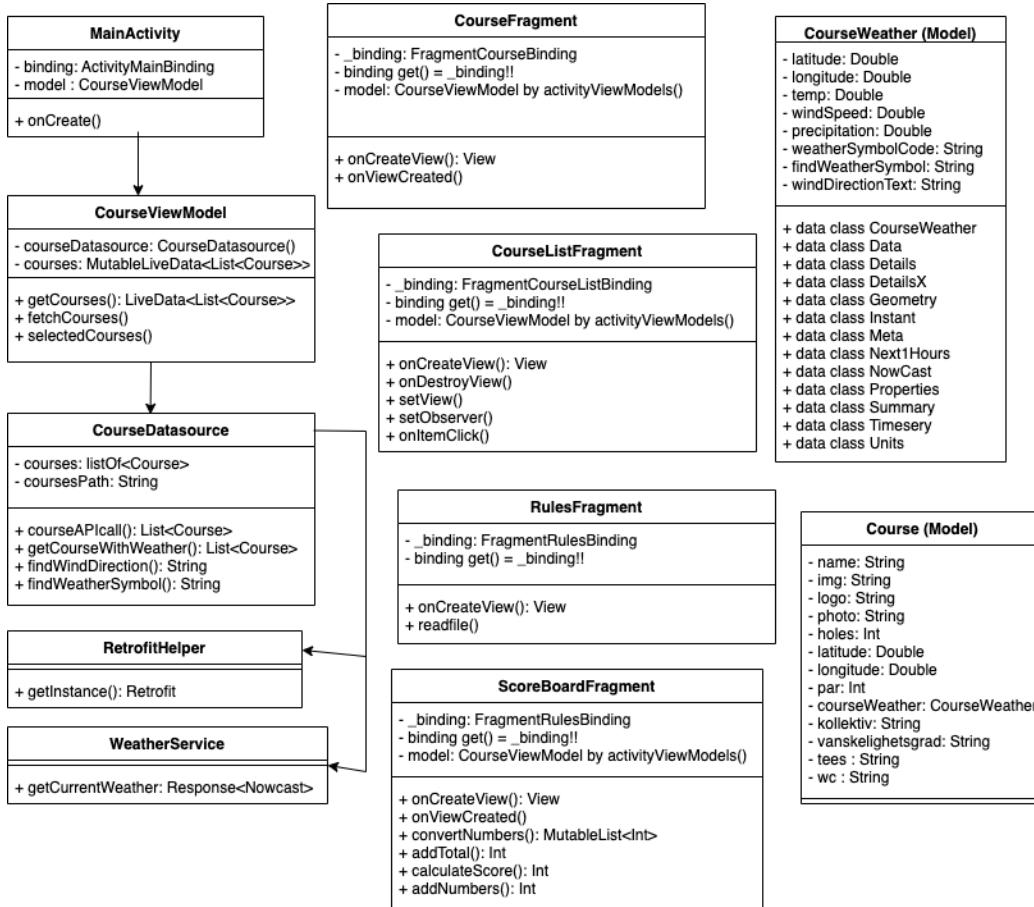
Figur 11 illustrerer et sekvensdiagram basert på use case 2. Illustrasjonen viser hva som skjer ved oppstart (setup) av applikasjonen, og hva som skjer når bruker trykker på en frisbeegolfbane (course), poengkortet (scoreboard) eller regelsiden (rules) i applikasjonen. Objektene i sekvensdiagrammet er forenklet, og objektene i systemet er kategorisert som Fragment, ViewModel, Repo (Repository) og Service.



Figur 11: Sekvensdiagram basert på use case 2. Strekfigur representerer brukeren. Fragment, ViewModel, Repo og Servic representerer ulike objekter. Sekvensene representerer objektenes metoder. Heltrukne piler indikerer metodekall, mens striplet piler indikerer retur.

Klassediagram

Figur 12 illustrerer et klassediagram basert på use case 2. Illustrasjonen viser hvilke klasser som er sentrale i systemet.



Figur 12: Klassediagram basert på use case 2. Boksene illustrerer ulike klasser. Tegnene (-) og (+) representerer henholdsvis tilhørende attributter og operasjoner i klassen. Heltrukken pil representerer enveis assosiasjon.

3.3 Ikke-funksjonelle krav

Ved å fokusere på ikke-funksjonelle krav under utviklingsprosessen kan gruppemedlemmene få en forståelse for hvordan systemet skal implementere de funksjonelle kravene. Basert på ISO 25010 Software Quality Model ble de mest relevante produktegenskapene for *FROLF* bestemt å være brukervennlighet (usability) og ytelse (performance efficiency) (ISO, 2017). Brukervennlighet er knyttet til hvor greit systemet er å bruke. Som tidligere nevnt ble målgruppen satt til å være nybegynnere innen frisbeegolf, uavhengig av alder. Med en bred målgruppe er det nødvendig at systemet skal oppfattes som enkelt å lære og bruke. På bakgrunn av dette ble det satt som mål at applikasjonen skal ha minimale krav til opplæring ved bruk. I tillegg vil det være nødvendig å tenke på universell

utforming. For å kunne oppfylle brukervennlighet som produktkrav, vil det være relevant å utføre brukertester for å undersøke brukervennligheten. Produktkravet ytelse er knyttet til hvor raskt og hvor mye plass et system vil trenge (Lindsjørn, 2022c). Det er ønskelig at *FROLF* skal være en tids-effektiv applikasjon. Under utviklingen må det vurderes hva som skal lagres lokalt på applikasjonen, og hva som kan hentes kun ved bruk av internett. Dette vil i stor grad være avhengig av tilgjengelige API, og relevant dataformatering. Under arbeidsprosessen av det nevnte produktkravet vil det være nødvendig å jevnlig teste applikasjonen og minimere antall feil i systemet.

3.4 Designmønster

Kravspesifikasjonen og UML-diagrammene gir grunnlag for design og implementasjon av systemet. Under implementasjonen vil det være nødvendig å fordele ansvar for utviklingen av systemet, og tillate gruppemedlemmene å sitte med hver sin enhet. Etterhvert som de ulike enhetene av systemet er klare, kan de fusjoneres. For å kunne jobbe på denne måten var det viktig at gruppen hadde en felles forståelse for arkitekturen og valgte en egnet strategi for utviklingen. Det var derfor nødvendig å bestemme seg for et designmønster å jobbe etter. Et egnet designmønster for et system vil kunne gi retningslinjer for ansvarsfordeling, og tilrettelegge for at systemet kan være oversiktlig og brukervennlig.

Model-View-ViewModel (MVVM) er en type designmønster som brukes i Android-utvikling. MVVM er basert på å dele programmet inn i komponentene modell (model), visning (view) og visningsmodell (viewmodel), med ulike roller. Modell holder på dataene, visning holder brukergrensesnittene, mens visningsmodell er bindingen mellom modellen og visningen, og har som rolle å konvertere dataobjektene fra modellen slik at objektene kan administreres og presenteres. Basert på MVVM kan det benyttes en tolags-logisk arkitektur under utviklingen av applikasjonen, med et presentasjonslag (UI layer) og et datalag (Data layer). View og ViewModel vil være knyttet til presentasjonslaget, mens Model er knyttet til datalaget (Android for Developers, 2022d). For eksempel, i sekvensdiagrammene var de identifiserte objektene Fragment, ViewModel, Repo og Service. Basert på dette prinsippet representerer Fragment og ViewModel presentasjonslaget, mens Repo og Service representerer datalaget. I klassediagrammene vil CourseFragment, CourseListFragment og CourseViewModel være relatert til presentasjonslaget, mens CourseDatasource, RetrofitHelper, WeatherService, Course (model) og CourseWeather (model) vil være knyttet til datalaget. Arkitekturen til applikasjonen vil kunne bli mer forutsigbart og lesbart for både gruppemedlemmene og eksterne.

Fra UML-diagrammene er det tydelig at systemet for applikasjonen vil inneholde samtlige objekter med ulike roller. Det vil være nødvendig å hele tiden vurdere kohesjonen og koblingen i systemet. Det vil være ønskelig å oppnå høy kohesjon og lav kobling. Høy kohesjon innebærer at objektene i systemet har moderat ansvar med oppgaver innenfor et funksjonelt område. Under modelleringen ble det forsøkt å lage mindre klasser med konkrete ansvarsområder. Ved lav kobling har objektene i systemet begrenset samarbeid med andre objekter, og objektene er i mindre grad avhengig av hverandre (Lindsjørn, 2022b). Under implementasjonen må det kontinuerlig vurderes om objektene kan bli mindre, få færre ansvarsområder og redusere avhengigheten til andre objekter. Dette kan

resultere i en enklere utviklingsprosess, en mer skalerbar applikasjon og gjøre systemet mer testbart. Systemet blir mer testbart, fordi problemene er spesifikke og separate. Dette fører til at eventuelle feil i en del, ikke vil redusere kvaliteten i resten av programmet.

4 Designprosess

Designprosessen i *FROLF* består av tre hoveddeler. Den første delen tar for seg datainnsamlingen i starten av prosjektet, og inkluderer en kartleggende spørreundersøkelse og et dybdeintervju. Den andre delen forklarer endringene i designet fra den første skissen til det endelige produktet, og den siste delen tar for seg planlegging og gjennomføring av brukertesting gjort senere i prosjektet.

4.1 Spørreundersøkelse og dybdeintervju

Ettersom at det var planlagt en datainnsamling tidlig i prosessen var det naturlig å benytte seg av en kartleggende spørreundersøkelse. Spørreundersøkelser gir tilgang til et stort antall mennesker, og er derfor nyttig for å skaffe seg et overblikk eller en “avbildning” av en brukergruppe. (Lazar mfl., 2017, s. 124). En spørreundersøkelse ble derfor utført for å få en oversikt i bredden. Undersøkelsen regnes i hovedsak som kvantitativ, til tross for at den noen steder samlet inn deltakernes personlige meninger og refleksjoner. Spørreundersøkelsen er lagt ved i vedlegg C.

I tillegg til bredde var det ønskelig å se på materialet i dybden. Direkte samtaler med færre deltakere kan skaffe nyttige perspektiver og brukbar data som spørreundersøkelser kan overse (Lazar mfl., 2017, s. 223). Ved å holde et dybdeintervju i tillegg til spørreundersøkelsen, ble dataen triangulert, og validiteten til svarene fra undersøkelsen kunne dermed vurderes (Lazar mfl., 2017, s. 14).

4.1.1 Gjennomføring av spørreundersøkelse

Det var særlig fem områder som var interessante når det gjaldt funksjonalitet og design; andre applikasjoner, vær, layout, regler og andre tilleggsfunksjoner. Undersøkelsen ble gjennomført ved bruk av UiO sitt innsamlingsverktøyet kalt Nettskjema (UiO, 2022). Dette verktøyet var foretrukket å bruke siden det er en sikker løsning. Undersøkelsen ble delt gjennom plattformen Facebook på to av medlemmenes private sider, i tillegg til to grupper for frisbee-entusiaster. De som svarte ble anonymisert. Etter en opprydding i svarene var det totalt 189 svar som ble tatt i beregning. Av disse 189 deltakerne var de fleste menn i aldersgruppen 18-25 år. Nesten halvparten av dem spiller flere ganger i uka, og over halvparten av dem beskriver sine ferdigheter som middels gode. Resultatene fra spørreundersøkelsen er lagt ved i vedlegg D.

Andre applikasjoner

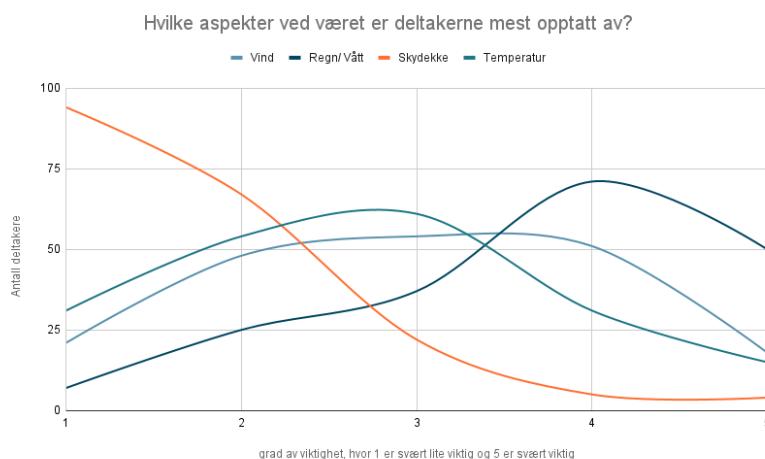
Innledningsvis ble det nevnt at gruppen var gjort oppmerksom på at det allerede fantes flere applikasjoner for frisbeegolfere. En populær applikasjonen er blant annet UDisc. Det var derfor av interesse å undersøke hvor populær denne applikasjonen var blant deltakerne, og hva de eventuelle negative og positive sidene ved den var. Resultatene viste at hele 78% av deltakerne benytter seg av UDisc, og at de fleste er svært fornøyde med hva den har å tilby. Mange synes at den er praktisk og enkel i bruk. Vær oppmerksom på at deltakerenes ferdigheter kan ha innvirkning på svarene, i og med at andelen

aktive frisbeegolfere var ganske stor. Basert på svarene var det to kommentarer som skilte seg ut. Én av deltakerne mente at applikasjonen kunne være “litt mye å forstå i begynnelsen”. En annen mente at den hadde et “noe kronglete brukergrensesnitt”.

Været

I tillegg til å være et førstevalg for nybegynnere, var et sentralt fokus for prosjektet å kunne samle værdata og banedata til ett sted. Det var dermed interessant å finne ut hvilke aspekter ved været som er viktigst for deltakerne når de skal spille. Ikke minst om det var et behov for en frisbeegolf-applikasjon med værdata i det hele tatt. Resultatene viste et stort flertall som sjekket været dersom de skulle spille (86,2%).

Figur 13 illustrerer i hvilken grad deltakerne har ment at de ulike aspektene ved været er viktig for dem. Grafen viser at nedbør er det som har mest å si for frisbeegolferne. Vind og temperatur virker å være middels viktig, mens skydekke er uinteressant for de fleste. Deltakerne kunne også legge igjen en kommentar på slutten av undersøkelsen dersom de ville det. Her var det blant annet en som skrev at det hadde vært “dødskult” med en værfunksjon. En annen sa at det er viktig for han å sjekke forholdene for å kunne kle seg korrekt.



Figur 13: Basert på svarene i spørreundersøkelsen ble det laget et linjediagram med aspekter ved været.

Regler

I tillegg til å kombinerebane- og værdata, var en av de tenkte funksjonene til applikasjonen å gi en enkel og lokal tilgang til regler under spilling. En del av undersøkelsen ble viet til spørsmål om dette. Resultatet viste at litt over 86% av deltakerne er opptatt av regler når de spiller, og rundt det samme prosenten av deltakere svarte at de ville søkt opp reglene dersom det oppsto en uenighet på banen.

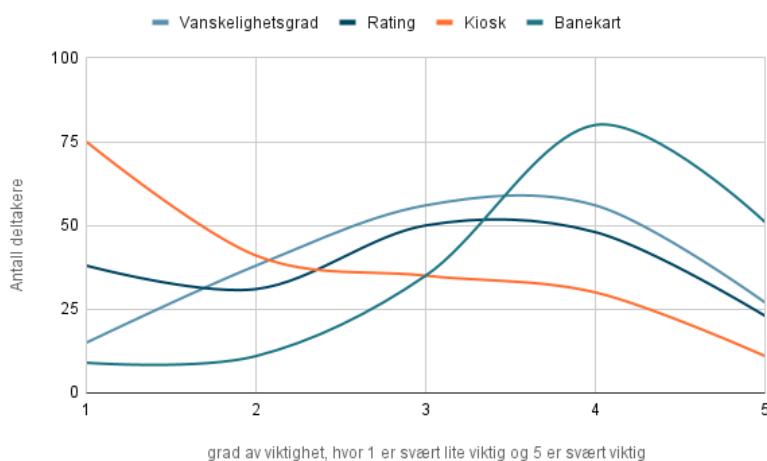
Layout

En naturlig del av designprosessen ville være å undersøke hva de potensielt fremtidige brukerne av applikasjonen foretrakk av layout. Resultatene viste at deltakerne foretrakk å navigere gjennom en

navigasjonsbar. I tillegg foretrakk deltakerne heller å finne banene i en kartoversikt, sammenlignet med en listeoversikt.

Andre funksjoner

En idémyldring innad i gruppen gjorde at følgende tilleggsfunksjoner i applikasjonen ble vurdert; informasjon om vanskelighetsgrad, rating, banekart og nærmeste kiosk. For å sørge for at funksjonene som var viktigst for målgruppen var de som ble implementert, ble det inkludert en mulighet for rangering av disse funksjonene i spørreundersøkelsen. Figur 14 illustrerer et linjediagram basert på svarene. Figuren viser at det viktigste for spillerene er banekart. Vanskelighetsgrad og rating virker å være litt over middels viktig.



Figur 14: Basert på svarene i spørreundersøkelsen ble det laget et linjediagram over viktigheten av ulike funksjoner.

4.1.2 Gjennomføring av dybdeintervju

Ettersom et flertall av deltakerne i spørreundersøkelsen var aktive frisbeegolfspillere, var gruppen interessert i å holde et intervju med en nybegynner eller en middels god frisbeegolfspiller. På denne måten ville innsamlingen hente informasjon fra en bredere andel av brukergruppen, og øke sannsynligheten for at applikasjonen til slutt ville støtte flere ferdighetsnivåer.

Før intervjuet tok sted ble det utformet en intervjuplan. Intervjuplan er vedlagt i vedlegg E. Det ble bestemt at intervjuet skulle være semi-strukturert, da dette tillater intervjuobjektets kommentarer å lede til nye spørsmål som kanskje ikke hadde blitt tenkt på tidligere (Lazar mfl., 2017, s. 8.5.1). Intervjuet ble holdt av de to gruppemedlemmene med mest frisbeegolf-erfaring. På denne måten ble det sikret best mulig flyt i intervjuet, da både intervjuer og intervjuobjekt hadde kjennskap til de samme begrepene. Det ble tatt opptak av intervjuet, noe som intervjuobjekt ble gjort oppmerksom på gjennom et samtykkeskjema. Samtykkeskjemaet er vedlagt i Vedlegg F. Spørsmålene stilt i intervjuet var mer eller mindre de samme som i spørreundersøkelsen. Hovedformålet med intervjuet ble derfor å undersøke om brukeren hadde innspill som støttet eller motsa resultatene fra spørreundersøkelsen, og eventuelt om annet viktig hadde blitt oversett.

Ettersom at dataen fra intervjuet var utelukkende kvalitativ, ble den først transkribert, før den deretter ble sortert inn i et affinity-diagram. Et affinity-diagram utvikles ved hjelp av virtuelle post-it-lapper. Dette muliggjør å samle notater fra intervjuet inn i et hierarkisk kart av problemområdet. Dette kan hjelpe med å forstå forholdet mellom områder i de ulike temaene og undertemaene (Lazar mfl., 2017, s. 8.5.2). Diagrammet er lagt ved i vedlegg G.

Intervjuobjektet svarte mye av det samme som deltakerne i spørreundersøkelsen. I det ene intervjuet ble det bekreftet at nedbør og vind er spesielt viktig under et spill. I tillegg til vindhastighet mente intervjuobjektet at vindretning var viktig. Når intervjuobjektet ble spurta om regler var det tydelig at de viktigste reglene var knyttet til hvordan spillere skal oppføre seg på frisbeegolfbanen. Intervjuobjektet mente at applikasjonen burde ha en generell intro for de som er nye på banen, slik at nybegynnere vet hvordan de skal oppføre seg. Andre funksjonaliteter som ble nevnt var; informasjon om når sola går ned, nærmeste parkering, vanskelighetsgrad og spilletid.

4.1.3 Funn fra spørreundersøkelse og dybdeintervju

Et par av medlemmene hadde i tidligere prosjekter hatt en brukersentrert tilnærming til design. I disse prosjektene har resultatene fra datainnsamlingen i større grad vært det som har bestemt implementasjon og funksjon. I dette prosjektet derimot, handlet det om å lage et programvareprodukt, noe som betyr at utviklerne bestemmer applikasjonens funksjonalitet (Sommerville, 2019, s. 2). Derfor ville resultatene fra datainnsamlingen i denne sammenhengen virke mer som en bekrefte eller avkrefte av idéer, heller enn det som bestemte funksjonaliteten.

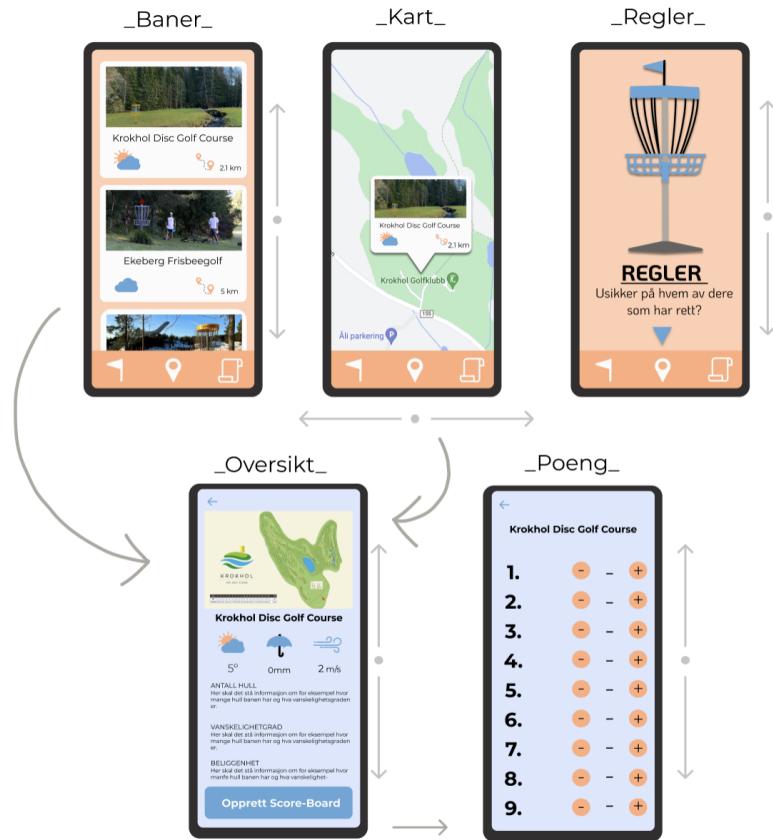
Funnene fra datainnsamlingen stemte godt overens med visjonen som ble utarbeidet i begynnelsen av prosjektet, beskrevet senere i rapporten i *Prosessdokumentasjon*. Værdata er identifisert som viktig for frisbeegolferne i spørreundersøkelsen, og det ble bestemt at informasjon om regn, vind og temperatur skal inkluderes i designet. Ingen av de andre frisbeegolf-applikasjonene tilbyr informasjon om været, og vil være en unik funksjonalitet i *FROLF*. Til tross for at deltakerne ikke var spesielt opptatt av skydekke, ble det bestemt at et skydekke-ikon skal vises i kombinasjon med temperaturen av estetiske grunner, og siden enkle værapplikasjoner alltid har med skydekke. I tillegg til vindhastighet ble informasjon om vindretning inkludert i designet siden et av intervjuobjektene uttrykte at dette var viktig.

Spørreundersøkelsen og dybdeintervjuet bekreftet at regler spiller en viktig rolle i et frisbeegolf-spill. Det ble enighet om at regelsiden bevares, og at den skal kunne aksesseres under spilling. For å opprettholde visjonen om en enkel applikasjon, ble det bestemt at kun de viktigste reglene skal presenteres i regelsiden. De mer avanserte reglene skal kunne aksesseres gjennom en spinner. Blant annet vil “etikette” reglene som det ene intervjuobjektet ønsket seg, vises her.

I tillegg til å bekrefte visjonen ga innsamlingen en oversikt over hvilke funksjoner som var mest appetiserende for de fremtidige brukerne. Siden flertallet foretrakk å finne banen gjennom en kartoversikt, ble gruppen enige om at dette skal være hovedsiden til applikasjonen.

Som nevnt tidligere uttrykte det ene intervjuobjektet tydelige krav som ble inkludert i kravspesifikasjonen. Kravene ble plassert på må, bør, og kan, alt etter som hva gruppemedlemmene selv var enige

i, og opplevde som realistisk å implementere. De oppdaterte kravene er allerede vist i tabell 1. Ut fra kravene ble følgene skisse av design utarbeidet, illustrert i figur 15.



Figur 15: Digital skisse av layout etter brukerinnsamling.

4.2 Designendringer

Til tross for at det ble observert store likheter mellom den første skissen og det endelige designet, ble det gjort flere endringer i løpet av utviklingsprosessen. Noen endringer ble gjort basert på det utviklerne ønsket for applikasjonen, og andre endringer ble gjort basert på det deltakerne i de ulike datainnsamlingene ønsket.

Poengkort

Det største avviket fra den første skissen er poengkortets utforming. I utgangspunktet var det tenkt at brukeren skulle loggføre antall kast ved hjelp av pluss- og minusknapper slik som det er illustrert i figur 15. Gruppen fant fort ut at denne løsningen ble en utfordring å implementere da énbane ofte har opptil 18 hull. Altså hadde dette fragmentet trengt 36 knapper og 36 ulike onClick-listeners med egen funksjonalitet. I tillegg kan et design som er avhengig av knapper for å føre poeng bli frustrerende dersom det skal loggføres et høyt antall kast. Det ble dermed enighet om at kast skulle føres ved bruk av et tall-tastatur istedenfor.

I utgangspunktet var det tenkt at muligheten for å beregne total og poeng skulle være nederst på

poengkortet. Dette virket intuitivt dersom det skulle legges sammen tall. Ettersom kortet tok form fant gruppen ut at poeng-beregneren var mer tilgjengelig hvis den lå øverst på skjermen. Beregneren hadde vært skjult ved første møte dersom den hadde ligget nederst på skjermen. Ved å flytte den øverst har brukeren oversikt over total fra begynnelsen av. Med denne endringen vil brukeren forhåpentligvis bli bedre kjent med beregneren og dens funksjonalitet. Med denne endringen kan brukeren sjekke totalpar på banen med én gang poengkortet opprettes.

Navigasjon

Navigasjonen i applikasjonen ble endret fra slik det opprinnelig var tenkt. I utgangspunktet skulle bruker kunne komme til baneoversikten gjennom både banelista og kartet. Dette var bakgrunnen for å spørre deltakerne hvilken oversikt de foretrak (se delkapittel om *Brukerundersøkelse*). Gruppen fant ut at navigasjon fra kart til baneoversikt var vanskeligere å implementere enn antatt. Derfor ble denne funksjonaliteten nedprioritert til fordel for å sikre stabilitet i applikasjonen og i det hele tatt å få kartet til å fungere. Kartet sin nåværende funksjon er i hovedsak å se avstand fra egen posisjon og å få veibeskrivelser. Ved en eventuell videreutvikling av applikasjonen hadde navigasjon fra kart til baneoversikt vært essensielt, da det er denne informasjonen som er unikt ved produktet.

Figur 15 viser at det var tenkt at navigasjonsbaren skulle forsvinne når ved tastetrykk på en bane, slik at oversikten dekket hele skjermen. Gruppen fant etterhvert ut at dette ville fjerne brukerens muligheten til å sjekke reglene samtidig som poeng føres. Ettersom dette var en viktigere funksjon enn at oversikten skulle dekke hele skjermen ble det bestemt at navigasjonsbaren skulle være tilgjengelig i alle fragmentene.

Estetikk

Rent estetisk er sluttproduktet ganske likt de første skissene. Da muligheten for å navigere fra kart til oversikt ikke ble implementert ble det bestemt at banemarkørene skulle holdes enkle. Gruppen så for seg et vindu med begrenset informasjon ville virke forvirrende dersom det naviges videre til mer informasjon. For å opprettholde visjonen om et enkelt design ble det valgt at baneoversikten skulle inneholde så lite informasjon som mulig. Dette er illustrert i figur 15.

4.3 Brukertesting

Etter den første datainnsamlingen stod utvikling av systemet i fokus i en lengre periode. Grunnet begrenset tid ønsket gruppen å sikre at de grunnleggende funksjonene ble implementert før de ble testet. Det var ønskelig å unngå distraksjonene som ofte oppstår når det involveres flere mennesker i en designprosess. Gruppen så likevel verdien i å involvere brukere i produktutviklingen, da det er de som skal benytte seg av applikasjonen. Det ble bestemt at en brukbarhetstest skulle utarbeides, og at den skulle gjennomføres når den grunnleggende funksjonaliteten var på plass. Når testingen skjer på et sent tidspunkt i den generelle utviklingsprosessen utføres det som kalles en summativ brukbarhetstest (Lazar mfl., 2017, s. 272).

Målet med brukbarhetstesting er å teste om produktet oppleves brukbart for målgruppen når de gjør de oppgavene det var designet for, samt å teste hvorvidt brukerne er tilfreds med sin opplevelse

(Sharp mfl., 2019, s. 534). I følge boken “*Research Methods in HCI*” av Jonathan Lazar m.fl. finnes det flere ulike måter å gjennomføre en brukbarhetstest på. I dette prosjektet er det tatt utgangspunkt i Rubin og Chisnell sin prosessmodell fra 2008. Den første delen av denne modellen handler om å utvikle en plan. Planleggingen innebærer å fastslå en lokasjon og finne deltakerne (Lazar mfl., 2017, s. 274). Resten av prosessen innebærer å utføre brukertesten, utspørre deltakerne, analysere dataen og rapportere funn. Prosessen er illustrert i figur 16.



Figur 16: Rubin og Chisnell sin prosessmodell oversatt til norsk (Lazar mfl., 2017, s. 274).

4.3.1 Gjennomføring av brukertesting

Planen for undersøkelsen ble utarbeidet i et Google dokument. Her ble blant annet dato, tidspunkt og lokasjon fastsatt. Testen ble gjennomført under kontrollerte omgivelser, på et lukket rom for å finne de mest åpenbare feilene. I motsetning til å ha testen i naturlige omgivelser, hvor i dette tilfellet ville vært på en frisbeegolfbane. I disse omgivelsene ville det vært vanskeligere å luke ut åpenbare feil på grunn av mange miljøfaktorer som vil påvirke (Lazar mfl., 2017, s. 276).

Brukertesten ble gjennomført på IFI med tilfeldige studenter som deltagere. Under brukertesten ble deltakerne presentert en oppgaveliste med ulike oppgaver som brukerne skulle gjøre i applikasjonen. Målet var at disse oppgavene skulle teste alle funksjonene til produktet. Listen inneholdt derfor spørsmål om kartet, værdata, banedata, regelsiden og poengkortet. Planen for brukertesting er vedlagt i vedlegg H. Et samtykkeskjema som sikrer deltakernes rettigheter ble utarbeidet og skrevet ut. Det ble signert av deltakerne i forkant av testen. I skjemaet ble det informert om at det ville bli tatt opptak, og at de kunne trekke seg når de ville. Deltakerne gjorde testingen individuelt under de kontrollerte omgivelsene. To av gruppemedlemmer var tilstede under testingen, hvor den ene hadde overordnet ansvar, og den andre noterte hva deltakeren gjorde. Applikasjonen ble kjørt på en Android-enhet. Enheten lå hele tiden på bordet slik at den kunne observeres av gruppemedlemmene.

Etter oppgavene var utført hadde deltakeren mulighet til å komme med tilbakemeldinger.

4.3.2 Funn fra brukertesting

Gjennom brukertestingen kunne gruppen observere hvordan brukerne reagerte i forhold til bruk av applikasjonen. Dette var en viktig del av utviklingsfasen. På denne måten kunne feil og mangler kartlegges, eller oppdage deler som var uklare eller upraktiske for brukeren på en kort og billig måte. Dataen var viktig for videre utvikling av applikasjonen og for å kunne vurdere om de ulike målene ble nådd.

Funn fra brukertesten er utfordringer knyttet til navigasjon, poeng, tilgjengelighet til banefragmentet fra kart, og overflødig informasjon. Samtlige av deltagerne forventet at navigasjonsbaren nederst i applikasjonen fungerte som en tilbakeknapp. Dette er en vanlig funksjonalitet i andre applikasjoner. Enkelte av deltakerne hadde utfordringer med å finne spinneren til regelsiden trolig på grunn av lav kontrast mellom spinneren og bakgrunnen. Da deltakerne trykket på “Beregn poeng”-knappen, ble de sendt helt ned til bunnen av siden og måtte bla opp igjen for å se det endelige resultatet. Dette opplevde de som forvirrende. Noen av brukerne reagerte på at banelisten ikke var alfabetisk. En av brukerne var ikke veldig kjent med stedsnavn i Oslo. Derfor etterlyste denne brukeren en søkefunksjon for å kunne se navnene til de ulike banene i kartet. En av brukerne påpekte at det var overflødig å inkludere informasjon om kollektiv trafikk inne på banesiden, siden de heller gikk inn på Google Maps-applikasjonen fra kartet i *FROLF*.

Positive tilbakemeldinger var blant annet at brukerne likte designet på poengkortet og at de var imponert over det generelle utseendet av applikasjonen. En av deltakerne påpekte at de likte at poengkortet i *FROLF* fordi det lignet på poengkortet som benyttes i golf. Det var utfordringer knyttet til å gjøre endringer i designet av applikasjonen. Brukertesten ga ikke nødvendigvis informasjon om hvordan applikasjonen kunne forbedres, men om hva som måtte endres. Umiddelbare endringer som gruppen valgte å prioritere er alfabetisk rekkefølge på banene og at poengkortet hoppet ned til bunnen.

5 Produktdokumentasjon

5.1 Teknologi

Applikasjonen er utviklet i programvareverktøyet Android Studio, versjon Bumblebee 2021.1.1. Android Studio er det offisielle IDE (Integrated Development Environment) for utvikling av Android-applikasjoner, og er basert på IntelliJ Jetbrains IDEA (Android for Developers, 2022e). Minimum SDK (Software Development Kit) nivå ble satt til API:29, Android 10 (Q). Programmeringsspråket Kotlin ble brukt for backend-utvikling av applikasjonen. Kotlin er industristandard for Android og er utviklet av JetBrains (Android for Developers, 2022b). UI layout for applikasjonen er deklarert i XML i Android Studio. XML er en samlebetegnelse for eXtensible Markup Language og er et oppslagsspråk (Microsoft, 2022). Versjon 1.0 ble brukt. Android Studio støtter en rekke ulike Android-emulatorer som har nesten alle funksjonene tilsvarende en ekte Android-enhet (Android for Developers, 2022h). Ved bruk av Android-emulator var det mulig å teste applikasjonen på laptop.

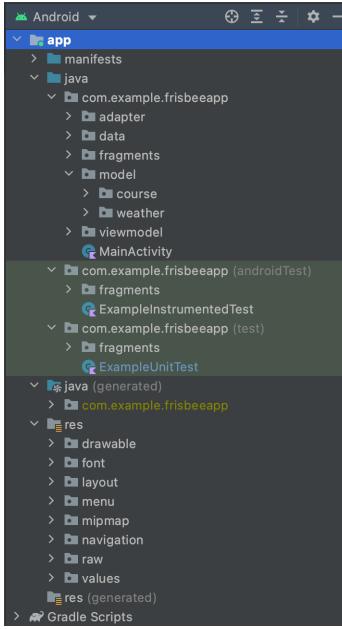
Foretrukket emulator for prosjektet var Pixel 4 XL API 29. For grunnleggende testing ble JUnit4 benyttet. JUnit4 er en innebygd funksjon i Android Studio som enkelt kan benyttes for å lage, kjøre og se resultater av tester (Android for Developers, 2022j). Integrasjonstest ble utført ved bruk av emulatoren. GitHub ble brukt for versjonshåndtering. Verktøyet har tilrettelagt for koordinering av samarbeid, og håndtering og administrere endringer på prosjektet.

5.2 Valg av API-nivå

API-nivået for applikasjonen ble bestemt å være 29. Dette API-nivået er relativt nytt, og har stor dekningsgrad av Android-enheter. Nivået dekker omtrent 50,8% av Android-enheter. Riktig valg av API-nivå er svært viktig for at applikasjonen skal kunne ha ønsket funksjonalitet og samtidig passe målgruppen best mulig. På bakgrunn av de funksjonelle kravene nevnt i kravspesifikasjonen var nivå 29 det laveste som kunne bli valgt. For eksempel, kan-kravet 3.6 er knyttet til dark mode som først støttes i nivå 29. En annen viktig fordel er at nivå 29 har et større fokus på personvern, spesielt da det kommer til lokasjonsdata. Dette er noe gruppen ser på som en nødvendighet, ettersom applikasjonen kan ta i bruk brukerens posisjon om dette er ønskelig, slik det er nevnt i kan-kravet 2.0. Det ble antatt at målgruppen til *FROLF* er frisbeegolfspillere i Norge. Majoriteten av nordmenn kan betraktes som teknologi-entusiaster og har en stor betalingsvillighet for toppmodellene til ulike produsenter av smartenheter (Tek.no, 2017). På bakgrunn av dette er det antatt at brukerne av applikasjonen vil ha enheter som støtter nivå 29. Ved eventuell videreutvikling av applikasjonen bør det tas hensyn til å øke API-nivå for å møte Google Plays krav om API, som fra og med august 2022 er API-nivå 31 (Android for Developers, 2022f). Ved å ta i bruk et nyere API-nivå vil det kunne forbedre brukeropplevelsen med flere funksjoner og økt sikkerhet.

5.3 Design og arkitektur

Android-prosjektet er basert på Model-View-ViewModel (MVVM) som designmønster, og har en tolags-logisk arkitektur med et presentasjonslag og et datalag. I prosjektet er filene strukturert i relaterte mapper for å holde orden på de ulike filene. Etter gruppens beste evne som utviklere, er det forsøkt å lage flere mindre klasser fremfor få store klasser med mange ansvarsområder. Basert på ansvarstilordningen til objektene skal kun objektene som har kunnskap, det vil si dataene, ha ansvaret. Derfor er det forsøkt å separere klassene. Dette ble gjort for å oppnå høy kohesjon og lav kobling. Figur 17 illustrerer et skjermbilde av mappestrukturen i prosjektet.



Figur 17: Mappestruktur til Android-prosjektet av *FROLF*.

Visning er det brukeren ser og kan interagere med. I prosjektet er .xml-filene knyttet til utseende og har følgende lokasjon: app; res; layout. Ulike .kt-filer er hva brukeren interagerer med, og har følgende lokasjon: app; java; com.example.frisbeeapp. Model-mappen inneholder dataklassefiler som benyttes til å lage frisbeegolfbane-objekter og vær-objekter. Det meste av data er i form av en model. Mappen navngitt data er et oppbevaringssted for data (repository), og inneholder relevante filer som blant annet håndterer API-kall og henting av data. Adapter-mappen inneholder CourseAdapter.kt, hvor et adapter-objekt er bindeleddet mellom et AdapterView og data realert for visningen. Den er ansvarlig for å lage en visning for hvert element i datasettet (Android for Developers, 2022a). Mappen navngitt fragments inneholder ulike fragmenter. Et fragment er en del av applikasjonens brukergrensesnitt som er bundet til en aktivitet, og er egnet til å administrere brukergrensesnittet til en enkelt skjerm, eller del av en skjerm (Android for Developers, 2022c). Det er også en egen mappe for ViewModel.

5.4 HTTP-biblioteker for deserialisering

For å kunne deserialisere objektene fra API-ene ble det benyttet ulike HTTP-klienter. HTTP-klienter er nødvendig å bruke for å kunne motta, sende og opprette HTTPS-forespørsler. Det ble totalt brukt tre ulike API-er: et for ulike frisbeegolfbaner i Oslo-regionen, et for vær av Metrologisk institutt og kart fra Google Maps. De to førstnevne var skrevet på JSON-format, mens det er ingen deserialisering knyttet til Google Maps. En av fordelene ved å bruke JSON er at det er enkelt og lett å lese. HTTP-klientene Retrofit (Kittinun Vantasin, 2022) og Fuel (Kittinun Vantasin, 2021) ble benyttet til API-ene henholdsvis knyttet til vær og frisbeegolfbaner. Retrofit og Fuel er to ulike industristandarder som er både brukervennlig og veldig dokumentert. Hovedgrunnen til at det ble brukt to ulike klienter var fordi gruppen ønsket å teste ulike klienter, og få bedre kjennskap til de. Klientene sørger for asynkrone API-kall og gjør at prosessene for datahenting skjer uavhengig av hverandre.

På denne måten går ikke responsen utover ytelsen av applikasjonen. Begge klientene bruker Gson i bakgrunnen. Gson er et Java/Kotlin bibliotek for serialisering og deserialisering av objekter til og fra JSON (Google, 2022). I prosjektet ble Gson brukt til å konvertere JSON string til Java/kotlin objekt.

5.5 Egenutviklet frisbeegolfbane-API

Det ble utviklet et eget frisbeegolfbane-API, som inneholder en liste av ulike frisbeegolfbaner i Oslo-regionen. Til hver bane er følgende informasjon gitt i denne rekkefølgen: tilgang på WC (wc), banekart (img), optimalt antall kast (par), klubblogo til eier av banen (logo), banenavn (name), utkaststed (tees), holes (hull), frilufts bilde tilknyttet banen (photo), breddegrad (latitude), forslag for kollektivtransport til frisbeegolfbanen fra Jernbanetorget (publicTransport), lengdegrad (longitude) og vanskelighetsgrad (difficulty). En av de store fordelene ved å bruke et eget API, er at det enkelt kan oppdateres dersom det er nødvendig. Gruppemedlemmene som er utviklere av applikasjonen er derfor ikke avhengig av en ekstern faktor for å regelmessig oppdatere API-et.

API-et er utviklet ved bruk av verktøyet nPoint.io (n:point, 2018). Verktøyet lagrer API-et i et JSON-format på internett, og er gratis å bruke. Det kreves tilgang til API-et for å kunne endre det. En annen fordel med nPoint er at det ikke krever noe form for API-nøkkel for å ta den i bruk. Merk at denne nettsiden kun er en midlertidig løsning for startfasen av utviklingsprosjektet, siden nPoint ikke støtter større og mer kompakte JSON-filer. Dette er derimot en bedre løsning enn å lagre API-et lokalt på maskinen. Per dags dato er det kun frisbeegolfbaner i Oslo og omegn som er en del av API-et. Dersom applikasjonen skal videreføres vil det være en mulighet å legge inn baner i de andre store byene, og til slutt hele Norge.

5.6 API fra Meteorologisk institutt

Et bredt utvalg av data fra Metrologisk institutt er åpent og tilgjengeliggjort på deres nettside api.met.no. I applikasjonen *FROLF* ble API-et Nowcast 2.0 benyttet. Bakgrunnen for valgt API var at dette API-et dekker ønsket funksjonalitet. Applikasjonen skal kun vise umiddelbar værmelding. Nowcast leverer to-timers værvarsel hvor prognosene oppdateres hvert femte minutt, og skal være Metrologisk institutt sitt beste estimat på de nåværende værforholdene. For eksempel ble Location-forecast 2.0 vurdert å bruke i applikasjonen. Dette API-et dekker værmelding for et spesifisert sted med en ni-dagers periode. Grunnen til at Locationforecast ikke ble valgt var fordi det inneholder overflødig data som kan påvirke ytelsen av applikasjonen.

Ved å oppgi breddegrad og lengdegrad for et spesifisert sted gir API-et værmelding for stedet (MeteorologiskInstitutt, 2022). For hver enkelt frisbeegolfbane ble følgende data for den neste kommende timen (next_1_hours) hentet ut: lufttemperatur målt i celsius (air_temperature), nedbørsmengde målt i mm (precipitation_amount), vindhastighet målt i m/s (wind_speed), vindretning i grader (wind_from_direction), og symbolkode for værttegn (symbol_code). API-et er begrenset til Norden. Ved videre utvidelse til områder utenfor Norden, eller ønske av andre funksjonaliteter, kan andre API-er vurderes.

5.7 API fra Google Maps

Kartet i *FROLF* er implementert med bruk av Google Maps sitt API (Google Maps Platform, 2022). API-et krever en egen API-nøkkel, og har en gratis prøveperiode på 90 dager med utløpsdato 10.06.22. For videreutvikling av applikasjonen vil det være nødvendig å betale for kart-tjenesten. Google Maps sine egne funksjoner er benyttet til å legge til markør, det vil si lokasjon til de ulike frisbeegolfbanene, og vise brukerens posisjon. Personvern-funksjonen i Android versjon 10 er benyttet for å muliggjøre at bruker kan dele sin lokasjon med applikasjonen. Kartet er opprettet i et fragment og samsvarer dermed med resten av applikasjonens fragmenter. Dermed kan bruker enkelt navigere seg til og fra kartet ved hjelp av navigasjonsbaren i bunn. Biblioteket EasyPermissions er benyttet for å forenkle håndteringen av rettigheter tilknyttet brukers lokasjon. Selve kartet ble opprettet av metoden `onMapReadyCallback` som blir kalt på asynkront i `onViewCreated` for at applikasjonen ikke skal stoppe under innlasting av kartet. Alt som skal vises i kartet må kodes inn i `onMapReadyCallback`. Markøren, brukers posisjon, zoom-knapper og knapp for å zoome inn til brukers posisjon ligger i callbacket. Hver markør er tilknyttet en `OnClick`-metode som viser navnet til banen når bruker trykker på tilhørende markør.

5.8 Utfordringer med API

Det er ingen garanti for at man vil motta data fra API-ene *FROLF* bruker. Dette kan for eksempel skje dersom API-et er under vedlikehold eller hvis innhold fjernes. Dersom dette blir et tilfelle, vil ikke applikasjonen krasje og den vil returnere en NULL-verdi. Dette gjelder både det selvlagde frisbeegolfbane-API-et og valgt API fra Meteorologisk institutt. En mulig løsning vil være å lagre tidligere data som er hentet ut, slik at man kan bytte ut NULL-verdiene med denne dataen. Videre er også gruppens selvlagde frisbeegolfbane-API en midlertidig løsning, ettersom nPoint ikke støtter større og kraftigere API. Så dersom man ønsker å utvide API-et, bør andre muligheter vurderes. Gruppen valgte å gå bort fra disse løsningene, da tidsfrist ved innlevering nærmest seg.

5.9 Kvalitetsegenskaper ved applikasjonen

De viktigste kvalitetsegenskapene ved applikasjonen er brukervennlighet, ytelse og funksjonalitet. Det kan argumenteres for at *FROLF* har høy brukervennlighet. Den kan brukes av alle med en Android-enhet med API nivå 29 eller høyere. Applikasjonen krever ingen autentisering, og kan brukes umiddelbart etter nedlasting. Ingen personlig data fra brukeren verken lagret eller delt. Brukeren velger selv om det er ønsket å dele lokasjon. Uavhengig av valget vil brukeren ha tilgang på applikasjonen. Brukervennligheten ble evaluert ved hjelp av brukbarhetstesten beskrevet i Designprosess av rapporten.

FROLF har høy ytelse og krever lite lagringsplass. Applikasjonen opptar ca. 25 MB i minne på en Android-enhet. Det kan antas at kartet og regelsiden krever mest minne. Kartet er komplekst og inneholder mange funksjoner som må lastes inn. Regelsiden bruker .txt-filer for de ulike reglene og inneholder et bilde. Det meste av funksjonalitet i *FROLF* er avhengig av internetttilkobling. Værdata, frisbegolfbanene og kart hentes ved bruk av internett. Det er kun regelsiden som er uavhengig av dette. Systemet gjør det den skal gjøre så lenge den har tilgang på internett. Både ytelse og

funksjonalitet ble evaluert ved bruk av testing beskrevet i Testdokumentasjonen i rapporten.

6 Testdokumentasjon

Testing er viktig for å sjekke om produktet man lager faktisk gjør det man har programmert det til å gjøre. Når en test feiler betyr det at man har avdekket et problem som må fikses (Sommerville, 2019, s. 259). I dette prosjektet ble testing gjennomført for å både finne og forhindre programmeringsfeil og forståelsesfeil. Formålet med dette var å sikre kvaliteten til applikasjonen og å unngå systemkrasj. Testingen i dette prosjektet er basert på ISTQBs (The International Software Testing Qualifications Board) syv prinsipper for testing. Det er utført kontinuerlig testing underveis ved hjelp av Logcat og emulator. I tillegg er det skrevet 17 enhetstester og utført en alternativ løsning for integrasjonstesting.

6.1 Testprinsipper

ISTQB har utarbeidet syv prinsipper for testing som skal hjelpe utviklere å finne ut hva, hvordan og når de skal teste. Prinsippene hjelper også utviklerne å finne ut hva de kan forvente og hvordan man skal rapportere funn (Bowen, 2020). Gruppen har forsøkt å ta stilling til disse prinsippene gjennom hele prosjektet - særlig under testingen. De mest relevante prinsippene er følgende: (1) Testing viser tilstedeværelse av defekter, (2) fullstendig testing er umulig, (3) tidlig testing og (7) "fravær av defekter"-misforståelsen.

Prinsipp 1 går ut på at testing benyttes for å vise tilstedeværelsen av feil, ikke for å vise at systemet er feilfritt. Gruppen opplevde dette som veldig sant da det stadig dukket opp områder i koden som inneholdt feil til tross for at disse delene av koden var antatt "feilfrie". Som prinsipp 2 nevner, er fullstendig testing umulig. Derfor vil man heller ta risikovurderinger og prioritere de områdene av koden som er viktigst for at systemet ikke skal krasje. I dette prosjektets tilfelle hadde medlemmene problemer med integrasjonstesting. Det kan tenkes at prioriteringen i testingen hadde vært annerledes dersom medlemmene hadde visst hvordan man skulle gjøre korrekte integrasjonstester i hele applikasjonen. De fleste testene er gjort i poengkortet, til tross for at dette ikke er det viktigste fragmentet i applikasjonen. Gruppen opplever banedataen og værdataen som viktigere, og hadde prioritert å teste disse fragmentene dersom ferdighetene hadde strukket til. Prinsipp 3 går ut på at testing bør starte så tidlig som mulig i utviklingsprosessen. Gruppemedlemmene var fra tidligere emner vant til å kjøre koden ofte. Altså startet systemtestingen tidlig i prosessen. Prinsipp 7 handler om at det ikke hjelper å rette opp i feil som ikke oppfyller brukerens behov og forventninger. Ved å gjennomføre datainnsamling med brukere og brukertesting sørget gruppen for å støtte dette (Henrik Alfheim, 2022).

6.2 Generelt om testing i prosjektet

Den typen testing som ble gjort mest gjennom hele prosjektet er systemtesting. Dette innebærer å teste systemet som helhet, heller enn å teste de individuelle systemegenskapene (Sommerville, 2019, s. 270). Det kan tenkes at systemtesting ble gjort mest, da det ligner mest på kjøring i terminal slik gruppemedlemmene er vandt til fra tidligere emner. Underveis i kodingen ble også logcat benyttet.

Spesifikt med taggen “debug”, slik at man enkelt kunne logge beskjeder til logcat-terminalen. På denne måten var det mulig å sjekke hvilke deler av applikasjonen som fungerte og hvilke som ikke gjorde det, til tross for at hele koden ble kjørt. Dette ligner en slags enhetstesting, da utviklerne var interessert i enkelte deler av koden og ikke alt som helhet.

GitHub ble benyttet under hele prosjektet for å sikre at kun velfungerende kode ble lagt til i systemet. Hver branch ble testet før den ble pushet til master. Det vil si at hvert gruppemedlem var ansvarlig for implementeringen av den koden de hadde skrevet. På denne måten har feil blitt lukket ut, før det ble implementert i koden. Likevel vil ikke være mulig å unngå alle feil, så det vil alltid være behov for testing og vedlikehold.

6.3 Enhetstesting

Målet med enhetstesting er å undersøke programenheter isolert. Testene bør utvikles samtidig som man skriver den tilhørende enheten med kode (Sommerville, 2019, s. 264). Som nevnt tidligere ble applikasjonen kjørt kontinuerlig i emulatoren. Generelt i prosjektet kunne det vært skrevet flere enhetstester underveis i programmeringen, og det kan tenkes at dette ville gjort det lettere å finne kilden til feil de gangene applikasjonen krasjet. Samtidig opplevede gruppen at koden hadde svært få steder hvor enhetstester egnet seg, i og med at de fleste funksjonene i koden ikke har returverdier. Selv om testingen ikke ble gjort underveis slik den er ment for, var gruppen opptatt av at det skulle gjennomføres ihvertfall én gang i løpet av prosjektet. I applikasjonen er det svært få steder hvor store mengder data håndteres, og det eneste stedet som henter inputverdier fra brukeren er i poengkortet. Derfor ble enhetstesten utført på dette fragmentet.

Android Studio ble benyttet som verktøy for enhetstesting. Enhetstesten heter ScoreBoardFragmentTest.kt, og har følgende lokasjon app; java; com.example.frisbeeapp (test); fragments;. Koden ble testet der verdiene kunne variere. Fragmentet har 18 muligheter for å legge til antall kast brukt på hvert hull. Ut ifra brukerens input, beregnes variablene total, differansen og poeng. Totalen er antall kast brukt på en hel bane. Differansen er avstanden mellom dette tallet og par, og poeng er null pluss eller minus denne differansen avhengig av om totalen er over eller under par.

Enhetstestene addition_largeNumbersIsWrong(), addition_largeNumbersIsCorrect() og calculate_maxValueTotal() er skrevet for å teste de naturlige grensene til koden, som er den mest effektive måten å oppdage feil gjennom enhetstesting. Dersom man overskridet disse grensene kan det også testes “overflow” (Sommerville, 2019, s. 267). Det finnes ingen naturlige grenser på antall ganger man kan kaste en frisbee. Grensen for maksimalt antall kast ville derfor ligge i koden, og i Kotlin sin MAX_VALUE. For å teste maksgrensen ble det gjort enhetstester med denne verdien som total. Flere av hullene ble satt til MAX_VALUE for å teste grenseverdier og overflow. Applikasjonen krasjet ikke, men utregningen ble feil, grunnet at totalen ikke kan overskride MAX_VALUE. Sannsynligheten for at noen bruker så mange kast på ett hull er ekstremt liten, men det var likevel hensiktsmessig å teste at dersom en bruker skrev inn et veldig høyt tall, så ville ikke applikasjonen krasje.

Enhetstestene addition_smallNumberIsWrong(), addition_smallNumberIsCorrect() og calculate_minValueTotal() er skrevet for å teste minimumsgrensen. Grensen for hvor få kast man kan

bruke på en runde vil i teorien ligge på 1 eller på 0 dersom man ikke kaster i det hele tatt. Koden vil sende en melding via en Toast til brukeren hvis totalen er mindre enn 0. I teorien vil det ikke være mulig å loggføre et negativt tall gjennom tastaturet, da input-typen til EditText er satt til “number”. I frisbeegolf er det ikke mulig at antall poeng er lavere enn negativ par, da dette vil forutsi at man har kastet et negativt antall ganger. Gruppen testet likevel med negative verdier for å sjekke om koden holdt dersom brukeren hadde klart å føre et negativt tall. Dette ble gjort ved å sette MIN_VALUE både som input på enkelte hull og som total. Applikasjonen krasjet ikke og beregning var korrekt, men gir ikke mening.

```
//Test edge cases and overflow
//Den egentlige verdien er -6442450944, men tallet blir ikke lavere enn MIN_VALUE = -2147483647
@Test
fun addition_smallNumbersIsWrong(){
    assertEquals(-6442450944, scoreboardObject.addTotal(listSmall))
}
@Test
fun addition_smallNumbersIsCorrect(){
    assertEquals(Int.MIN_VALUE, scoreboardObject.addTotal((listSmall)))
}
```

Figur 18: Eksempel på to av enhetstestene.

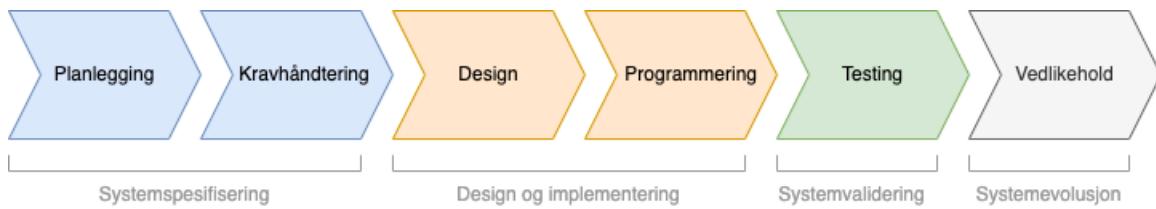
Dersom en kode tar inn numeriske verdier bør man alltid teste med 0 (Sommerville, 2019, s. 267). Gruppen lagde flere enhetstester med 0-verdier, både i brukerinputfelt og totalutregningen. Disse er addition_withZeroIsWrong(), addition_withZeroIsCorrect(), addition_withOneZeroIsWrong(), addition_withOneZeroIsCorrect() og calculate_withZero(). Det er også utarbeidet enhetstester som tester tilfeldige input. calculate_overPar() og calculate_underPar() tester en situasjon der brukers total er henholdsvis større og mindre enn par. calculate_isWrong() er en test som blir godkjent dersom verdien som legges inn er uforventet, og calculate_isCorrect() som blir godkjent dersom verdiene er det som forventes. Alle enhetstestene ble godkjent.

6.4 Integrasjonstesting

Målet med integrasjonstesting er å undersøke grensesnittet, og samspillet mellom komponenter og integrasjonene i systemenheter (Henrik Alfheim, 2022). Det ble utført en tilnærmet integrasjons-test underveis i prosjektet ved bruk av Android-emulator og en fysisk Android-enhet. Android-emulatoren ble flittig brukt gjennom hele prosjektet, mens applikasjonen ble først testet på en fysisk Android-enhet i sluttfasen. I begge tilfellene gikk testene ut på å navigere seg frem til de ulike fragmentene. Funksjonaliteten ble testet ved å trykke på alt som var mulig å trykke på. Ved å observere responsen ble det avklart om systemet gjorde det den skulle gjøre. Gruppen forsøkte å skrive integrasjonstester til deler av funksjonaliteten i applikasjonen i Android Studio. Dette var en utfordrende og tidskrevende aktivitet. På grunn av begrenset tid ble det tatt en felles avgjørelse om å nedprioritere den skriftlige testen.

7 Proseszdokumentasjon

Systemutvikling er metodisk og målrettet arbeidprosess med å utvikle og forvalte programvaresystemer av høy kvalitet innen gitte tids- og kostnadsrammer. Systemutvikling omhandler alle steg, fra tidlig planlegging frem til vedlikehold av systemet etter det er lansert og tatt i bruk (Sommerville, 2011, s. 7). Systemutviklingsprosessen som følges i dette prosjektet, illustrert i figur 19, er delt inn i fire faser: (1) systemspesifisering, (2) design og implementering, (3) systemvalidering og (4) systemevolusjon. Første fase går ut på å planlegge og spesifisere hva som skal lages innenfor hvilke rammer. Produktvisjon, modellering og kravspesifikasjon ble gjennomført i denne fasen. I andre fase skal systemspesifikasjonen realiseres. I denne fasen inngår design og programmering, også kalt implementasjon. Implementasjon er den mest tidkrevende fasen. I tredje fase skal systemet testes og ferdigstilles. Testingen utføres av et teknisk tungt team, mens i implementasjonsfasen arbeider designere og utviklere sammen. Fjerde fase består av vedlikehold og videreutvikling etter kundens og markedets behov. Prosessdokumentasjonens struktur følger fasene i figur 19, men det påpekes at fasene overlapper i virkeligheten.



Figur 19: Modellen illustrerer prosjektets syv hovedaktiviteter. Modellen er brukt som utgangspunkt for å planlegge aspekter i prosjektet. Systemutviklingsprosessen kan deles inn i fire faser og syv hovedaktiviteter. Fasene er (1) systemspesifisering, (2) design og implementering, (3) systemvalidering og (4) systemevolusjon. Hovedaktivitetene er planlegging, kravhåndtering, design, programmering, testing og vedlikehold.

7.1 Planlegging

7.1.1 Utviklingsmetode

I dette prosjektet er det benyttet smidige systemutviklingsmetoder. Alle smidige utviklingsmetoder baserer seg på *Manifestet for smidig utvikling* som kan oppsummeres i 12 prinsipper, presentert i figur 20 (Agile Manifesto, 2022). Prinsipp 1 var irrelevant, fordi gruppen ikke hadde en produkteier eller kunde. Prinsipp 2 ble fulgt ved å bygge opp en forventning om at detaljene av funksjonaliteten kom til å forandre seg underveis. Gruppen hadde en forventning om at programvaren måtte tilpasses, ettersom det ble tilegnet dypere forståelse og ny læring. Gruppen ble enige om å være åpne for kravendringer, selv sent i utviklingen. Derfor ble kravspesifikasjonen redigert ved flere anledninger. Videre ble prinsipp 3 om å opprettholde korte arbeidsiterasjoner fulgt ved å ha tre statusmøter i uken. Gruppens tre møter i uken førte også til muligheten for å jobbe sammen og kommunisere ansikt-til-ansikt (Prinsipp 4 og 6). Dette førte videre til at gruppen opparbeidet tillit til hverandre og motiverte hverandre (Prinsipp 5). For å sikre fungerende programvare, ble må-kravene i kravspesifikasjonen holdt så minimalt som mulig (Prinsipp 7 og 10). Ved å være fleksible og ikke følge faste tidsbokser,

klarte gruppen å opprettholdt en balanse mellom kvalitet og et jevnt tempo (Prinsipp 8). Ved å følge kravspesifikasjonen og brukerhistoriene og teste applikasjonen i emulator ofte, ble det rettet oppmerksomhet til teknisk kvalitet og godt design (Prinsipp 9). Prinsipp 11 var delvis irrelevant, grunnet få formelle rammer og at dette ikke er et reellt prosjekt delegert av en organisasjon, men et selvstendig studentprosjekt. Ved hjelp av ukentlig retrospektmøter justerte gruppen aferden sin for å optimalisere og effektivisere arbeidsprosessene (Prinsipp 12). (Agile Manifesto, 2022)



Figur 20: De 12 smidige prinsippene basert på *Manifestet for smidig utvikling* (Agile Manifesto, 2022).

Gruppen har arbeidet etter den smidig utviklingsmetoden Scrumban, en hybrid av Scrum og Kanban. Gruppens arbeidsmetode lignet Kanban, fordi det var fritt å legge til flere arbeidsoppgaver (items) i en sprint dersom det er ledig kapasitet, det var valgfritt å prioritere arbeidsoppgaver i produktkøen (product backlog) og det ble benyttet frie roller. Produktkøen er en gjøremål-liste med arbeidsoppgaver som kan være feil, funksjonalitet og produktforbedringer (Sommerville, 2019, s. 29). Samtidig var det et ønske om å se på ukene som sprinter, for å sikre god flyt og tidsfrister, som ligner på Scrum. Derfor jobbet gruppen i en ukentlig syklus, med planleggingsmøte på mandager, statusmøte og felles arbeidsøkt på onsdager og retrospektivtmøte på fredager.

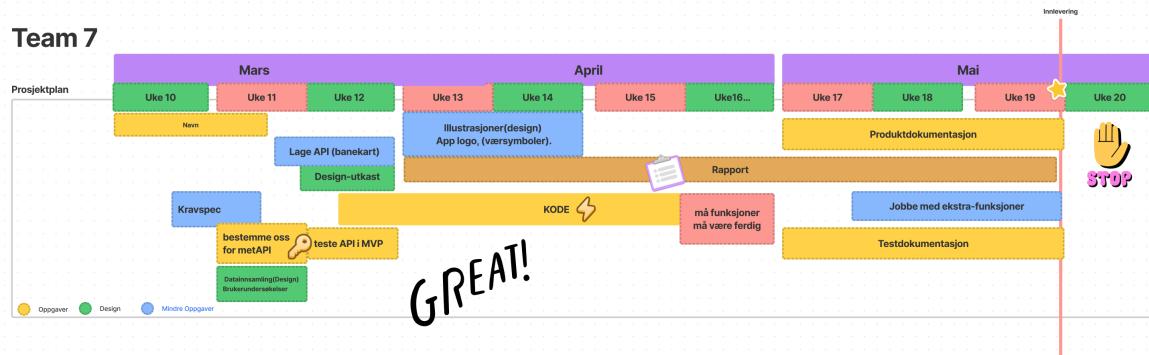
Planleggingsmøtene gikk ut på å diskutere fremgangsmetoder og fordele gjøremål. I arbeidsøktene hadde medlemmer i gruppen mulighet til å parprogrammere, gruppeprogrammere og arbeide sammen. Alle møtene, både mandag, onsdag og fredag, startet med et statusmøte der gruppemedlemmene på rundgang fortalte status på sitt arbeid. Gruppen benyttet Trello til denne gjennomgangen. Trello ble også benyttet for å holde oversikt over både produktkø og andre gjøremål under hele prosjektets gang. Det ble opprettet et tavle for hver av ukene, der oppgaver på agendaen fritt kunne flyttes mellom

de ulike ukene basert på kapasitet. Referat og møteleder rullerte for hver uke. I retrospektmøtene diskuterte gruppen prosessforbedring. Hvert gruppemedlem delte sine meninger om arbeidets kvalitet, og gruppen identifiserte svakheter, styrker, muligheter og trusler. Retrospektmøtet ble avsluttet ved å notere ned endringer i arbeidsprosessen i referatet, som alle hadde med seg videre i arbeidet.

7.1.2 Verktøy

Verktøyene som ble benyttet for å støtte arbeidet var Trello, Teams, Facebook Messenger, Overleaf, GitHub, Miro, Figma og Google Disk. Grunnet varierte erfaringsgrunnlag med verktøyene, ble det opprettet en ansvarlig for hvert verktøy. Ved prosjektets oppstart ble Teams valgt som kommunikasjonskanal, av tre grunner. Først, for å skille studieliv fra privatliv. For det andre, at meldinger forsvinner fort i Messenger, og for det tredje, at Teams inneholder en nyttig kodeformateringsfunksjon. Likevel ble Facebook Messenger benyttet hyppigere enn Teams. Det kan tenkes at dette hadde med gruppemedlemmenes vane ved å bruke Facebook til ”alt”. Overleaf ble benyttet for rapportskriving. Verktøyet muliggjør effektiv samskriving og ryddig kildeføring. GitHub ble benyttet til å samarbeide om kode. GitHub er integrert i Android Studio og fungerte dermed utmerket til å kontinuerlig oppdatert kode mellom medlemmene i gruppen og føre historikk. Dermed hadde gruppen oversikt over tidligere og daværende kode. Miro ble benyttet for å kommunisere design og visualisere idéer. Google Disk fungerte som et felles møtepunkt hvor alle filer, bilder, design, møtereferater og arbeidslogger ble delt. Figma ble benyttet til å utvikle designet. Verktøyene ble benyttet med den hensikt å støtte opp om samhandling og effektivitet gjennom systemutviklingsprosessen.

Det ble utviklet en overordnet prosjektplan i forbindelse med obligatorisk oppgave 3 tidligere i emnet, illustrert i figur 21. Gruppen benyttet prosjektplanen til å vurdere fremgangen i prosjektet. Det vil si at den fungerte mer veiledende enn en fastslått plan.



Figur 21: Overordnet prosjektplan

7.1.3 Ideutvikling og produktvisjon

Ved prosjektets oppstart ble det utviklet to applikasjonsidéer. Den første idéen var en solbade-applikasjon med UV-data som hovedfunksjon. Andre ide var en frisbeegolf-applikasjon med mål om å lage en mer brukervennlig applikasjon enn de avanserte frisbeegolf-applikasjonene, som for eksempel Udisc og Metrix. Etter avstemning landet valget på å utvikle en frisbeegolf-applikasjon.

Det var et ønske om å lage en original applikasjon, som potensielt kunne benyttes av gruppen i ettertid. Derfor ble det viktig at applikasjonen skulle ha andre funksjoner enn det som allerede eksisterte på applikasjonsmarkedet. Applikasjonen ble navngitt *FROLF*. Ordet frolf er et slengord for frisbeegolf, og brukes gjerne om en uformell form for frisbeegolf (FROLF, 2022). Figur 22 gir et inntrykk av designet i *FROLF*.



Figur 22: Skisse av introsiden til frisbeegolf-applikasjonen *FROLF*

I planleggingsfasen ble det utarbeidet en produktvisjon med formål om å veilede arbeidet fremover, som også ble presentert i 1 *Innledning*. *FROLF* er en frisbeegolf-applikasjon for nye spillere som mangler en enkel applikasjon til å starte med, som tilbyr rask oversikt over baner, værdata, regler og poengkort. I motsetning til eksisterende frisbeegolf-applikasjoner, tilbyr vårt produkt et enkelt brukergrensesnitt med kun de nødvendige funksjonene for nybegynnere. Dette var et godt grunnlag før gruppen gikk inn i fasen med å formulere krav.

7.2 Kravhåndtering

Kravhåndtering er viktig for oppdragsgiver, planlegging, arkitektur, og for å støtte videreutvikling og vedlikehold. Prosjektets kravhåndtering består av use case modellering, sekvensdiagram, klassediagram og kravspesifikasjon, og beskrives grundig i del 3 *Kravspesifikasjon og modellering*. Kravene ble utviklet på bakgrunn av kravinnsamlingsmetodene kvantitative spørreundersøkelser, dybdeintervju med potensiell bruker og idemyldring i form av funksjonalitetsworkshop og designworkshop. Kvantitative spørreundersøkelser og dybdeintervju er utdypet i del 4 *Designprosess*.

Funksjonalitetsworkshoppen bestod av felles og individuell idemyldring, kategorisering av idéene og til slutt utvelgelse og rangering av ønsket funksjonalitet. Workshoppen startet med felles idemyldring, der idéer ble skrevet ned på post-it-lapper og plassert på en tavle. Videre gikk gruppen over til det som viste seg å være den mest effektive metoden, den individuelle ideskrivingen. Dette gikk ut på å skrive så mange forslag som mulig på post-it-lapper innen en bestemt tid. På denne måten hadde gruppen mange mulige idéer å arbeide med. Kvantitet stod foran kvalitet på dette stadiet. Videre ble et utvalg av lappene kategorisert etter tre prioriteter: *må*, *bør* og *kan*. Resultatet av workshoppen ble første utkast av prosjektets kravspesifikasjon, presentert i vedlegg I. Design-workshoppen gikk ut på

å diskutere arkitektur og applikasjonens layout. Med produktvisjonen og kravspesifikasjonen som grunnlag, ble designet visualisert på en tavle. Diskusjoner om arkitektur ga et grunnlag for å starte med modelleringen.

7.2.1 Kravspesifikasjon

Kravspesifikasjonen ble revidert etter vurdering av den gjennomførte brukerundersøkelsen. Revidert kravspesifikasjon er beskrevet i tabell 2 i del 3 *Kravspesifikasjon og modellering*. I den nye kravspesifikasjonen er også prioriteringen av enkelte krav endret på, basert på identifiserte brukerbehov. Innen prosjektets slutt ble alle åtte må-krav og tre bør-krav implementert.

Må-kravene er essensielle for at applikasjonen skal kunne fungere og at applikasjonen skal kunne oppnå produktvisjonen. Derfor ble alle må-krav implementert før gruppen bega seg ut på Kan-kravene. Det var viktig i prosjektet at de elementene og funksjonene som ble implementert, fungerte som de skulle dersom de ble sjekket av i kravlisten.

Mange av kan-kravene hadde gruppen ambisjoner om å implementere, samtidig som de var bevisste at det potensielt ble lite tid til det. Kan-kravene er derfor hovedsaklig beholdt videreutvikling. Dersom det skulle vise seg at gruppen implementerte må- og bør-kravene innen god tid, kunne gruppen implementert nye funksjoner fra kan-kravene. En utfordring med å implementere helt ny funksjonalitet mot prosjektets sluttfase, er at det fører til høy risiko for at produktet leveres med lite grundig testing av siste funksjonalitet og dermed potensielle feil. Derfor satte gruppen et sluttidspunkt for når siste funksjonalitet skulle påstartes.

To eksempler på krav som ble forsøkt implementert er krav 2.3 *Lagre poeng for hver runde* og 2.4 *Introside til applikasjonen (når man åpner applikasjonen første gang)*. Mot slutten av prosjektet utforsket gruppen å implementere en Room-database for å oppnå krav 2.3, da gruppelærere mente at dette var overkommelig. Databasen skulle benyttes til å lagre runder i historikk. Grunnet bekymringer knyttet til å implementere ny funksjonalitet nær innleveringsfristen for prosjektet, ble databasen nedprioritert til fordel for å teknisk testing av applikasjonen. Krav 2.4 *Introside til applikasjonen (når man åpner applikasjonen første gang)* kunne implementeres ved å legge inn en “Splash screen” på to forskjellige måter. Første løsning er å endre tema i applikasjonen ved oppstart og har ingen effekt på kjøretiden på applikasjonen. Den andre løsningen er å opprette en ny aktivitet og benytte en “sleep”-metode. Den nye aktiviteten vil dermed kjøre før hovedaktiviteten kjøres. Ulempen med andre løsning er at resten av applikasjonen ikke blir lastet inn før etter introsiden er vist. Til slutt viste seg at “Splash Screen” ikke støttes av API-nivået applikasjonen var laget i (Android for Developers, 2022i).

Tabell 2: Følgende funksjonell krav ble tatt hensyn til under utviklingsprosessen av applikasjonen. Kravene beskriver hva systemet skal kunne gjøre. Prioritert rekkefølge fra høy-middels-lav er henholdsvis må-bør-kan. Kravene med hake betyr at kravene har blitt implementert i applikasjonen.

Prioritet	Nr	Kravspesifikasjon
Må		
✓	1.0	Oversikt over banene (recycler)
✓	1.1	Liste over alle baner i Oslo, med tilhørende banekart
✓	1.2	Vise lokal værmelding for valgt område (temperatur, vindstryke, vindretning, nedbør og skydekke)
✓	1.3	Vise en beskrivelse for hver bane (Kollektiv, vanskelighetsgrad, tees og WC)
✓	1.4	Regelside med de viktigste reglene
✓	1.5	Føre poeng/kast til de ulike hullene
✓	1.6	Følge utvalgte WCAG-prinsipper
✓	1.7	Introside - Logo når bruker åpner applikasjon
✓	1.8	Scoreboard som tillater å bytte fragment uten at dataen forsvinner
Bør		
✓	2.0	Vise brukerens posisjon i kartet
✓	2.1	Nullstill zoom inn/ut på kart
✓	2.2	Utvidet regelside med kategorier (f.eks. folkeskikk og mando)
	2.3	Lagre poeng for hver runde
	2.4	Introside til applikasjonen (når man åpner applikasjonen første gang)
	2.5	Vise vanskelighetsgrad på de ulike banene
	2.6	Vise rating på de ulike banene
	2.7	Vise spilletid på hver bane
	2.8	Vise soloppgang og solnedgang
	2.9	Nedlastningsskjerm med snurrende frisbee
Kan		
	3.0	Tilby dark mode og light mode
	3.1	Tilby søk i fritekst bland baner
	3.2	Opprette profil
	3.3	Scoreboard med plass til flere personer
	3.4	Lagre favorittbane
	3.5	Slette favorittbane
	3.6	Instillinger-side (eks. clear history, dark mode, profilinstillinger)
	3.7	Legge igjen egen review på banen
	3.8	Vise pågang på banen
	3.9	Vise parkeringsinformasjon
	3.10	Vise om banen har ly for vinden
	3.11	Sortere listen med baner etter avstand, pågang, rating, kollektiv, vær
	3.12	Legge til flere baner i Norge
	3.13	Vise avstand til banene fra "min posisjon"
	3.14	Tilby satellitt/terreng-kart
	3.15	Måle avstand på kast
	3.16	Tilby banekart i Google Earth (polygon)
	3.17	Legge til en funksjon som viser UV-varsel for en gitt lokasjon og bane
	3.18	Vise nærmeste kiosk/badevann i forhold til banen
	3.19	Anbefale pakkeliste basert på værforhold
	3.20	Tilby videoer med teknikker og "tips og triks"

7.3 Design

I designfasen transformeres kravene til en helhetlig designspesifisering, og er en nødvendig forløper til programmeringsfasen. I denne fasen ble applikasjonens arkitektur modellert og brukergrensesnittet skissert. Designprosessen har sikret at kravene blir dekt og at applikasjonen er brukervennlig. Designet har vært essensielt for å effektivt implementere programmet. Utviklerne unngår omskriving av kodedeler siden de har oppbygget en forståelse av programmets helhet. Brukervennlighet innebærer at bruker enkelt utfører grunnleggende oppgaver første gang de møter designet, at designet er lett å huske til neste gang, at bruker effektivt kan utføre oppgaver i applikasjonen og er tilfreds med designet. Applikasjonen er testet på brukere, og så endret for å øke brukervennligheten. Brukerinvolveringen sikrer at arkitekturen er intuitivt oppbygd. Designfasen er beskrevet i detalj i del 4 *Designprosess*.

7.4 Programmering

I følge et studie i programvareutvikling med 300 profesjonelle utviklere som deltakere, øker parprogrammering kodekvaliteten for juniorprogrammerere, uavhengig av kompleksiteten av problemet som skal løses (Arisholm mfl., 2007). Derfor kodet gruppen mye i par, i tillegg til å kode en del med hele gruppen tilstede. GitHub ble benyttet for å samarbeide, håndtere og overvåke endringer. GitHub er et distribuert versjonshåndteringsystem, som vil si at brukerne har en kopi av koden i en lokal branch, istedet for å stadig hente ut de nyeste filene. Git muliggjør eksperimentering med prosjektet i et separat miljø. På prosjektets sentrale master, kunne alle se hva som er blitt gjort. ved å jobbe på lokale brancher kunne hadde gruppen muligheten til å tilbakestille eventuelle feil før det ble pushet og pullet til master.

7.4.1 Fragmenter

Gruppen bestemte seg for å først få på plass kritiske deler som fragmenter, kart, datahåndtering, navigasjonsbar og RecyclerView. Det ble diskutert om gruppen skulle benytte JetPack Compose eller Empty Activity. Gruppen kom fram til å benytte Empty Activity, grunnet tidligere erfaring med det fra obligatoriske oppgaver i emnet. Applikasjonens arkitektur er bygget opp av én aktivitet og flere fragmenter som er koblet sammen. Fordelen med å bruke fragmenter er at et fragment representerer en gjenbrukbar del av applikasjonens brukergrensesnitt (Android for Developers, 2022c). Ved å ta i bruk fragmenter ville det også være mulig å bruke Navigation Component. Fordelen med Navigation Component er at man får et oversiktlig kart over fragmentene og deres kobling med hverandre, samt at det er lettere å bytte mellom fragmenter. En annen fordel ved å bruke fragmenter og Navigation Component, er at den også støtter “multiple backstacks”. Dette vil si at hvis man går fra et fragment til et annet, vil tilstanden lagres og når brukeren da trykker tilbake til det fragmentet brukeren kom fra, vil tilstanden være slik det var. Når brukeren trykker seg videre i et fragment, vil også tilstanden lagres på en “stack” slik at når brukeren trykker seg tilbake, så slipper fragmentet å lages på nytt hver gang.

En utfordring ved å bruke fragmenter, er at koden må settes opp på en annen måte enn oppbygging med aktiviteter. Fragmenter har flere “lifecycle”-metoder hvor man må legge de ulike delene av koden

riktig for at applikasjonen skal oppføre seg på ønsket måte. I motsetning til activity, der onCreate-metoden benyttes mest, vil det i Fragments veksles mellom metodene onCreate, onCreateView og onViewCreated. Fragmenter kan ikke leve alene, men må være vert for en aktivitet eller et annet fragment. Fragmentets visningshierarki blir en del av, eller knytter seg til, vertens visningshierarki (Android for Developers, 2022c). Dette gjorde det også vanskeligere når Google Maps skulle implementeres, da brukers egen posisjon ikke ville vise under første oppstart av applikasjonen. Årsaken til dette er at kartet blir generert samtidig som bruker får opp vindu med spørsmål om tillatelse. Dette er kun første gang applikasjonen starter opp og det aldri har blitt gitt tillatelse om posisjon.

7.4.2 Kart

For at applikasjonen skulle få opp kart og de ulike posisjonene til banene på kartet, måtte en på gruppen opprette en bruker hos Google. Brukeren ga tilgang til en API-nøkkel som applikasjonen måtte ha for å kunne vise Google sine kart. Til tross for at API-nøkkelen er begrenset, hadde ikke gruppen behov for ytterligere funksjonalitet. For å programmere i Android Studio med kart, måtte alle gruppemedlemmene legge inn nøkkelen manuelt i “local properties”-filen på hver sin maskin. Dette brukte gruppen litt tid på i starten. Det var utfordrende å sette opp kart i applikasjonen, da det kunne gjøres på flere måter. Valget falt på å benytte et callback som blir kalt når kartet er klart til bruk. I dette callbacket ligger alt som blir opprettet i kartet, altså egen posisjon, zoom-knapper og alle markører som viser hvor de ulike banene ligger på kartet. Baneinformasjonen blir hentet fra en delt viewmodel og lagt inn ved oppstart.

7.4.3 Banedata

Ved prosjektets oppstart var et usikkert moment å få tilgang til API med baneinformasjon. Derfor ble dette utredet i planleggingsfasen av systemutviklingsprosessen. Plan A gikk ut på å få tilgang til Disc Golf Course Review eller Disc Golf Metrix, som er offentlige API-er som brukes av eksisterende frisbeegolf-applikasjoner og tilbyr utfyllende informasjon om alle verdens frisbeegolfbaner. Plan B gikk ut på å utvikle et eget API. For å sikre at plan B var gjennomførbar, rådførte gruppen seg med en av fjorårets grupper i emnet som hadde utviklet eget API. Fjorårets gruppe stilte seg bekymret til å utvikle et eget API, grunnet at skrivefeil hadde forårsaket mye ekstra arbeid. Etter oppklaring viste det seg at gruppen håndterte en betraktelig større mengde data og i tillegg lagret API-et lokalt i applikasjonen. For å være på den sikre siden, ønsket gruppen å utvikle det egne API-et snarest for å få testet om det fungerte tidlig nok til å kunne endre case. Etter prosjektets gang viste det seg at selv om gruppens API var lite, så oppstod det skrivefeil som gjorde at en del timer gikk tapt. Det er sikkere å benytte seg av API-er som er velbrukt, fordi da er de fleste feil allerede oppdaget.

Gruppen kontaktet UDisc og Disc Golf Metrix, de to største aktørene innen frisbeegolf-applikasjoner, for å undersøke hvilke API-er de benyttet og for å høre om de hadde nyttige tips. UDisc bruker en egen database, hvor frisbeegolfklubbene selv må kontakte UDisc for å legge inn deres bane. UDisc fortalte videre at de er i prosessen med å utvikle eget API, men denne ville ikke være offentlig tilgjengelig på en god stund. Disc Golf Metrix bruker allerede et eget API. Dette API-et har svært lite informasjon om hver bane, men inneholder til gjengjeld de fleste frisbeegolfbanene i verden. Plan A

ble forkastet, på grunnlag av at begge de offentlige API-ene viste seg å ha en del strenge krav som må oppfylles for å få tak i deres API-nøkkelen. Dette var blant annet å vise til en beta-applikasjon, og vise til deres logo alle steder informasjon fra API-et blir brukt. Med tanke på tidsperspektivet til prosjektet, tok gruppen en beslutning om å gå til plan B. Plan B ville kreve ekstra innsats og økte sjansen for at potensielle skrivefeil kunne forårsake problemer. Likevel konkluderer gruppen med at det ville gi flere fordeler senere i prosjektet. Etter diskusjon med gruppelærere ble det bestemt å utvikle ett eget API i form av en JSON-fil. Det egenutviklede API-et inneholder en del mer informasjon om de ulike banene enn Disc Golf Metrix. Denne informasjonen er hentet fra de ulike frisbeegolfklubbene sine hjemmesider. Gruppen undersøkte muligheten for å laste opp filen til UiO-server, men endte med å laste opp JSON-filen på nettsiden <https://www.npoint.io/>. nPoint vil være en midlertidig løsning, da den ikke klarer å håndtere større API, men det vil fungere optimalt for prosjektet.

7.4.4 Regelside

Regelsiden er en av funksjonene som applikasjonen tilbyr, da det er nyttig å sjekke og oppklare regler under en frisbeegolfrunde. Derfor har en rekke undersøkelser blitt gjort for å få skrevet ned de viktigste og vanligste reglene fra PDGA (Professional Disc Golf Association) sin regelbok. Reglene er delt opp etter temaene superenkla regler, etikette, hazard (forbudte områder), kasterekkefølge, mando (påbudt), markering av disk, sikkerhetsregler og fulle regler. Reglene ligger i hver sin .txt-fil i app; java; res; raw. Det gir muligheten til å endre eller legge til mer detaljer i reglene dersom det er behov for det. Dette kom også med utfordringer, da .txt-filenes oppbygning er veldig enkel. For eksempel håndteres ikke linjeskift, så regeltekstene ble uoversiktlig når de vistes i applikasjonen. En løsning på dette ville være å legge teksten inn i strings.xml-filen, men igjen vil dette se veldig rotete og ustukturert ut i koden. Gruppen kom derfor frem til et kompromiss, hvor de superenkla reglene ble lagt inn i strings.xml-filen mens resterende regler ble gjort om til avsnitt og ligger fortsatt som .txt-filer. Grunnen til dette er at de superenkla reglene inneholder en kort forklaring av de ulike reglene. Det var derfor viktig å beholde et punktvist format. Videre ønsket gruppen å sette opp reglene i et DropDownView, som forutsetter at bruker trykker på en overskrift for å vise ønsket regelkategori. Etter litt prøving og feiling ble denne idéen forkastet, med grunnlag for at koden ble for vanskelig å implementere og gruppen fant lite informasjon om hvordan DropDownView fungerer. Gruppen valgte heller å benytte seg av en spinner, da alle hadde kjennskap til funksjonen gjennom de obligatoriske innleveringene. Spinneren ligner på DropDownView, men gjør at bruker må trykke på spinneren for å se alle kategoriene og kan kun se en kategori om gangen.

7.4.5 Værdata

Retrofit ble benyttet for å kalle på Meteorologisk institutt sitt API Nowcast 2.0. Gruppen valgte Nowcast fremfor Locationforecast, fordi Nowcast dekker ønsket funksjonalitet. Dette API-et dekker værmelding for et spesifisert sted med en ni-dagers periode. Grunnen til at Locationforecast ikke ble valgt var fordi det inneholder overflødig data som kan påvirke ytelsen av applikasjonen. En manual for Retrofit ble fulgt for å implementere API-kallet. Gruppen opplevde det som relativt enkelt å sette opp API-kallet, men utfordrende å trekke ut spesifikke verdier. Etter at gruppen løste det, viste seg at dette var mye enklere enn antatt. Ved å oppgi breddegrad og lengdegrad for et spesifisert

sted gir API-et værmelding for stedet (MeterologiskInstitutt, 2022). Dataen som hentes ut er basert på kravspesifikasjonen. Værdata ble innlemmet i course-objektene. Course-objektet har objektet CourseWeather som parameter. Derfor har koden kun én viewmodel. CourseWeather inneholder de nevnte parameterne som skal hentes fra NowCast.

Alle værsymboler kunne lastes ned fra dokumentasjonen fra Metrologisk institutt, men gruppen valgte å designe fire værsymboler for å passe resten av designet i applikasjonen. Værsymbolene ble lagret lokalt i drawable-mappen. I dokumentasjonen fra Metrologisk institutt var det derimot over 50 ulike symbolkoder, med tilhørende værsymboler. Derfor ble det laget en metode som kategoriserte symbolkodene etter fire kategorier: fair day, heavyrain, cloudy og clear sky. Det ble konkludert at å lagre fire symboler lokalt, istedenfor å hente fra nett, ikke ville påvirke kjøretiden til applikasjonen.

Det egenutviklede API-et ble endret en del underveis i prosjektet. Skrivefeil i det egenutviklede API-et førte til forsinkelser under arbeidet med å presentere værdata sammen med baneinformasjonen. Gruppen endret koordinatene i bane-API til fire desimaler, siden det stod i dokumentasjonens fra Metrologisk institutt “Terms of Service”-side at koordinater skal begrenses til fire desimaler. Kall med fem desimaler og mer vil returnere feilmeldingen 403 *Forbidden*. En annen endring som ble gjort er å lagre koordinatene som Double istedenfor String, fordi denne variabeltypen trengtes til kallet.

7.5 Testing

Testing utføres kontinuerlig i programmeringfasen, ved ferdigstilling og ut systemets levetid. Teknisk testing er beskrevet i del 6 *Testdokumentasjon*, og brukertesting er beskrevet i del 4 *Designprosess*. Etter testfasen ville applikasjonen ha blitt lagt ut. Google Play har en rekke krav til applikasjonene som skal legges ut på deres plattform, blant annet krav til API-nivå. Fra august 2021 krevde Google Play at alle nye applikasjoner skulle legges ut med API-nivå 30 eller høyere. Dette er ett nivå høyere enn API-nivået til *FROLF*. For å endre API-nivå må det gjøres en del spesifikke endringer som beskrives i Android-dokumentasjonen. Enkelte metoder kan være avviklet (deprecated). I tillegg kan det ha kommet ny forbedret funksjonalitet og metoder som er ønskelig å benytte (Android for Developers, 2022g). Det ble heller prioritert å sikre applikasjonen kjørbarhet og potensielt tilpasse API-nivået senere. Applikasjonen kan lastes ned uten Google Play slik som beskrevet i brukerdokumentasjonen. Valg av API-nivå er beskrevet i produktdokumentasjonen.

Store deler av koden er utviklet etter beste praksis ved å følge guider fra Android Developer. Eksempler på dette er Google Maps og bruk av HTTP-bibliotek. Å følge konvensjoner kan minimere teknisk gjeld. Til neste prosjekt kunne gruppen satt seg inn i testteorien før programmeringsfasen, for å tilegne seg et overblikk av testingen underveis. Dermed kunne gruppen brukt mindre tid på testingen, og potensielt fått tid til grundig integrasjonstesting.

7.6 Vedlikehold og videreutvikling

Vedlikeholdsfasen er den siste fasen i systemutviklingsprosessen. Etter at systemet er implementert, kan vedlikeholdsarbeidet begynne. Produktdokumentasjonen fungerer som en overlevering dersom

applikasjonen skal videreutvikles og vedlikeholdes av andre utviklere. Vedlikehold av applikasjonen er viktig for at den skal fortsette å fungere. Android oppdateres ofte og det antas at funksjoner som fungerer i dag, ikke nødvendigvis vil fungere i fremtiden. Funksjoner kan bli avviklet, altså at de ikke lenger er sett på som beste praksis av ulike årsaker. I fremtiden kan det hende at funksjoner som er brukt under utvikling av applikasjonen har blitt avviklet, og da bør disse bli oppdatert til det som er anbefalt av Android.

For at koden skal være forståelig for andre utviklere, er det lagt vekt på å skrive lett forståelig kode. Alt av kode er kommentert og det er benyttet beskrivende variabelnavn og metodenavn, for å kommunisere kodens hensikt. Det er mange funksjonaliteter som gruppen ikke har rukket å implementere, men som kan implementeres med mer tid. Dersom applikasjonen skal videreutvikles, er det mange kan-krav som kan implementeres.

8 Refleksjoner

8.1 Personlig refleksjon

Celine

Jeg gikk inn i dette prosjektet med svært lite kjennskap til hva det innebar. Som designstudent føltes det uvirkelig at vi skulle lage en applikasjon helt alene, og jeg var usikker på hva jeg kunne bidra med. Etter et par møter med gruppen skjønte jeg raskt at vi utfylte hverandre godt, og at det var plass til alle. Koding i Android Studio var noe jeg gruet meg veldig til. Heldigvis var gruppen innstilt på at alle skulle skrive kode, noe som gjorde at jeg føler meg sikrere på programmering nå. Gjennom prosjektet har jeg slitt med sykdom, og dette har påvirket både energinivå og hukommelse. Fra tidligere emner er jeg vant til å ha oversikt og kontroll. For å prioritere helsa var jeg nødt til å legge fra meg dette behovet dette semesteret. Heldigvis har jeg hatt gode medstudenter i prosjektet med enda mer struktur enn jeg har til vanlig, og det har vært veldig betryggende. Jeg er stolt av det vi har laget, og synes alle gruppemedlemmene har lagt inn en ordentlig innsats. Jeg er også veldig fornøyd med at gruppen har hatt et godt sosialt miljø, og at vi endelig har hatt et prosjektarbeid på campus.

Daniel

Etter å ha jobbet med prosjektet i 10 uker, har jeg både lært mye og blitt kjent med nye studenter. Jeg synes at prosjektarbeidet har vært både gøy og utfordrende. Det har også vært veldig lærerikt med tanke på hvordan det vil være å jobbe når man kommer seg ut i arbeidslivet etter endt utdanning. Min største interesse inn mot gruppearbeidet var definitivt selve programmeringsdelen, og der føler jeg også at jeg har fått lære mye nytt og utviklet mine egne ferdigheter. Det som har vært mest utfordrende, var det å komme inn i en gruppe hvor alle hadde sitt studieløp med ulike emner og timeplaner, samt det å sette seg inn i alt. Selv med utfordringene, syns jeg vi kom godt ut av de 10 ukene og sitter igjen med noe vi kan være stolte av. Jeg tror også at med litt lengre tid, ville vi klart å få en enda bedre applikasjon med mer funksjonalitet.

Eirin

I løpet av studietiden har jeg erfart mye gruppearbeid. Å samarbeide med andre og jobbe i grupper er noe jeg alltid har vært trygg med, og jeg var derfor veldig spent på hvordan dette prosjektet ville være. Allerede fra første dag har jeg møtt blide medstudenter, og jeg ser tilbake på tiden som en positiv opplevelse. Til tider har det vært utfordrende at vi har hatt ulike timeplaner, og samtlige av oss har hatt eksamener underveis i semesteret. Engasjementet i gruppa rundt prosjektet har også vært i bølgdaler. Dette har gjort det vanskelig å være like motivert for hver uke, og har nok også preget produktiviteten. Jeg synes likevel vi har vært flinke til å tilpasse oss hverandre etter omstendighetene. Gruppearbeidet har vært tidskrevende. Vi har diskutert mye og tatt oss god tid til å sette oss inn i ting. Under prosjektet har vi tilnærmet oss en Kanban-arbeidsmetodikk, til et fremtidig prosjekt kunne det vært interessant å prøve ut Scrum.

Nora

IN2000-prosjektet har vært svært givende. Da jeg søkte opptak til IFI hadde jeg lest om dette emnet og syns det var kult at man utviklet en applikasjon, siden jeg liker å skape og se konkrete resultater av arbeidet. Dette emnet har jeg derfor sett frem til lenge. Jeg er fan av dyptdykkende prosjekter, og at emnet telte 20 poeng la opp til å fordype seg. Ved emnets oppstart var jeg bekymret for at jeg skulle havne på gruppe med uengasjerte studenter. Derfor inntok jeg gruppearbeidet med en nogenlunde skeptisk tankegang, men innså etterhvert at jeg må ha tillit til de andre gruppemedlemmene, selv om jeg ikke kjenner de så godt, for at gruppearbeidet skal fungere mer optimalt. I prosjektet har jeg lært mye nytt om design og fått føle på hvordan en systemutviklingsprosess kan foregå. Det har vært fint å få bekreftet at koding ikke trenger å være så krevende som IN1010, og at det finnes moderne språk som Kotlin. Jeg har blitt betraktelig bedre til å “google” frem informasjon, og er fascinert over hvor fort funksjoner blir “deprecated”. Det er givende å vite at IT-feltet er i konstant utvikling og ser ut til å bli mer og mer aktuelt i lang tid fremover. Gjennom digøk-retningen har vi hatt flere store gruppeprosjekter og skrevet rapporter på størrelse med denne, så Eirin og jeg kunne bidra med å løfte rapporten. Det har også vært fint å få brukt kunnskap fra IN1030 og forstått viktigheten av dette emnet.

Oscar

Jeg syntes dette prosjektet har vært svært lærerikt og interessant. Det har gitt meg ett helt nytt innblikk i hvordan utviklingsprosessen vil foregå videre i arbeidslivet. Jeg har også lært mye mer om de andre utdanningsprogrammene her på instituttet, og faktisk sett hvor forskjellene ligger i kompetansen til de ulike linjene. Personlig er jeg veldig interessert i case-valget gruppen tok for seg, da jeg har spilt en del frisbeegolf, og jeg mener at applikasjonen vår faktisk kan konkurrere mot de andre frisbeegolf-applikasjonene på markedet. Selv om vi tidlig i prosjektet ble enige om å ikke ha faste oppgaver, føler jeg likevel at de ulike medlemmene i gruppen har fått ansvarsområder som passer deres bakgrunn, hvorav jeg har fått noe mer ansvar for programmeringsdelen. Jeg føler likevel at jeg har fått et godt innblikk i alle delene i prosjektet, ettersom vi har hatt tre møter i uka der vi følger hverandre opp. Det som har vært mest utfordrende er å finne tidsrom som passer alle sammen,

da de ulike studierettingene kommer med forskjellige timeplaner. Spesielt ble det vanskelig da jeg hadde hjemmeeksamen i IN2140, hvor dette prosjektet ble noe nedprioritert. Likevel syntes jeg at gruppen har respektert dette, og at vi står igjen med et prosjekt vi alle kan vise videre til i fremtiden.

Peder

Jeg syntes det var gøy å kunne ha et prosjekt med studenter som ikke bare var fra samme linje på IFI. Det var også veldig gøy å prøve oss på smidige prinsipper slik at vi kunne se fordelene med dem. Tidligere har denne måten å utvikle på virket litt rart, men etter å ha snakket med andre folk i arbeidslivet, har jeg sett at det kan være en veldig god erfaring for framtiden. I starten stemte jeg på et annet case. Men jeg føler mulighetene løsningen vår tillot, var mye større bedre. Jeg syntes det var vanskelig å vite hvor mange funksjoner som var nok. Jeg er litt splittet mellom kvalitet og kvantitet. Spesielt når alt er nytt og man må lære seg hvordan ting fungerer. Da blir det vanskelig å implementere nye funksjoner spesielt når man er usikker på hvor lang tid det tar å gjennomføre disse. Med programmering kan du sitte mange timer uten å vite svaret. Da kan det være vanskelig å gi et godt svar på om det tar lang tid å bli ferdig. De andre gruppemedlemmene har vært veldig flinke og jeg føler jeg kunne tatt initiativ til mer, både koding og skriving i starten av prosessen. Både GitHub og Overleaf var nye programmer for meg, som jeg tror jeg var litt redd for å begynne å bruke i starten av prosjektet. Jeg ser nå hvor gode disse hjelpemiddlene er og vil ta med dette videre.

8.2 Retrospektiv

Gruppearbeid og arbeidsfordeling

Gruppen har reflektert rundt hvordan gruppearbeidet og arbeidsfordelingen har fungert, og i enighet ble det kommet frem til både positive og negative sider. Blant gruppen har motivasjonen vært bra fra start til slutt. Alle har møtt opp til de planlagte møtene, og stilt opp dersom andre på gruppen har trengt ekstra hjelp. Arbeidsmengden var spesielt stor frem til påskeferien, men avtok noe etter da flere av gruppens medlemmer også måtte prioritere eksamen i andre emner. Det var spesielt vanskelig å finne tidsrom som passet alle, da gruppen består av seks personer fra tre forskjellige studierettinger. Gruppen føler likevel at prosjektet har hatt en jevn arbeidsfordeling, litt ettersom når hvert enkelt gruppemedlem har tid.

Selv om motivasjon og arbeidsfordeling har vært god gjennom prosjektet, har gruppen funnet frem forbedringspotensialet knyttet til fremtidige prosjekter. Tidlig i prosjektet ble det bestemt at alle gruppemedlemmene skulle få prøve seg litt på de forskjellige arbeidsoppgavene. Dette var ikke tilfellet for alle. Enkelte fikk faste roller i prosjektet. Dette var ikke nødvendigvis en ulempe. Faste roller har sikret god arbeidsflyt, og gruppen tror at sluttpunktet ble bedre ved at medlemmene jobbet med egne styrker og interesseområder. Onsdager var satt av til felles arbeidsøkt, men gruppen ser i ettertid at det hadde vært nyttig med flere dager med fellesarbeid i løpet av uken. Et alternativ hadde vært å utvide retrospektmøtet på fredagene, til lengre felles arbeidsøkter. Videre har de fleste arbeidsoppgavene blitt fordelt parvis, hvor inndelingen har vært basert på studie og timeplan. Det kan tenkes at enkelte arbeidsoppgaver kunne vært bedre løst dersom det ble delt inn i par fra ulike

studieretninger, da en bredere kompetanse kunne resultert i andre løsninger.

Kommunikasjon og verktøy

Kommunikasjonen i løpet av prosjektet blir ansett som svært god fra gruppemedlemmer. Alle føler seg sett og hørt, og både Facebook Messenger og Teams har blitt tatt i bruk aktivt dersom spørsmål eller uklarheter har kommet opp. Samtlige medlemmer har også vært flinke til å ta kontakt dersom de har problemer, slik at andre medlemmer kan bidra med sin kompetanse.

Verktøyene som ble benyttet for å støtte arbeidet var Trello, Teams, Facebook Messenger, Overleaf, GitHub, Miro, Figma og Google Disk. Grunnet variert erfaringsgrunnlag med de ulike verktøyene, ble det opprettet en ansvarlig for hvert verktøy. Teams har i hovedgrunn blitt brukt til prosjektrelatert informasjon, samt kontakt med veiledere. Facebook Messenger har derimot blitt benyttet til mer generell kommunikasjon og koordinering. GitHub var nytt for samtlige av gruppemedlemmene, noe som ble ansett som tungvint i startfasen, men bedret seg i ettertid. Alle ser nytten i verktøyet og er svært fornøyde. Overleaf var nytt for de fleste i gruppen, og var det verktøyet som ble møtt med mest skepsis i starten. Medlemmene med mest erfaring ble satt til hovedansvarlig for verktøyet. Dette gjorde resterende av medlemmene mer trygge. Gruppen angrer ikke på valget, da man ser fordelene i lengre rapporter. Trello ble i hovedsak brukt som et kanbanboard, med oversikt over alle arbeidsoppgavene. Ingen hadde erfaring med Trello i begynnelsen, men var enkelt for samtlige å sette seg inn i. Gruppen fant imidlertid ut at oppretting av mange tavler krever betaling, men gratisversjonen holdt akkurat ut prosjektperioden. Dersom gruppen skulle fortsette med prosjektet, blir det sett på som nødvendig å betale for tjenesten. Google Disk fungerte som et felles møtepunkt hvor alle filer, bilder, design, møtereferater og arbeidslogger ble delt. Dette verktøyet var alle medlemmer kjent med fra før, og fungerte utmerket. Miro ble brukt til grafiske løsninger og fungerte bra. Figma hadde kun blitt brukt av ett av medlemmene tidligere, men ble ansett som enkelt å lære. Selv om Figma hadde sine begrensninger, resulterte det i svært fine bilder.

Designprosess

Gruppen valgte å legge vekt på designprosessen siden en tredjedel av gruppen har en del erfaring og kompetanse innenfor design. Disse medlemmene har fått brukt mye pensum fra tidligere emner, samt at resterende medlemmer har fått et innblikk i designdelen i et utviklingsprosjekt. Det har derimot krevd en del planlegging, og tatt mye tid. Likevel er gruppen imponert over designprosessen i dette prosjektet, og synes det er flott at dette ble vektlagt.

Veien videre

Gruppen er stolte over *FROLF*. Brukertestens resultater viser til en brukervennlig og vellaget applikasjon. Alle må-kravene er oppfylt, samt flere av bør-kravene. Potensialet for applikasjonen er stort. Det finnes mange veier videre for applikasjonen, og gruppen har flere ønskede funksjonaliteter som ikke ble implementert. For videre utvikling, bør kravspesifikasjonen bli brukt. Gruppens opprinnelige ønske var å legge applikasjonen ut på Google Play. Da må API-nivå økes. Dette vil trolig medføre andre endringer i applikasjonen for å støtte dette nivået.

Referanser

- Agile Manifesto. (2022). *Manifestet for smidig programvareutvikling*. Hentet 10. mai 2022, fra <https://agilemanifesto.org/iso/no/principles.html>
- Android for Developers. (2022a). *Adapter*. Hentet 28. april 2022, fra <https://developer.android.com/reference/android/widget/Adapter>
- Android for Developers. (2022b). *Develop Android apps with Kotlin*. Hentet 28. april 2022, fra <https://developer.android.com/kotlin>
- Android for Developers. (2022c). *Fragments*. Hentet 11. mai 2022, fra <https://developer.android.com/guide/fragments>
- Android for Developers. (2022d). *Guide to app architecture*. Hentet 28. april 2022, fra <https://developer.android.com/topic/architecture>
- Android for Developers. (2022e). *Meet Android Studio*. Hentet 28. april 2022, fra <https://developer.android.com/studio/intro>
- Android for Developers. (2022f). *Meet Google Play's target API level requirement*. Hentet 1. april 2022, fra <https://developer.android.com/google/play/requirements/target-sdk>
- Android for Developers. (2022g). *Migrating from Android 10 (API level 29) to Android 11 (API level 30)*. Hentet 10. mai 2022, fra <https://developer.android.com/google/play/requirements/target-sdk#expandable-1>
- Android for Developers. (2022h). *Run apps on the Android Emulator*. Hentet 28. april 2022, fra <https://developer.android.com/studio/run/emulator>
- Android for Developers. (2022i). *Splash screens*. Hentet 11. mai 2022, fra <https://developer.android.com/guide/topics/ui/splash-screen>
- Android for Developers. (2022j). *Test your app*. Hentet 28. april 2022, fra <https://developer.android.com/studio/test>
- Arisholm, E., Gallis, H., Dybå, T. & Sjøberg, D. (2007). Evaluating Pair Programming with Respect to System Complexity and Programmer Expertise. *Software Engineering, IEEE Transactions on*, 33, 65–86. <https://doi.org/10.1109/TSE.2007.17>
- Bowen, D. (2020). *ISTQB's 7 Principles of Testing*. Hentet 19. mai 2022, fra <https://medium.com/@douglastylerbowen/istqbs-7-principles-of-testing-ca5c53b57f8b>
- FROLF. (2022). *What is Frolf?* Hentet 10. mai 2022, fra <https://www.frolf.com/>
- Google. (2022). *Gson*. Hentet 10. mai 2022, fra <https://github.com/google/gson>
- Google Maps Platform. (2022). *Maps SDK for Android overview*. Hentet 20. mars 2022, fra <https://developer.android.com/google/play/requirements/target-sdk>
- Henrik Alfheim, S. A. (2022). *IN1030: Kvalitet og test i prosjektarbeid*. Hentet 10. mai 2022, fra <https://www.uio.no/studier/emner/matnat/ifi/IN2000/v22/forelesninger/in2000.2022.02.07-test-og-kvalitet-slides.pdf>
- ISO. (2017). *ISO/IEC 25010:2011*. Hentet 10. mai 2022, fra <https://www.iso.org/standard/35733.html>
- Kittinun Vantasin. (2021). *Fuel*. Hentet 10. mai 2022, fra <https://github.com/kittinunf/fuel>
- Kittinun Vantasin. (2022). *A type-safe HTTP client for Android and Java*. Hentet 10. mai 2022, fra <https://square.github.io/retrofit/>

- Lazar, J., Feng, J. H. & Hochheiser, H. (2017). *Research Methods in Human-Computer Interaction* (2nd). Utgivelsessted: Morgan Kaufmann Publishers.
- Lindsjørn, Y. (2022a). *IN1030: Modellering av krav*. Hentet 10. mai 2022, fra https://www.uio.no/studier/emner/matnat/ifi/IN1030/v22/forelesningsressurser/in1030_2022.03.29_use-case-modellering.pdf
- Lindsjørn, Y. (2022b). *IN2000: Kravhåndtering, modellering, design og prinsipper*. Hentet 5. mai 2022, fra <https://www.uio.no/studier/emner/matnat/ifi/IN2000/v22/forelesninger/in2000.2022.02.16.modellering.pdf>
- Lindsjørn, Y. (2022c). *IN2000: Kravhåndtering, modellering, design og prinsipper*. Hentet 9. mai 2022, fra https://www.uio.no/studier/emner/matnat/ifi/IN2000/v22/forelesninger/in2000.2022.3.7_metode.pdf
- Maus, A. (2010). *NF1050: Kravhåndtering*. Hentet 10. mai 2022, fra <https://www.uio.no/studier/emner/matnat/ifi/INF1050/v10/undervisningsmateriale/INF1050Forelesning4-Krav%C3%A5nteringv2010.pdf>
- MeteorologiskInstitutt. (2022). *Velkommen til api.met.no*. Hentet 1. april 2022, fra https://in2000-apidproxy.ifi.uio.no/index_no.html
- Microsoft. (2022). *XML for nybegynnere*. Hentet 28. april 2022, fra <https://support.microsoft.com/nb-no/office/xml-for-nybegynnere-a87d234d-4c2e-4409-9cbc-45e4eb857d44>
- n:point. (2018). *JSON storage bins that won't break your app*. Hentet 10. mars 2022, fra <https://www.npoint.io>
- Owren, B. (2019). *Kontrastfarger - spennende motspillere*. Hentet 5. mai 2022, fra <https://www.ifi.no/kontrastfarger-spennende-motspillere>
- Papajorgji, P. J. & Pardalos, P. M. (2014). *UML Diagrams*. Springer US. https://doi.org/10.1007/978-1-4899-7463-1_6
- Sharp, H., Rogers, Y. & Preece, J. (2019). *Interaction Design: beyond human-computer interaction* (5. utg.). Indiana: John Wiley Sons.
- SNL. (2019). *Aiolos - vindenes hersker*. Hentet 28. april 2022, fra https://snl.no/Aiolos_-_vindenes_hersker
- Sommerville, I. (2011). *Software Engineering* (9th). New Jersey: Pearson.
- Sommerville, I. (2019). *Engineering Software Products* (1st). New Jersey: Pearson.
- Sundaramoorthy, S. (2022). *Introduction to UML Diagrams and Its Components*. Taylor Francis Ltd. <https://doi.org/10.1201/9781003287124-1>
- Tek.no. (2017). *Nordmenn handler mer og dyrere elektronikk*. Hentet 10. mai 2022, fra <https://www.tek.no/nyheter/nyhet/i/4qXdVe/nordmenn-handler-mer-og-dyrere-elektronikk>
- Tverga. (2022). *Hva er discgolf/frisbeegolf?* Hentet 10. april 2022, fra <https://tverga.no/aktivitet/discgolf/>
- UiO. (2021a). *Hva lærer du?* Hentet 3. mars 2022, fra <https://www.uio.no/studier/program/info-design/hva-lerer-du/>
- UiO. (2021b). *Hva lærer du?* Hentet 3. mars 2022, fra <https://www.uio.no/studier/program/informatikk-ledelse/hva-lerer-du/>

- UiO. (2021c). *Hva lærer du?* Hentet 3. mars 2022, fra <https://www.uio.no/studier/program/informatikk-programmering/hva-lerer-du/>
- UiO. (2022). *Nettskjema*. Hentet 10. mars 2022, fra <https://www.uio.no/tjenester/it/adm-app/nettskjema>
- UUtilsynet. (2022a). *Universell utformig av apper*. Hentet 11. mai 2022, fra <https://www.uutilsynet.no/regelverk/universell-utforming-av-apper/230>
- UUtilsynet. (2022b). *WCAG 2.0 - standarden*. Hentet 12. mai 2022, fra <https://www.uutilsynet.no/wcag-standarden/wcag-20-standarden/86>

Vedlegg

A Teamavtale

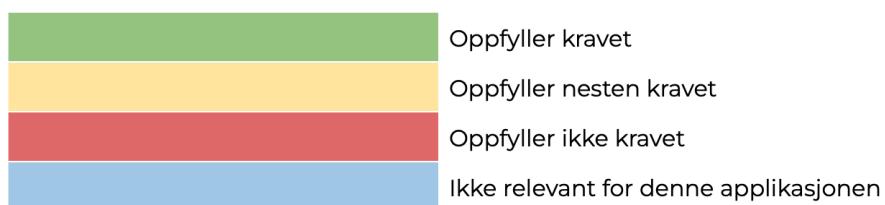
Teamavtalen danner grunnlag for samarbeidet mellom medlemmene i gruppen. Gruppens yteevne er avhengig av at alle medlemmene tar medansvar og bidrar til et motiverende læringsmiljø. Tilstedeværelse, tidsbruk, forventninger til den enkeltes bidrag, og konfliktløsning ble diskutert for å oppnå en felles forståelse av hverandres forventninger og for å bestemme felles krav.

Følgende krav ble satt:

1. Det skal være jevn arbeidsfordeling og hvert enkelt medlem skal utføre avtalt arbeid til avtalt tid. Skulle noe oppstå som gjør at dette punktet ikke følges skal det kommuniseres så snart som mulig til gruppen slik at de andre gruppemedlemmene kan bidra til at arbeidet blir utført.
2. Det er forventet at alle kan be hverandre om hjelp og at alle tilbyr sin hjelp der de kan.
3. Alle skal møte til avtalt tid. Om man ikke møter skal man gi beskjed i gruppechatten. Om dette skjer gjentatte ganger skal det tas opp i et drøftningsmøte.
4. Det skal jobbes effektivt med konkrete mål som fastsettes på mandagsmøtene.
5. Onsdagmøtene går til å jobbe samlet og oppdatere hverandre på status på arbeidet.
6. Fredagsmøtene går til å gjennomføre et retrospektiv av uka som har gått.
7. Alle har et felles ansvar for god informasjonsflyt. Dette kan innebære aktiv bruk av gruppechat på Teams.
8. Det forventes at kan jobbe opp til ca. 30 timer i uka til prosjektet.
9. Det skal være åpen kommunikasjon og stor takhøyde for tilbakemeldinger, både positive og konstruktive.
10. Alle skal være åpne for å ta på oss ulike arbeidsoppgaver, men gruppen ønsker å spille på den enkeltes styrke.
11. Det er ønskelig å praktisere en flat struktur. Dette innebærer at ingen utpekes som en enkelt leder i gruppen. Hver uke rulleres det på hvem som skal være møteleder og referent.
12. Skulle gruppen havne i en situasjon der avstemningen ender i 3-3, og ingen ønsker å endre sin stemme skal avgjørelsen tas ved stein, saks, papir.

B Universell utforming

Dette er en oversikt over hvilke WCAG 2.1 retningslinjer applikasjonen FROLF utfyller. Retningslinjene er inndelt i prinsipper. Forklaringer og inndeling er hentet fra uutilsynet.no sin nettside “WCAG sortert etter prinsipp” (UUtilsynet, 2022a). Disse prinsippene er hovedsaklig ment for nettsteder, men som uutilsynet nevner gjelder kravene i regelverket for nettløsninger, og nettløsninger omfatter både nettsteder og mange apper til nettrett og mobil. Apper som trenger internettforbindelse for å kunne brukes etter at de er lastet ned, skal følge kravene til universell utforming av ikt. De sier også at hvilke krav som gjelder i praksis, varierer etter appens funksjonalitet og kompleksitet. Kravene i WCAG gjelder innholdstype, og dersom denne innholdstypen finnes i appen, skal du følge kravet. (UUtilsynet, 2022a) Figur 23 indikerer i hvilken grad applikasjonen oppfyller kravet med tilhørende fargekode.



Figur 23: Grad av oppfylt krav med tilhørende fargekode.

Resultatet fra prinsipp *Robust* er oppsummert i figur 24.

III. Robust

Nummer		Nivå	Good?
4.1.1	Parsing (oppdeling) Alle sider skal være uten store kodefeil	A	
4.1.2	Navn, rolle, verdi Alle komponenter har navn og rolle bestemt i koden	A	

Figur 24: Prinsipp Robust. Følgende punkter ble forsøkt å ta hensyn til under implementasjonen av FROLF.

Resultatet fra prinsipp *Mulig å oppfatte* er oppsummert i figur 25.

I. Mulig å oppfatte

Nummer		Nivå	Good?
1.1.1	Ikke tekstlig innhold Gi brukeren et tekstalternativ for innhold som ikke er tekst.	A	
1.2.1	Bare lyd og bare video (forhåndsinnsnspilt) Gi brukeren et alternativ når innholdet presenteres kun som video eller lyd.	A	
1.2.2	Teksting (forhåndsinnsnspilt) Tilby teksting for video med lyd.	A	
1.3.1	Informasjon og relasjoner Ting skal være kodet som det ser ut som	A	
1.3.2	Meningsfylt rekkefølge Presenter innhold i en meningsfull rekkefølge.	A	
1.3.3	Sensoriske egenskaper Instruksjoner må ikke utelukkende være avhengige av form, størrelse, visuell plassering, orientering, eller lyd for å kunne bli forstått.	A	
1.4.1	Bruk av farge Ikke bruk presentasjon som bygger utelukkende på farge	A	
1.4.2	Styring av lyd Gi brukeren mulighet til å stoppe eller pause lyd som starter automatisk	A	
1.4.3	Kontrast Kontrastforholdet mellom teksten og bakgrunnen er minst 4,5:1.	AA	
1.4.4	Endring av tekststørrelse Teksten kan bli endret til 200% størrelse uten tap av innhold eller funksjon.	AA	
1.4.5	Bilder av tekst Bruk tekst i stedet for bilder av tekst.	AA	

Figur 25: Prinsipp *Mulig å oppfatte*. Følgende punkter ble forsøkt å ta hensyn til under implementasjonen av FROLF.

Resultatet fra prinsipp *Mulig å betjene* er oppsummert i figur 26.

II. Mulig å betjene

Nummer		Nivå	Good?
2.1.1	Tastatur All funksjonalitet skal kunne brukes kun ved hjelp av tastatur.	A	
2.1.2	Ingen tastaturfelle Unngå tastaturfeller	A	
2.2.1	Justerbar hastighet Tidsbegrensninger skal kunne justeres av brukeren	A	
2.2.2	Pause, stopp, skjul Gi brukeren mulighet til å stoppe, pause eller skjule innhold som automatisk endrer seg	A	
2.3.1	Terskelverdi på maksimalt 3 glimt Innhold skal ikke blinke mer enn 3 ganger per sekund	A	
2.4.1	Hoppe over blokker Gi brukeren mulighet til å hoppe direkte til hovedinnholdet.	A	
2.4.2	Sidetitler Bruk nytte og tydelige sidetitler	A	
2.4.3	Fokusrekkefølge Presenter innholdet i en logisk rekkefølge.	A	
2.4.4	Formål med lenke (i kontekst) Alle lenkers mål og funksjon fremgår tydelig av lenketeksten.	A	
2.4.5	Flere måter Tilby brukeren flere måter å navigere på.	AA	
2.4.6	Overskrifter og ledetekster Sørg for at ledetekster og overskrifter er beskrivende	AA	
2.4.7	Synlig fokus Sørg for at alt innhold får synlig fokus når du navigerer med tastatur	AA	

Figur 26: Prinsipp *Mulig å betjene*. Følgende punkter ble forsøkt å ta hensyn til under implementasjonen av FROLF.

Resultatet fra prinsipp *Forståelig* er oppsummert i figur 27.

III. Forståelig

Nummer		Nivå	Good?
3.1.1	Språk på siden Sørg for at språket til innholdet på alle nettsider er angitt i koden	A	
3.1.2	Språk på deler av innhold Sørg for at alle deler av innholdet som er på et annet språk enn resten av siden er markert i koden	AA	
3.2.1	Fokus Når en komponent kommer i fokus medfører dette ikke automatisk betydelige endringer i siden	A	
3.2.2	Inndata Endring av verdien til et skjemafelt medfører ikke automatisk betydelige endringer i siden	A	
3.2.3	Konsekvent navigering Navigasjonslenker som gjentas på flere sider skal ha en konsekvent rekkefølge	AA	
3.2.4	Konsekvent identifikasjon Elementer som har samme funksjonalitet på tvers av flere sider er utformet likt.	AA	
3.3.1	Identifikasjon av feil For feil som oppdages automatisk må du vise hvor feilen har oppstått og gi en tekstbeskrivelse av feilen.	A	
3.3.2	Ledetekster eller illustrasjoner Det vises ledetekster eller instruksjoner når du har skjemaelementer som må fylles ut	A	
3.3.3	Forslag ved feil Dersom feil blir oppdaget automatisk, gi brukeren et forslag til hvordan feilen kan rettes	AA	
3.3.4	Forhindring av feil (juridiske-, økonomiske, - og datafeil) For sider som medfører juridiske forpliktelser må det være mulig å kunne angre, kontrollere eller bekrefte dataene som sendes inn	AA	

Figur 27: Prinsipp *Forståelig*. Følgende punkter ble forsøkt å ta hensyn til under implementasjonen av FROLF.

C Spørreundersøkelse

Hvem er målgruppen?

Folk som spiller frisbeegolf av og til, eller har prøvd det før. Vi tenker å legge ut spørreundersøkelsen ulike steder på facebook. Vi tenker også å sende det til folk som spiller frisbeegolf “proft”, de får vi tak i gjennom ‘Frisbeegolf Norge’ - facebooksiden.

Hvordan formulere det?

“Hei! Har du prøvd frisbeegolf, eller spiller det på fritiden? Vi jobber med å lage en applikasjon for de som spiller frisbeegolf for gøy! Svar gjerne på denne undersøkelsen, slik at vi kan finne ut hva som er viktig for deg når du skal spille.”

Hva vil vi finne ut?

- Ferdighetsnivå/ Erfaring
- Hvor ofte man spiller
- CardView som hovedside eller Maps
- Hvilke ‘widgets’ som er mest relevant (temperatur, regn, vind, vær)
- Navigasjonsbar eller hamburgermeny
- Om de er opptatt av regler
- Om de har brukt UDisc (frustrasjoner?)
- Tidligere erfaring med å finne baner (Hvordan de finner? Har de slitt? Var det lett?)

Alternativ - mal

1. Ja - Nei
2. Alltid - Svært Ofte - Ofte - Sjeldent - Svært Sjeldent - Aldri
3. Helt enig - Litt enig - Nøytral - Litt uenig - Helt uenig
4. Skala

Spørsmål

1. Kjønn

Han - Hun - Annet

2. Alder

under 12 - 12-18 - 18-25 - 25-35 - 35- 55 - over 55

3. Hvor ofte spiller du frisbeegolf?

Har prøvd en gang før - Har spilt noen ganger før - Noen ganger i året- Noen ganger i måneden - Flere ganger i uka - Hver dag

4. Hvordan vil du beskrive dine ferdigheter?

Nybegynner - Middels god - Svært god - Spiller turneringer

5. Har du erfaring med applikasjonen UDisc?

Ja - Nei

6. Hvis ja - Hva er hovedgrunnen til at du bruker UDisc?

Tekstfelt (2-5 setninger)

7. Hvis ja - Er det noe du liker med appen? I så fall hva?

Tekstfelt (2-5 setninger)

8. Hvis ja - Er det noe du misliker med appen? I så fall hva?

Tekstfelt (2-5 setninger)

9. Hvordan går du frem for å finne frisbeegolfbaner? (Eks. Gjennom venner, Søker på internett, UDisk Andre applikasjoner etc.)

Tekstfelt (2-5 setninger)

10. Hvordan får du tak i banekart når du skal spille?

UDisc - GoogleMaps - Banens hjemmeside - Annet

11. Hvis annet - Vennligst spesifiser?

Tekstfelt (2-5 setninger)

12. I en situasjon hvor du har behov for en frisbeegolfbane. Hadde du foretrukket å finne banen i et kart, eller i en liste-oversikt?

Kart - Liste

13. Er du opptatt av regler når du spiller?

Ja - Nei

14. Ville du sjekket reglene dersom det oppsto en uenighet på banen?

Ja - Nei

15. Sjekker du været på banen før du skal spille ?

Ja - Nei

16. Foretrekker du å navigere i appen gjennom en navigasjonsbar eller en hamburgermeny? /LEGG INN BILDE MED NAVN/

Navigasjonsbar - Hamburgermeny

17. Hva bruker/ brukte du for å notere poeng når du spiller/spilte?

Tekstfelt (2 -5 setninger)

Skala

Når du skal spille frisbeegolf. På en skala fra 1-5;

1. hvor viktig er det for deg med banekart?
2. hvor viktig er det for deg å vite hvor mye det blåser der du vurderer å spille?
3. hvor viktig er det for deg å vite om det regner/ er vått der du vurderer å spille?
4. hvor viktig er det for deg å vite skydekke der du vurderer å spille?
5. hvor viktig er det for deg å vite temperaturen der du vurderer å spille?
6. hvor viktig er det for deg å vite vanskelighetsgraden der du vurderer å spille?
7. hvor viktig er det å vite andre spillere sin rating av banen du vurderer å spille på?
8. hvor viktig er det for deg å vite nærmeste kiosk eller butikk til der du vurderer å spille?
9. hvor viktig er det for deg å enkelt hente frem reglene når du spiller?
10. hvor viktig er det for deg å holde oversikt over poengene når du spiller?

D Resultater fra kvalitativ spørreundersøkelse

Denne spørreundersøkelsen er laget av Peder Frivold Skotte og Celine Varmann Aamodt, i sammenheng med prosjektarbeidet i IN2000 (Software Engineering med prosjektarbeid) ved Universitetet i Oslo, Institutt for Informatikk. Undersøkelsen ble sendt ut via nettskjema. Den ble delt på to facebookgrupper for frisbee-entusiaster , samt facebooken til både Peder og Celine. De som svarte ble holdt anonyme. Etter en opprydding i svarene var det totalt 189 svar som ble tatt i beregning.

D.1 Generell oversikt

Spørsmål 1: "Hvilket kjønn er du ?". Svarresultatene er lagt ved i tabell 3.

Svar	Antall	Prosent
Kvinne	26	13,8 %
Mann	162	85,7 %
Annet	1	0,5 %

Tabell 3: Svarresultatet fra spørsmål 1.

Spørsmål 2: "Hvilken aldersgruppe tilhører du?". Svarresultatene er lagt ved i tabell 4.

Svar	Antall	Prosent
under 12	0	0 %
12-17	10	5,3 %
18-25	68	36 %
26-35	66	34,9 %
36-55	38	20,1 %
over 55	7	3,7 %

Tabell 4: Svarresultatet fra spørsmål 2.

Spørsmål 3: "Hvor ofte spiller du frisbeegolf?". Svarresultatene er lagt ved i tabell 5.

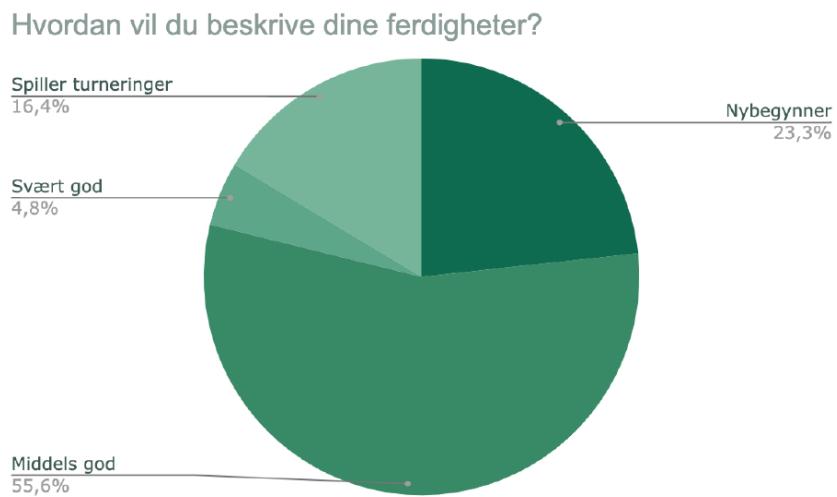
Svar	Antall	Prosent
Har prøvd en gang før	11	5,8 %
Har spilt noen ganger før	14	7,4 %
Noen ganger i året	19	10,1 %
Noen ganger i måneden	46	24,3 %
Flere ganger i uka	93	49,2 %
Hver dag	6	3,2 %

Tabell 5: Svarresultatet fra spørsmål 3.

Spørsmål 4: "Hvordan vil du beskrive dine ferdigheter?". Svarresultatene er lagt ved i tabell 6. Figur 28 viser et kakediagram over svarene.

Svar	Antall	Prosent
Nybegynner	44	23,3 %
Middels god	105	55,6 %
Svært god	9	4,8 %
Spiller turneringer	31	16,3 %

Tabell 6: Svarresultatet fra spørsmål 4.



Figur 28: Kakediagram over svarene i tabell 6.

Som man kan se fra disse svarene er de fleste av deltakerne menn i aldersgruppen 18-25 år. Nesten halvparten av dem spiller flere ganger i uka, og over halvparten av dem beskriver sine ferdigheter som middels gode.

D.2 Andre applikasjoner

Ettersom at det allerede eksisterer en frisbeegolf-app (UDisc) var vi interessert i å vite hvor mange som allerede bruker den appen, og hva de eventuelt liker eller ikke liker med den.

Spørsmål 5: "Har du erfaring med applikasjonen UDisc?". Svarresultatene er lagt ved i tabell 7.

Svar	Antall	Prosent
Ja	149	78,8 %
Nei	40	21,2 %

Tabell 7: Svarresultatet fra spørsmål 5.

Spørsmålene presentert nedenfor tillot brukerne å svare uten å måtte velge mellom forutbestemte alternativer. Ettersom responsen på spørreundersøkelsen var mye større enn forventet, resulterte dette i en overveldende mengde svar. Mange av deltakerne svarte mer eller mindre det samme. I og

med at flesteparten av de som deltok spilte frisbeegolf flere ganger i uka var det også noen svar som opplevdes i overkant spesifikke for vårt prosjekt. Vi har plukket ut de punktene som er mest relevante for utformingen av vår applikasjon, samt samlet de svarene som var like.

Spørsmål 6: “Hva er hovedgrunnen til at du bruker UDisc?”

- Enkel i bruk
- Praktisk
- Oppdage nye baner
- Oversikt over baner
- Eneste som finnes
- Loggføring/ Lagre runder
- Kart over banen
- Den brukes i turneringer

Spørsmål 7: “Er det noe du liker med appen? I så fall hva?”

- Statistikk på eget spill
- Enkel i bruk
- Kartfunksjonen
- At “alle” bruker den
- Kan logge for gruppen når vi spiller sammen
- Den har baner rundt hele verden
- All informasjon ligger lett tilgjengelig
- Liker hvordan den ser ut
- At man kan finne ut hvor langt man kaster
- GPS-kart over banene
- Godt UI

Spørsmål 8: “Er det noe du mislikter med appen? I så fall hva?”

- Burde vært lettere å legge til venner (sosial)
- Koster penger (Betalingsmur for Pro)
- Kan være litt mye å forstå i starten
- At Dark Mode ikke er en mulighet

- Noe kronglete brukergrensesnitt *Dette er i stor grad noe som handler om personlig preferanse.
Verdt å nevne at noen har svart det motsatte også.
- Vil gjerne ha notis for hver bane, slik at man kan skrive diskvalg

D.3 Været

I tillegg til å være et førstevalg for nybegynnere, var vi også opptatt av at vi kunne bidra med en app som samlet værdata og banedata på ett sted. Derfor ville vi finne ut av hvilke aspekter ved været de som spiller frisbeegolf er opptatt av. Ikke minst om de faktisk er opptatt av været når de skal spille.

Spørsmål 9: "Sjekker du været på banen før du skal spille?". Svarresultatene er lagt ved i tabell 8. Figur 29 illustrerer et kakediagram av svarene.

Svar	Antall	Prosent
Ja	163	86,2 %
Nei	26	13,8 %

Tabell 8: Svarresultatet fra spørsmål 9.



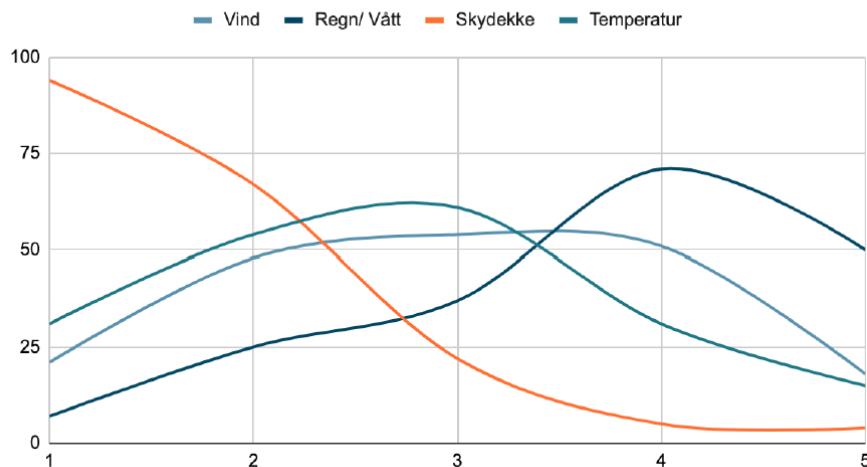
Figur 29: Kakediagram over svarene i tabell 6.

Spørsmål 10: "På en skala fra 1-5 der 1 er svært lite viktig og 5 er svært viktig". Svarresultatene er lagt ved i tabell 9. Figur 30 illustrerer et linjediagram over svarene.

Spørsmål	1	2	3	4	5
Hvor viktig er det for deg å vite hvor mye det blåser der du vurderer å spille?	21	48	54	51	18
Hvor viktig er det for deg å vite om det regner/ er vått der du vurderer å spille?	7	25	37	71	50
Hvor viktig er det for deg å vite skydekket der du vurderer å spille?	94	67	22	5	4
Hvor viktig er det for deg å vite temperaturen der du vurderer å spille?	31	54	61	31	15

Tabell 9: Svarresultatet fra spørsmål 10.

Hvilke aspekter ved været er deltakerne mest opptatt av?



Figur 30: Linjediagram over svarene i tabell 9.

Vi la også inn en mulighet for deltakerne til å tilføye noe selv på slutten av undersøkelsen, dersom de ville det. Da var det noen deltakere som skrev dette om været.

“Værfunksjon og generell info hadde vært dødsiktig”

- Spiller flere ganger i uka (spiller turneringer). Bruker UDisc.

“ Det er viktig for meg å sjekke ut forholdene før å kunne kle meg korrekt. Sporten kan spilles i all slags vær. Bortsett fra snø. Jeg hater snø.”

- Spiller noen ganger i måneden (middels god). Bruker UDisc.

Man kan se på resultatene at værforhold er viktig for disse deltakerne. Det er litt over 85% som har svart “ja” på om de sjekker været før de skal spille. Det viktigste aspektet med været virker å være om det regner eller er vått på banen. Minst viktig er skydekke, mens vind og temperatur virker å være middels viktig.

D.4 Regler

I tillegg til værfunksjon og baneoversikt, var vi interessert i å vite hvor viktig det var for deltakerne med regler.

Spørsmål 11: “Er du opptatt av regler når du spiller?”. Svarresultatene er lagt ved i tabell 10.

Svar	Antall
Ja	164
Nei	25

Tabell 10: Svarresultatet fra spørsmål 11.

Spørsmål 12: “Ville du sjekket reglene dersom det oppsto en uenighet på banen?”. Svarresultatene

er lagt ved i tabell 11.

Svar	Antall
Ja	160
Nei	29

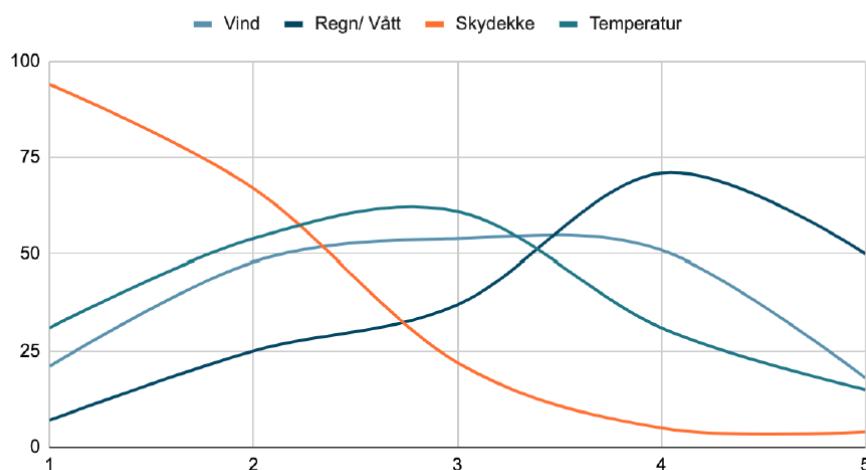
Tabell 11: Svarresultatet fra spørsmål 12.

Spørsmål 13: “På en skala fra 1 til 5, hvor 1 er svært lite viktig og 5 er svært viktig. Hvor viktig er regler for deg?”. Svarresultatene er lagt ved i tabell 12. Figur 31 illustrerer et linjediagram over svarene.

	1	2	3	4	5
Antall	31	35	69	39	18

Tabell 12: Svarresultatet fra spørsmål 13.

Hvilke aspekter ved været er deltakerne mest opptatt av?



Figur 31: Linjediagram over svarene i tabell 12.

D.5 Layout

Vi var også interessert i hvilke preferanser disse deltakerne hadde når det kom til applikasjonens layout. Det viktigste for oss var å finne ut hvordan de foretrekker å navigere i applikasjoner, og hvilken funksjon appen skulle ha som hovedside.

Spørsmål 14: “Foretrekker du å navigere i appen gjennom en navigasjonsbar eller en hamburgermeny?” *Brukerne ble presentert med bilder av valgalternativene. Svarresultatene er lagt ved i tabell 13.

Svar	Antall	Prosent
Hamburgermeny	33	17,5 %
Navigasjonsbar	110	58,2 %
Vet ikke	46	24,3 %

Tabell 13: Svarresultatet fra spørsmål 14.

Spørsmål 15: “I en situasjon hvor du har behov for en frisbeegolfbane. Hadde du foretrukket å finne banen i et kart, eller i en liste-oversikt?”. Svarresultatene er lagt ved i tabell 14.

Svar	Antall	Prosent
Kart	149	79,3 %
Liste	2	11,7 %
Vet ikke	17	9 %

Tabell 14: Svarresultatet fra spørsmål 15.

D.6 Andre tilleggsfunksjoner

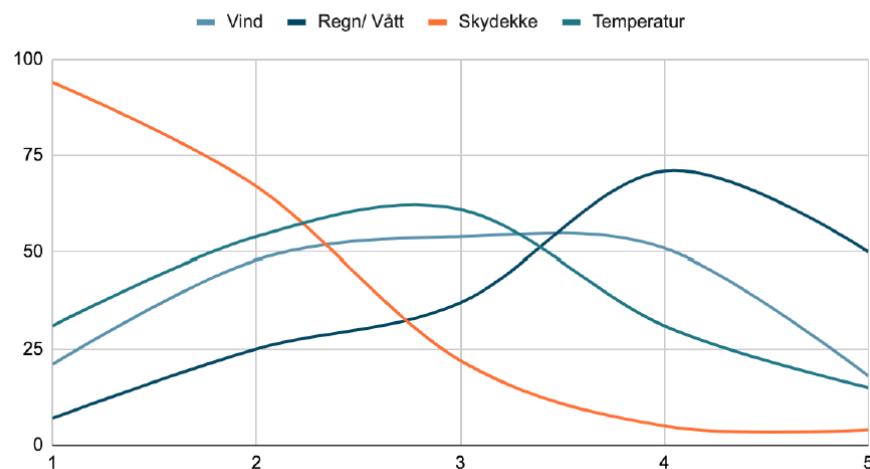
For å sørge for at vi implementerte de viktigste funksjonene var vi interessert i å vite hvilke av følgende aspekter som var viktig for frisbeegolferne når de skal spille.

Spørsmål 16: “På en skala fra 1 til 5, hvor 1 er svært lite viktig og 5 er svært viktig. Hvor viktig er disse aspektene for deg når du skal spille frisbeegolf?” Svarresultatene er lagt ved i tabell 15. Figur 32 illustrerer et linjediagram over svarene.

	1	2	3	4	5
Vanskelighetsgrad	15	38	56	56	27
Rating	38	31	50	48	23
Kiosk	75	41	35	30	11
Banekart	9	11	35	80	51

Tabell 15: Svarresultatet fra spørsmål 16.

Hvilke aspekter ved været er deltakerne mest opptatt av?



Figur 32: Linjediagram over svarene i tabell 15.

E Intervjuplan

Hva trenger vi?

- mulighet for å ta notater
- nettskjema-diktafon
- PC/Mac
- samtykkeskjema

Oppnak

- opptak gjennom diktafon på fysisk intervju
- ta opptak vha. zoom-funksjon på digitalt intervju

Struktur

- semi-strukturert, etterlign en samtale for å gjøre intervjuobjekt komfortabel
- åpne spørsmål - legge opp til egne refleksjoner og oppfølgingsspørsmål
- skape relasjon for godt videre samarbeid - intervjuobjekt kan også bli kjent med intervjuer til en viss grad

Annet

- stillhet kan være bra
- tid til å reflektere
- pass på tempo, ikke for fort
- få personen til å føle seg trygg
- 5 why's! spør spør spør!

Guide:

Innledning: 5 min

- Fortelle kort om prosjektet, tema:
 - Forklare formål, avklare eventuelle etiske spørsmål og frivillighet
 - Samtykkeskjema
- Fortelle om hva slags data som vil bli samlet inn og hvordan det blir oppbevart
- Ta bilde til dokumentasjon
- Spørre om intervjuobjekt trenger noe/vil hente noe før vi starter
- Noe du lurer på?

Oppvarming: 5-10 min

- Enkle spørsmål om f.eks. en selv, bli kjent med hverandre
 - Lett og ledig uten kontroversielle spørsmål
 - Viktig: semi-strukturert!
- Hva driver du med/okkupasjon?
 - Vil du si litt om din bakgrunn og utdanning?
 - Har du blitt intervjuet tidligere?
 - Noen av disse spørsmålene er også fra spørreundersøkelsen dersom du har svart på den.
 - Hvor finner du inspirasjon?
 - Hva liker du å holde på med utenom jobb?
 - Hvor bor du?
 - Trives du?
 - Hvordan er bo-situasjonen din?
 - Har du noen hobbyer? (hvis ikke de nevner det) også kanskje frisbeegolf?

Hoveddel

- Hvor ofte spiller du frisbeegolf?
- Hvordan vil du beskrive dine ferdigheter?
- Spiller du alene noen ganger?
- Er du interessert i lære deg riktig teknikk ?
 - hvis ja: Hvordan henter du informasjon om dette?
- Spiller du i dårlig vær (regn, vind, etc.)?
- Har vinden noe å si når man spiller frisbeegolf?
- Tenker du på UV indeksen på banen du skal spille på?
- Sjekker du været på banen før du skal spille?
- Ta oss gjennom en dag der du velger å spille frisbeegolf, fra du bestemmer deg for å spille til du er på banen.
 - Hvordan man finner banen
 - Poeng?
 - Konkurranse med andre?
- Sjekker du vanskelighetsgraden på banen du skal spille på?
- Spiller avstanden til banen en rolle for deg når du velger hvor du skal spille?

- Har det noe å si for deg hvor mange andre som er på banen når du skal spille?
- Hvordan kommer du deg til banen ?

Hvis bil: Er det greit å parkere? Stresser du over parkering?

- Har du brukt UDisc før?
- Hva er hovedgrunnen til at du bruker UDisc?

Hvis ja: Er det noe du liker med appen? I så fall hva?

Hvis ja: Er det noe du mislikter med appen? I så fall hva?

- Er du opptatt av regler når du spiller?
- Hva bruker/ brukte du for å notere poeng når du spiller/spilte?

Avrunding: 5-10 min

- er det noe intervjuobjektet selv vil ta opp?
- Hvordan synes du dette var?
- Er det noe du ønsker å ta opp/refleksjoner som du mener burde være med?

Avslutning

Tusen takk for at du ville delta på intervju med meg i dag, om det er noe som helst du lurer på - rundt spørsmålene/datainnsamling/prosjektet generelt - så må du bare ta kontakt. Du kan i tillegg finne informasjon i samtykkeskjemaet som du får med deg. Ha en fortsatt fin dag!

F Samtykkeskjema

Vil du delta i et intervju for utvikling av en brukervennlig frisbeegolf app?

Vi er seks studenter i emnet *IN2000 – Software Engineering med prosjektarbeid* ved Institutt for informatikk ved Universitetet i Oslo. Med dette skrivet ønsker vi å informere hva prosjektet vårt har som formål, spørre deg om du vil delta i prosjektet, samt berette hva deltagelsevil innebære for deg.

Formål

Formålet med vårt prosjekt er å undersøke brukskonteksten til en typisk frisbeegolf runde og finne ut hvilke sider i denne konteksten som en app kan hjelpe til med. For å undersøke dette ønsker vi å ha et intervju med deg.

Deltakelse

Du blir spurta om å delta fordi du faller innenfor vår målgruppe, definert som frisbeegolf spiller. Dersom du velger å delta ønsker vi å ta opptak av lyd for vår datainnsamling, samt observere. Intervjuet vil vare i 45-60 minutter.

Frivillig deltagelse

Det er frivillig å delta. Du kan når som helst avslutte eller trekke tilbake informasjon som er gitt. Du kan når som helst velge å trekke samtykket uten å måtte oppgi grunn. Dersom samtykket trekkes vil eventuelle personopplysninger som er innsamlet om deg slettes og det vil ikke innebære noen negative konsekvenser for deg at du velger å trekke ditt samtykke.

Personvern: innsamling, oppbevaring, behandling og bruk av dine opplysninger

Ingen sensitive personopplysninger (jf. Personvernforordningens artikkel 9 og 10) vil bli innsamlet. Personlige opplysninger om deg vil kun benyttes til formålene beskrevet i dette informasjonsskrivet. Vi behandler opplysningene konfidensielt og i samsvar med personvernregelverket.

Personlige opplysning innsamlet i opptaket vil bli anonymisert i transkriberingen og rapporteringen; ingen andre enn oss og vår sensur vil ha tilgang til dataen, og det som oppbevares av anonymisert rapportering fra intervjuet vil følge Universitetet i Oslo sine rutiner for sikker oppbevaring.

Navn og kontaktinformasjon vil ikke bli notert. Dataen kan ettersendes deg ved ønske. Dataen som oppbevares, inkludert anonymisert data, vil ikke bli publisert og vil heller ikke kunne tilbakeføres til deg.

Hva skjer med innsamlet data når studentprosjektet avsluttes?

Alle opptak blir slettet senest 20.05.22. Dette gjelder også anonymiserte og avidentifiserte opplysninger om deg.

Rettigheter

Vi behandler opplysninger om deg basert på ditt samtykke. Så lenge du kan identifiseres i datamaterialet, har du rett til:

- Innsyn i hvilke personopplysninger som er registrert om deg, og å få utlevert en kopi av opplysningsene,
- å få rettet personopplysninger om deg,
- å få slettet personopplysninger om deg, og
- å sende klage til Datatilsynet om behandlingen av dine personopplysninger.

Har du spørsmål til undersøkelsen, eller ønsker å benytte deg av dine rettigheter, ta kontakt Peder (pederfs@ui.no) eller vår universitetsansatte fagkontakt Sergey Jakobsen (sergeyj@ui.no)

Før intervjuet begynner ber vi deg om å samtykke i deltagelsen ved å undertegne på at du har lest og forstått informasjonen på dette arket, og ønsker å stille opp til lydintervju.

Med vennlig hilsen Celine Varmann Aamodt, Daniel Kongshaug, Eirin Nyfløt, Nora Skjæveland Utseth, Oscar Berget Dahl, Peder Frivold Skotte

Universitetet i Oslo

Samtykkeerklæring

Jeg har mottatt og forstått informasjon om brukerundersøkelsen digitale læringsplattformer, og har fått anledning til å stille spørsmål. Jeg samtykker til at mine opplysninger behandles frem til semesteret er avsluttet.

Sted og dato

Fullt navn

Signatur

G Affinity Diagram

Figur 33 illustrerer et affinity diagram fra dybdeintervjuet.



Figur 33: Affinity Diagram av data fra dybdeintervju

H Brukertesting

Deltakere

Antall : ca. 5-10 deltagere

Hvor: Kappa

Når: Mandag 09.05, kl. 10:00-12:00

Hvem: Studenter vi finner på gangen på IFI.

Omgivelser

På et grupperom på IFI, altså kontrollert. Dette er fordi det er lett å finne deltagere på IFI, og fordi vi har tilgang til "skjermede" grupperom som studenter. OBS: Deltakerne må se inn mot veggen så de ikke blir distraheret

Målinger

Hovedhensikt: User satisfaction - Brukerens tilfredshet. Dette er fordi vi er relativt sikre på at vi ikke har noen alvorlige feil som krasjer applikasjonen. Vi er mer interessert i brukervennlighet og flow. (Sjekk bok om flow)

Annet vi kan måle: Antall feil, tid brukt på korrekt utførelse

Oppgaveliste

1. Hva er temperaturen på Muselunden Discgolfpark? (VÆR)
2. Opprett et poengkort for Holmenkollen DiscGolfPark (BANEDATA)
3. Forklar sikkerhetsreglene slik de er beskrevet i applikasjonen. (REGLER)
4. Finn ut hvordan man kommer seg til Krokholt Disc Golf Course med kollektiv transport (BANEDATA)
5. Er det toalettmuligheter ved Stovner Discgolfpark? (BANEDATA)
6. Finn banekartet til Muselunden Discgolfpark, og gjør slik at den dekker hele skjermen. (BANEDATA)
7. Finn veibeskrivelsen til Røa Frisbeegolfbane. (KART)
8. Finn vindhastighet og retning på Ekeberg Frisbeegolf. (VÆR)
9. Opprett et poengkort for Røa Fisbeegolfbane. (Brukeren gjør det først) Før 4 kast på alle hullene og beregn poeng.(POENG)

Utstyr

Taleopptak. Mulighet for å ta notater. Samtykkeskjema.

I Kravspesifikasjon

Tabell 16 viser kravspesifikasjonene gruppen ønsker ha med i prosjektet før brukerundersøkelsen ble gjennomført.

Tabell 16: Foreløpige krav til caset

Prioritet	Nr	Kravspesifikasjon
Må	1.0	Oversikt over banene (recycler)
	1.1	WCAG-prinsipper
	1.2	Velkommenside - første gang bruker åpner appen
	1.3	Frisbee-loader
	1.4	Vise lokal værmelding for valgt område (temp, vind, nedbør)
	1.5	Vise vanskelighetsgrad for hver bane
	1.6	Vise banekart
	1.7	Kartdata og kall fra API skal mellomlagres på enheten under kjøring
	1.8	Regler-side
	1.9	Egen logo for appstart (vurdere gif)
	1.10	Poengkort (navn, poeng)
Bør	2.0	"Vis i nærhetenknapp
	2.1	Tilby sortering av baner (vanskelighetsgrad, vær)
	2.2	Tilby søk i fritekst blant baner
	2.3	Rating av baner
	2.4	Km-avstand fra bruker
	2.5	Legge til bane som favoritt
	2.6	Brukers posisjon i kartet
	2.7	Liste over alle baner i Viken og Oslo
	2.8	Diverse værvarsler
	2.9	Regler til spillet
	2.10	Nullstill zoom inn/ut kart
	2.11	Tid per runde
	2.12	Pågang av andre brukere
	2.13	Opprette profil - lagre status/rekorder
	2.14	Info om parkering
	2.15	Sortere baner basert på avstand, pågang, rating, kollektiv
	2.16	Satellitt/terreng
Kan	3.0	Lagre poeng
	3.1	Avstand til nærmeste kiosk/butikk
	3.2	UV
	3.3	Banekart Google Earth (polygon)
	3.4	Flerne baner
	3.5	Nærmeste kiosk/badevann i forhold til banen
	3.6	Lagre/slette favorittbane
	3.7	Legge igjen egen review
	3.8	Dark mode
	3.9	Instillinger (eks. clear history, dark mode, profilinstillinger)
	3.10	Ha pakeliste til tur
	3.11	Måle avstand på kast (ala udisc)
	3.12	Videoer med teknikker / tips og triks
	3.13	Legg til ekstra spillere (favoritter i poengtabellen)
	3.14	Review av hver bane