

IN2000 - Software Engineering med prosjektarbeid

Prosjektrapport Team 4 21.05.2022

Soleklart

Mathias Gretland Ellingsen (*mathige*), Lina Filipaviciute (*linafi*), Parkavan Illavalagan (*parkavai*), Thomas Strandlie Johannessen (*thomsjoh*), Liv Hilde Sjøflot (*livhsj*), Ragnhild Margareta Wengaard (*ragnhmw*)

Veiledere:

Emil Damsgård Knutsen, Emil Eriksmoen Stensland

Faglærere:

Yngve Lindsjørn, Viktoria Stray, Henrik Løvold, Stein Michael Storleer





Innholdsfortegnelse

1 Presentasjon	5
1.1 <i>Introduksjon</i>	5
1.2 <i>Om Soleklart</i>	6
1.3 <i>Teamet</i>	6
2 Brukerdokumentasjon	7
2.1 <i>Målgruppe</i>	7
2.2 <i>Funksjonalitet</i>	8
2.2.1 Skjermbilder	8
2.2.2 Beskrivelse av tjenesten	10
2.2.3 Dataressurser	11
2.2.4 System og tilgjengelighet	12
2.3 <i>Universell utforming</i>	12
2.3.1 Oppfattbart	12
2.3.1.1 Ikke-tekstlig innhold	12
2.3.1.2 Skillbart	12
2.3.2 Mulig å betjene	13
2.3.3 Forståelig	13
2.3.4 Robust	13
3 Kravspesifikasjon og modellering	13
3.1 <i>Funksjonelle krav</i>	13
3.2 <i>Use case diagram</i>	15
3.3 <i>Aktivitetsdiagram</i>	16
3.4 <i>Sekvensdiagram</i>	19
3.5 <i>Ikke-funksjonelle krav</i>	20
4 Produktdokumentasjon	22
4.1 <i>Grunnleggende informasjon</i>	22
4.1.1 Oversikt	22
4.1.2 Teknologier	22
4.1.2.1 Utvikling	22
4.1.3 Mappediagram	22
4.1.4 Klassediagram	23
4.1.5 Beskrivelse av eksterne API-er	24
4.1.5.1 Geografiske informasjonssystemer	24
4.2 <i>Systemarkitektur og objektorienterte prinsipper</i>	24
4.3 <i>Kvalitetsegenskaper</i>	25
4.3.1 Inndatavalidering og feilhåndtering	25
4.3.2 Pålitelighet	26
4.3.3 Vedlikeholdsvennlighet	26
4.3.4 Svartid og internetttilkobling	27
5 Testdokumentasjon	28
5.1 <i>Testmetoder</i>	28
5.2 <i>Formål</i>	28



5.3	<i>Verktøy</i>	29
6	Prosessdokumentasjon	30
6.1	<i>Smidige utviklingsmetoder</i>	30
6.1.1	Valg av smidige metoder	30
6.1.2	Innledende fase	31
6.1.3	Mellomfasen	32
6.1.4	Avsluttende fase	34
6.2	<i>Verktøy</i>	34
6.2.1	Planlegging, dokumentasjon og kode	35
6.2.1.1	Trello	35
6.2.1.2	Android Studio	35
6.2.1.3	GitHub	35
6.2.1.4	Google Drive	36
6.2.1.5	Microsoft Office	36
6.2.2	Kommunikasjon	36
6.2.2.1	Discord	36
6.2.2.2	Messenger	37
6.2.2.3	Teams	37
6.2.3	Modellering og design	37
6.2.3.1	Draw.io	37
6.2.3.2	Miro	37
6.2.3.3	Proto.io	37
6.2.4	Datainnsamling og analyse	38
6.2.4.1	UiO Nettskjema	38
6.2.4.2	Geovisualisering	38
6.3	<i>Produktvisjon og valg av case</i>	38
6.3.1	Valg av case	38
6.3.2	Utvikling av visjon	39
6.3.3	Interessenter og målgruppe	40
6.3.4	Valg av navn	41
6.4	<i>MVP og kravspesifikasjon</i>	42
6.4.1	MVP – <i>minimum viable product</i>	42
6.4.2	Kravspesifikasjon	43
6.4.3	Retrospektivt blikk på MVP	44
6.5	<i>Valg av arkitektur</i>	45
6.5.1	Diskusjon i tidlig fase	45
6.5.2	Valg av design pattern	46
6.5.3	Arkitekturdesign	47
6.5.3.1	Retrospektivt blikk på arkitekturarbeid	48
6.6	<i>Visuell utforming</i>	49
6.7	<i>Universell utforming</i>	50
6.7.1.1	WCAG 2.1-krav	50
6.7.1.2	Brukbarhet	50
6.7.1.3	Nattmodus	51
6.7.1.4	Farger og fargeblindhet	51
6.8	<i>Brukerundersøkelser</i>	53
6.8.1	Kartleggende spørreundersøkelse	53
6.8.1.1	Metode	53
6.8.1.2	Gjennomføring	53
6.8.1.3	Resultater	54
6.8.2	Etnografisk undersøkelse	54
6.8.2.1	Metode	54



6.8.2.2	Gjennomføring	55
6.8.2.3	Resultater	55
6.8.3	Spørreundersøkelse knyttet til visuelle elementer	55
6.8.3.1	Metode	55
6.8.3.2	Gjennomføring	55
6.8.3.3	Resultater	56
6.8.4	Brukbarhetstesting	56
6.8.4.1	Metode	56
6.8.4.2	Gjennomføring	56
6.8.4.3	Resultater	57
6.9	<i>Videre utvikling</i>	59
6.9.1	Funksjonalitet og brukeropplevelse	59
6.9.2	Teknisk gjeld	59
7	Refleksjon	61
7.1	<i>Felles situasjonsforståelse</i>	61
7.2	<i>Sykdom og covid-19</i>	61
7.3	<i>Tidsbruk</i>	62
7.4	<i>Kommunikasjons- og arbeidskanaler</i>	63
7.5	Å være interaksjonsdesigner i IN2000	63
7.6	<i>Avsluttende refleksjon</i>	64
8	Kilder og referanser	65
9	Vedlegg	67
9.1	<i>WCAG 2.1</i>	68
9.2	<i>Mappeoversikt</i>	70
9.3	<i>Rapport fra Sprint Review 2 nettskjema</i>	71
9.4	<i>Prosjektkalender</i>	73
9.5	<i>Brukerhistorier</i>	74
9.6	<i>Diskusjon om design patterns</i>	75
9.7	<i>Visuell utforming</i>	76
9.8	<i>Spørreundersøkelse knyttet til visuelle ikoner</i>	80



1 Presentasjon

1.1 Introduksjon

Applikasjonen Soleklart er utviklet i forbindelse med et skoleprosjekt i faget *IN2000 Software engineerig med prosjektarbeid*, ved Universitetet i Oslo. Arbeidet presentert i denne rapporten er utført av *Gruppe nr. 4* som består av seks fulltidsstudenter ved Institutt for informatikk. Utviklingen av applikasjonen og utarbeidelse av rapporten har foregått over tidsperioden fra 27. februar 2022 til 21.mai 2022. Gjennom denne perioden har gruppen fått veiledning fra veilederne Emil Eriksmoen Stensland og Emil Damsgård Knutsen.

Ideen til applikasjonen Soleklart er utarbeidet og implementert av teamet selv, og utviklingen har foregått i henhold til gitte kode- og teknologikrav.

Denne rapporten har som formål å presentere applikasjonen og arbeidet som ble utført under utviklingen. Rapporten består av åtte hovedkapitler som hver har til hensikt å belyse ulike aspekter ved løsningen og utviklingsprosessen. Av hensyn til leserne av denne rapporten, er kapitlene skrevet med forskjellige perspektiver, og det vil derfor bli noe overlapping, og et tema kan bli nevnt flere ganger. Her følger en gjennomgang av hva man vil finne under de forskjellige hovedkapitlene, samt hvilke lesere kapitlene er tiltenkt.

I *kapittel 1* presenteres prosjektet og teamet. *Kapittel 2* omhandler brukerdokumentasjon som er ment for lesere uten tekniske forkunnskaper. Blant annet finner man en gjennomgang av løsningens funksjonaliteter, hvem som er løsningens målgruppe, samt hvilke hensyn som er tatt i forhold til universell utforming

Kapittlene 3-5 tar utgangspunkt i at leseren har kjennskap til løsningen og dens funksjonalitet, samt at leseren er kjent med objektorienterte prinsipper og kode. *Kapittel 3* tar for seg kravspesifikasjon og modellering for løsningen. I *kapittel 4* finner man produktdokumentasjon som klassediagram, beskrivelser av API-er og appens kvalitetsegenskaper. *Kapittel 5* tar for seg testdokumentasjon for løsningen.

Prosessdokumentasjonen, som er av særlig viktighet for å forstå bakgrunnen for avgjørelsene tatt under utviklingsprosjektet, blir beskrevet i *kapittel 6*. Her vil man finne begrunnelser og resonnemerter for avgjørelser som har funnet sted under utviklingsprosessen. Eksempelvis finnes en kort beskrivelse av målgruppen i *kapittel 2*, mens *kapittel 6* beskriver dette i større grad. Kapittelet gir også en gjennomgang av hvilke verktøy som er benyttet gjennom utviklingsprosessen, med en kategorisering av disse, samt hvilke erfaringer gruppen har gjort seg rundt bruken av de forskjellige verktøyene.

I *kapittel 7* finner man gruppens refleksjoner knyttet til hvordan prosjektarbeidet har gått sett opp mot de forventningene de hadde, hvilke erfaringer gruppen har gjort seg, samt hvilke forbedringer som kan implementeres ved en eventuell videreføring av prosjektet.



1.2 Om Soleklart

Soleklart er en applikasjon for Android-telefoner. Appen gir brukeren informasjon om tidspunkt og værvarsel (skydekke, temperatur, nedbør, vind og svevestøv) for soloppgang og solnedgang på en valgt lokasjon. For dem som vil se stjerne tilbyr Soleklart også en værvarsel to timer etter solnedgang. Soleklart har en tilleggsfunksjonalitet med ekstern kart over lokasjoner med gode solforhold ved soloppgang og solnedgang. Applikasjonens bruksopplevelse kan optimaliseres ved mulighet til å bytte kartstil og bruk i nattmodus. Samlet sett tilbyr Soleklart den viktigste informasjonen for at brukeren skal ha muligheten til å kunne planlegge en opplevelse, tur eller date i stemningsfullt lys.

1.3 Teamet

Teamet består av studenter fra linjene Programmering og systemarkitektur (PROSA) og Design, bruk og interaksjon (DESIGN). *Tabell 1* viser alle teamets medlemmer, studieretning samt en kort beskrivelse mens *Figur 1* er bildet av hele teamet. I tillegg til det presenteres det en “*stinky fish*” for hvert medlem. “*Stinky fish*” er en metafor for en ulempe eller et problem man har. I et gruppeprosjekt er det viktig å være ærlige om også de mer negative sidene ved seg selv, for lettere å kunne planlegge prosessen, gjøre samarbeidet enklere, og unngå konflikter (Bratteteig, 2021, s.292). Gruppen mente en slik ærlighet og konkretisering ville gjøre det lettere å ta hensyn til hverandre, men også å være bevisst sine egne begrensninger.

Tabell 1: oversikt over gruppens sammensetning med informasjon om studieretning, kort beskrivelse og “*stinky fish*”.

Navn	Linje	Beskrivelse	“ <i>Stinky fish</i> ”
Mathias	PROSA	Gladlaks som liker å kode, med god erfaring i å bruke git. Har noen prosjekter på siden, og er veldig glad i å progge i React. Stiller utrolig svakt med alt som innebærer å tegne.	Mangel på å forme en god struktur.
Lina	PROSA	Tidligere utdannet bioingeniør med noen års arbeidserfaring. Har en hel haug med fritidsprosjekter innenfor kunst og programmering. Hun har stor arbeidskapasitet, mye nysgjerrighet og det beste hun vet er å lære nye ting (og google ting).	Ofte har for mye fokus på effektivitet og er en kronisk micromanager.
Liv	DESIGN	Har gjennomført et stort prosjekt innen design hvor man satte brukernes behov og reell brukermedvirkning i fokus. Liker å organisere, samarbeide, fordele oppgaver og kan bidra med å forfatte rapport slik at denne er helhetlig og oversiktlig.	Beslutningsvegring
Parkavan	PROSA	Har god kompetanse innen diverse emner knyttet til programmering. Liker når oppgaver er strukturert og klargjort frister innad oppgavene. I tillegg til god samhandling mellom medlemmer i teamet. Er ikke flink med design og oppsett.	Pleier å minne andre om å holde frister og å planlegge ting godt på forhånd.



Ragnhild	DESIGN	Har tidligere utdanning i kreative og praktiske fag, og har gjennomført et stort gruppeprosjekt innen interaksjonsdesign. Er glad i innsikts- og designarbeid, lister og planer.	Har barn i barnehagealder, og er derfor ikke alltid like fleksibel med tanke på arbeidstid.
Thomas	PROSA	Jobber på sitt beste sent på kvelden, og foretrekker å progge da! Glad i å feilsøke godt gjemte "bugs". Har publisert spill på play store, og app store med Unity og C#.	Etter lengre perioder med koding, kobler hjernen helt ut og kommunikasjonsevner "reduseres".

Gruppen har ikke definert formelle roller i starten av prosjektet, men har valgt å ha en dynamisk rollefordeling, der rollene endres etter behovet og situasjonen i hver enkelt sprint. Likevel var det en forventning om at arbeidsoppgavene ville tildeles iht. medlemmernes faglige ekspertise.



Figur 1: Fra venstre bak: Thomas, Mathias, Ragnhild, Liv, Parkavan, Lina.

2 Brukerdokumentasjon

2.1 Målgruppe

Målgruppen til Soleklart er myndige innbyggere i Norge, med en gjennomsnittlig teknisk kompetanse, og en interesse for å oppleve soloppganger og solnedganger. Målgruppen har to undergrupper: en gruppe med gjennomsnittlig interesse for natur og friluftsliv, og en gruppe med over gjennomsnittlig interesse for natur og friluftsliv.

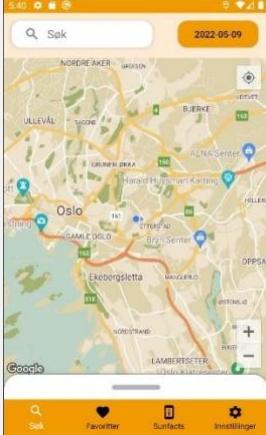
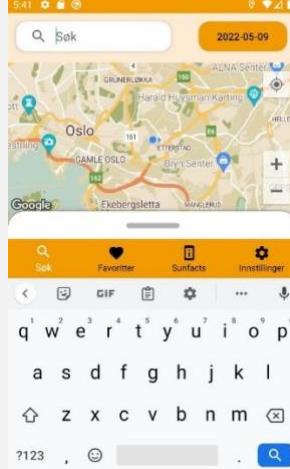


2.2 Funksjonalitet

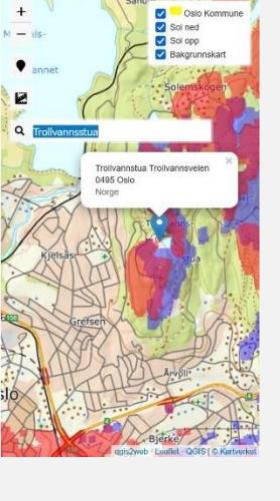
2.2.1 Skjermbilder

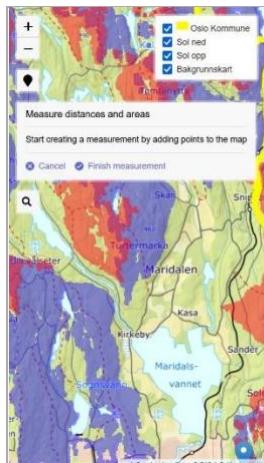
Tabell 2 viser skjermbildene og tekstlig beskrivelse av Soleklart sin funksjonalitet.

Tabell 2: skjermbilder og funksjonalitet i appen.

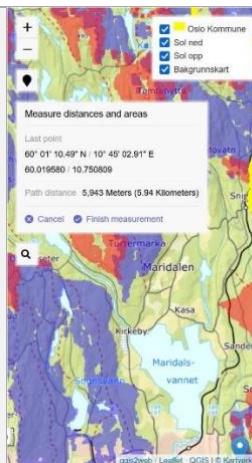
 <p>Soleklart</p>	<p>Ved oppstart vises en <i>splash screen</i> med logo og appens navn. Den første gangen appen tas i bruk, føres man også gjennom en <i>onboarding</i> som viser hvordan appen fungerer. Denne kan man hoppe over dersom man ønsker det.</p>	 <p>Menyen viser ulike hovedfunksjoner: Søk, favoritter, sunfacts og innstillinger. Aktiv meny-knapp: <i>Søk</i></p>	<p>Det vises et kart, og dagens dato er forhåndsvalet i øvre høyre hjørne. Man kan velge en lokasjon på 3 ulike måter:</p> <ol style="list-style-type: none"> 1. Zoome, enten med fingre eller knapper, dra kartet med fingrene, og klikke på et sted. 2. Velge sin egen lokasjon ved å trykke på ikonet for GPS i øvre høyre hjørne (dersom GPS-informasjon deles med appen). 3. Søke med tekst i input-tekstboksen.
 <p>Et tastatur åpnes ved trykk i input-tekstboksen. Når stedsnavnet er skrevet, startes søket ved å trykke på søkesymbolet i tastaturet.</p>	<p>Resultatet vises i en boks som dukker opp over kartet. Denne kan trekkes ned og skjules.</p>	 <p>Resultatet viser klokkeslett og værvarsel for soloppgang, solnedgang, og 2 timer etter solnedgang.</p>	<p>Symbolene viser vær og skydekke, temperatur, nedbør, vind og mengden svevestøv (PM10). Ikonene for vær endres basert på varselet. Ikonet for svevestøv/ luftkvalitet endrer farge i tråd med standarden fra Miljødirektoratet. Dersom luftkvalitetsmålinger ikke er tilgjengelig, vil ikonet være grått.</p>



 <p>Ved klick på knappen med dato (øverst i høyre hjørne), åpnes en kalender. Man har mulighet til å velge en annen dato for værvarselet syv dager frem i tid. Valgt dato er markert i blått. For å angre, kan man trykke på den allerede valgte dato. Ved trykk på en dato, blir dato valgt, og kalenderen lukkes. Resultater for det (tidligere) valgte stedet og valgt dato vises.</p>	<p>Aktiv meny-knapp: Favoritter</p> <p>Det er ikke implementert kode for å lagre favoritter, dette er kun en illustrasjon av hvordan det kunne sett ut.</p>
 <p>Aktiv meny-knapp: Sunfacts</p> <p>Dette er et rullefelt med lenke til et solkart, en lenke til råd om tiltak ved ulike nivåer av svevestøv, og korte fakta om sol.</p>	<p>Aktiv meny-knapp: Sok</p> <p>Ved klick på en favoritt, vil man sendes til <i>søk</i>, og resultatene for favorittens lokasjon vil vises.</p>
 <p>Brukeren kan søke etter steder og adresser i kartet. Kartet vil da panorere og zoom inn på dette stedet.</p>	<p>Sokkart åpnet i nettleser</p> <p>Sollkartet viser et Norgeskart, men solinformasjon er (i dette prosjektet) kun tilgjengelig for Oslo.</p> <p>Sollkartet viser solforholdene ved ulike steder i Oslo ved soloppgang og solnedgang (på sommerstid).</p>
	 <p>Ved å trykke på knappen med kart-markøren, blir brukeren bedt om å tillate deling av gps-informasjon. Dersom dette tillates, vil brukeren se sin egen posisjon på kartet.</p>



Ved klick på måle-ikonet, vil brukeren kunne utføre målinger i kartet.

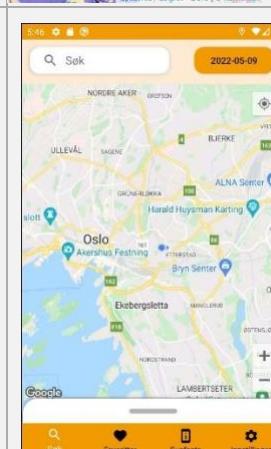


Brukeren klikker på et selvvalgt antall punkter i kartet, og får så en utregning av distanse, samt areal og omkrets dersom det er minst 3 punkter.

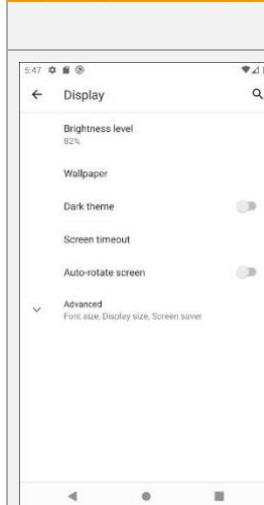


Aktiv meny-knapp: *Innstillinger*

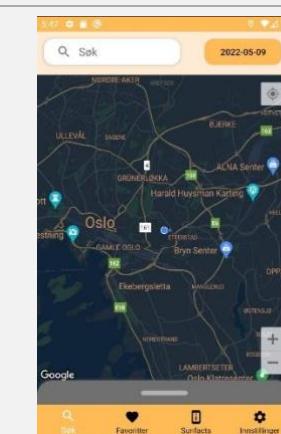
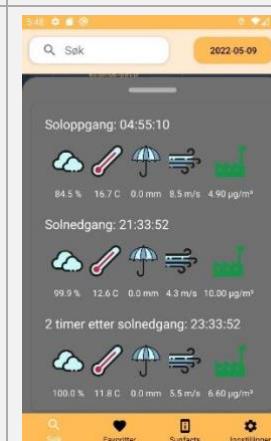
Brukeren kan velge mellom ulike kartstiler ved å aktivere de ulike bryterne.



Ved å velge kartstilen *standard*, vil kartet i *søk* vises som dette.



Dersom brukeren aktiverer *Nattmodus* på telefonen sin (Innstillinger/ Display), vil Soleklart også vises i mørke farger. Dette vises på samme rad, i kolonnen til høyre.



Solkartet er også tilgjengelig på GitHub - Pages (Wengaard, 2022).

2.2.2 Beskrivelse av tjenesten

Soleklart sin hovedfunksjon er å fremstille informasjon om værforhold for solnedgang og soloppgang på en oversiktlig og brukervennlig måte. I denne versjonen av appen får brukeren informasjon om værforholdene to timer etter solnedgang. Optimalt sett ville dette tidspunktet basere seg på når det faktisk er mørkt nok til å se stjerner, men dette varierer mye avhengig av årstiden, og slik informasjon er ikke tatt med i denne omgang.



Applikasjonen er utviklet med tanke på enkelhet, hastighet og brukervennlighet som hovedfokus, da gruppen mente at dette var essensielt for å oppnå et best mulig produkt. For å oppnå dette, kan hovedfunksjonen utføres ved kun å bruke hovedsiden *søk*. For enkelhetens skyld består derfor *søk* av kart-, søker-, kalenderfunksjon og fremstillingen av resultater, slik at man i praksis kun trenger å benytte seg av én side for å utføre det applikasjonen er laget for. Soleklart er også utviklet slik at for hver funksjon man kan utføre i applikasjonen, er det enten et ikon eller en tekst som tydelig beskriver hva den funksjonen gjør. Hensikten med dette, er at applikasjonen skal være så enkel som mulig å ta i bruk.

Hastigheten til systemet har stor innflytelse på hvor brukervennlig og enkel applikasjonen er å ta i bruk. Kartfunksjonen som tar inn og leverer mesteparten av dataen som fremstilles, spiller derfor en stor rolle i hvor rask selve applikasjonen er å bruke. Derfor bruker applikasjonen den nåværende posisjonen til brukeren og dagens dato for å gjøre et kall på API-ene med en gang applikasjonen lastes inn, uten å bruke selve kartfunksjonen. Dette ble implementert med en tanke om at brukeren ofte vil ha akkurat denne typen informasjon, og som et resultat, vil brukeren kunne få dette etter noen få sekunder. Disse resultatene presenteres ikke umiddelbart, men vil vises når brukeren trykker i kartet (på det som da gjerne er deres omrentlige posisjon), eller på GPS-symbolet. Ved å gjøre det på denne måten, får brukeren en følelse av å ha kontroll over systemet. Man unngår også at de føler at applikasjonen er insisterende, når de får muligheten til å velge en annen lokasjon, uten å at resultatet for egen lokasjon blir presentert mot deres vilje.

Luftkvaliteten som oppgis i værvarselet er PM10, og er konsentrasjonen svevestøv i lufta. Dette oppgis egentlig i døgnmiddel, men er omregnet til timemiddel basert på statistikk. Det finnes flere forurensningsklasser, men som en konsekvens av at den gamle bilparken (diesel) byttes ut med utslippsfrie biler, eller biler med strengere krav til utslipp, er NOX-nivåene stadig på vei nedover. Svevestøv kommer fra alle typer biler, og gjør at PM10 virket mest relevant i dette prosjektet.

Solkartet gjør det lettere for brukeren å finne de beste forholdene for å se sola idet den stiger over eller forsvinner under horisonten. Alle steder vil få sol relativt kort tid etter soloppgang, men dersom man ikke har direkte siktlinjer mot sola (fordi man for eksempel befinner seg på feil side av en ås), vil man bare oppleve at himmelen lysner. Det samme gjelder ved solnedgang. Med solkartet kan brukerne kunne se hvor de beste forholdene er, uavhengig av om de ønsker kun soloppgang eller solnedgang, eller lokasjoner med gode forhold for begge deler.

2.2.3 Dataressurser

Data presentert i applikasjonen Soleklart er hentet fra flere eksterne API-er. Data knyttet til værforhold er basert på informasjon hentet fra Locationforecast og NILU API-ene. Locationforecast er et API utviklet av Meteorologisk institutt ved Universitet i Oslo. NILU API er utviklet av Norsk institutt for luftforskning. Tidspunktene for soloppgang og solnedgang er hentet fra Sunrise API (Meteorologisk institutt, UiO). Soleklart sin kartfunksjonalitet er basert på Google Maps SDK. Geokoding av stedsnavn (gitt som brukerinput) til geografiske koordinater utføres ved bruk av Geocoder-klassen som bruker Geocoding API fra Android.



2.2.4 System og tilgjengelighet

Soleklart er en app for Android-telefoner. Appen er laget for API-nivå 29. Det vil si at den kjører på Android-versjon 10, som kom i september 2019. Dette er relativt nye telefoner, og når ikke en så bred brukergruppe som man kunne ønske. Grunnen til at API-nivå 29 er valgt, er at Soleklart har innebygget nattmodus-innstillinger som kun tilbys igjennom dette nivået eller høyere.

Innholdet i appen er helt avhengig av internetttilkobling, da værvarsler og solkart ikke kan presenteres uten dette. GPS-tilkobling er frivillig, men vil kunne øke brukbarheten.

2.3 Universell utforming

Forskrift om universell utforming av informasjons- og kommunikasjonsteknologiske (IKT)-løsninger stiller krav om at IKT-løsninger skal utformes i samsvar med Web Content Accessibility Guidelines (WCAG) 2.0 eller nyere. WCAG stiller krav til at produktet skal være lett oppfatte, mulig å betjene, forståelig og robust. Hver av disse kravene i WCAG har mange detaljerte suksesskriterier, og forskriften krever at minst 35 av 61 slike suksesskriterier oppfylles. WCAG 2.1 vil bli gjeldende fra februar 2023, og har 78 kriterier. Dette prosjektet resulterer ikke i et system som skal publiseres for allmenn bruk, og det er da anbefalt, men ikke pålagt å oppfylle WCAGs suksesskriterier. Her følger en kort beskrivelse av hvordan Soleklart stemmer overens med WCAG. I kapittel 6.7 *Universell utforming* er arbeidet med suksesskriteriene beskrevet, og i vedlegg 9.1 WCAG 2.1 finner man en komplett oversikt over hvilke suksesskriterier som er implementert og ikke.

2.3.1 Oppfattbart

2.3.1.1 Ikke-tekstlig innhold

Brukergrensesnittet skal kunne oppfattes av brukerne. Soleklart har ingen lyd eller video, og kan derfor se bort fra kravene knyttet til slikt innhold. Selv om det er flere måter å søke etter et sted, enten med tekst, egen posisjon, eller ved å trykke i kartet, kommer man ikke utenom det faktum at et kart er svært visuelt og lite kompatibelt med skjermlesere. Det gjelder også kartet for de beste stedene for soloppgang og solnedgang. Denne informasjonen er kun tilgjengelig visuelt.

2.3.1.2 Skillbart

Det skal være lett å skille fra forgrunn og bakgrunn fra hverandre, og fargene har derfor gjennomgått en kontrastanalyse. Fargene er fargeblindvennlige, og ingen informasjon formidles med kun farge. Brukeren kan velge mellom ulike fargestiler på kartene, og appen tilpasser seg dersom telefonen blir satt i nattmodus.

Det er krav til skriftstørrelser, men disse har vært vanskelig å følge, da feltene teksten skal inn i, ikke er så store. En bruker vil kunne justere størrelsen på visningen på sin egen mobil, men det kan resultere i at tekster og elementer ikke lenger passer inn i oppsettet feltene sine. Brukeren kan heller ikke velge skjermens retning som kan være landskap eller portrett.



2.3.2 Mulig å betjene

Soleklart kan aksesseres på Android smarttelefoner, der de fleste bruker skjermberøring. Knappene er i en størrelse godkjent av Android Studio, og bør dermed være store nok for interaksjon. Det er ingen tidsbegrensninger for å utføre handlinger i systemet, og det er tydelig hvor i systemet brukeren befinner seg. Tastaturnavigasjon er ikke implementert.

2.3.3 Forståelig

Språket er enkelt og uten sjargong, og ordbruk antas til å passe brukere med ulik språkkompetanse. Oppsett er forutsigbar og logisk, og endringer settes kun i gang av brukeren selv, dog uten ekstra bekreftelser. Det er tydelig hva slags input som forventes, og hvor man kan interagere med systemet. Assistanse i form av forslag til stedsnavn ved tekstbasert søk er ikke implementert. Dette gir en noe dårligere brukeropplevelse, og det ville være naturlig å inkludere i en senere iterasjon. Det medfører ellers ingen alvorlige konsekvenser. Resultatene i værvarselet er noe utydelig formulert, da noen av dem mangler forklarende ord.

2.3.4 Robust

WCAG krever at systemet skal være tilgjengelig for en rekke ulike assistansesystemer, som for eksempel skjermlesere. Det har ikke blitt prioritert i dette prosjektet.

3 Kravspesifikasjon og modellering

3.1 Funksjonelle krav

Applikasjonens Soleklart funksjonelle krav presenteres i *Tabell 3*. De funksjonelle kravene er delt opp i tre kategorier "må", "bør" og "kan" som indikerer prioriteten med hensyn til verdien av funksjonaliteten som kravet tilbyr. Verdien måles i kontekst av hvorvidt et oppfylt krav bidrar til å oppfylle applikasjonens formål. Kravene i "må"-kategorien er de som er mest essensielle og inngår i MVP (*minimal viable product*). "Bør"-kategorien dekker krav som anses som viktige, men deres oppfyllelse er ikke kritisk for applikasjonens formål. "Kan"-kravene har minst prioritet og sees som et supplement til de andre kravene. Kravene har også tildelt nummer, som i utgangspunktet er ment å gi en mer detaljert prioritering, som også er mer dynamisk og kan endres i større grad under utviklingen.

Under utførelse av dette prosjektet ble det implementert 20 av 30 funksjonelle krav, hvor alle av kravene i "må" kategorien og de fleste i "bør" kategorien er oppfylt.



Tabell 3: Oversikt over alle funksjonelle krav til applikasjonen Soleklart delt opp i kategorier etter prioritet. Funksjonelle krav som ble implementert i løpet av dette prosjektet er markert med uthevet skrift og gul farge.

Kategori	#	Kravspesifikasjon	Status
Må	1.1	Vise skydekke ved solnedgang og soloppgang for angitt dato og sted	+
	1.2	Vise tiden for solnedgang og soloppgang på angitt dato og sted	+
	1.3	Vise estimert luftkvalitet under soloppgang og solnedgang for angitt sted, basert på historisk data	+
	1.4	Applikasjonene skal ha egen logo og navn som vises i enhetens meny.	+
	1.5	Hente og vise brukerens nåværende posisjon på kartet	+
	1.6	Vise temperatur ved solnedgang og soloppgang for angitt dato og sted	+
	1.7	Velge lokasjon ved karttrykk	+
	1.8	Velge lokasjon ved tekst sok på stedsnavn	+
	1.9	Applikasjonen skal ha navigasjonsbar med minst 2 felt: hovedside og innstillinger	+
	1.10	Applikasjonen skal vise resultat for værforhold for den gitte datoен og stedet i et eget felt.	+
	1.11	Initielt hente standardpakke med data basert på brukerens nåværende posisjon og dagens dato	+
	1.12	Vise ikon for skydekke og luftkvalitet basert på innhentet data	+
Bør	2.1	Applikasjonen skal ha info-side med informasjon applikasjonen	+
	2.2	Brukeren skal kunne lagre favorittsted og gi eget navn til stedet.	-
	2.3	Brukeren skal kunne slette favorittsted	-
	2.4	Brukeren skal kunne skifte mellom nattmodus og dagmodus fra Android settings	+
	2.5	Kartdata og kall fra API mellomlagres på enheten under kjøring	-
	2.6	Loadingbar vises i resultat felt mens applikasjonen gjør API kall.	+
	2.7	Splashscreen vises når applikasjonen starter.	+
	2.8	Brukeren skal kunne skifte mellom minst to ulike kartstiler.	+
Kan	3.1	Applikasjonen skal vise Onboarding side ved første gangs bruk	+
	3.2	Applikasjonen skal vise informasjon om stjerner som er synlige etter solnedgang på den gitte dato.	-
	3.3	Få informasjon om stjernekonstellasjoner ved å la brukeren se gjennom mobilens kamera.	-
	3.4	Presentere en liste over nærliggende predefinerte parker og relevant informasjon	-
	3.5	Brukeren skal kunne bytte applikasjonens språk mellom norsk og engelsk.	-
	3.6	Brukeren skal kunne utføre en tekst sok etter parker i nærliggende områder	-
	3.7	Applikasjonen skal tilby informasjon om hvilke API-er er brukt til innhenting av data.	+
	3.8	Applikasjonen skal vise informasjon om synlige planeter etter solnedgang ved gitt dato.	-
	3.9	Applikasjonene skal presentere informasjon om sol og dens fenomener	+
	3.10	Velge nattmodus/ dagmodus fra applikasjon	-



3.2 Use case diagram

Figur 2 viser use case-diagrammet for brukstilfellet "Sjekk værvarslet for en annen dato ved trykk i kart". Dette brukstilfellet antas til å være hovedbrukstilfellet og vil dermed dominere i de fleste bruker interaksjoner med Soleklart. Diagrammet presenterer et overordnet bilde over interaksjonen mellom aktører og systemet Soleklart samt viser aktørenes mål og systemets skop. Den primære aktøren er brukeren, mens de sekundære aktørene er API-ene.

Tekstlig beskrivelse av brukstilfellet "Sjekk værvarslet for en annen dato ved trykk i kart" er som følge:

Use case "Sjekk værvarslet for en annen dato ved trykk i kart"

Prebetingelse:

*Brukeren befinner seg på skjermbildet for "Søk".
Brukeren har tilgang til internett.*

Postbetingelse:

Værvarslet for angitt tid og sted vises.

Hovedløp:

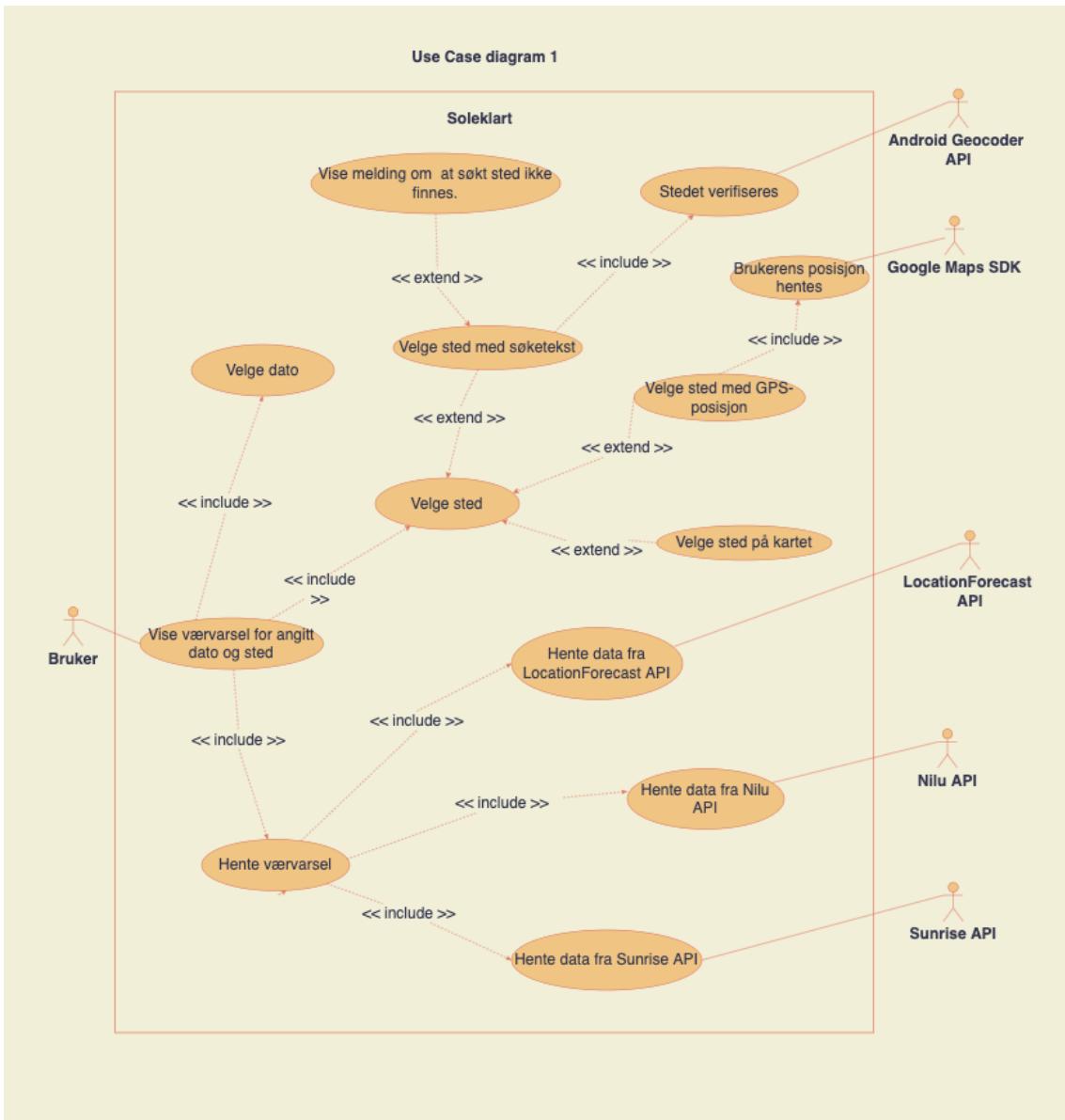
*Systemet presenterer dagens dato som forhåndsvalgt alternativ
Brukeren trykker på knappen med den valgte dato
Systemet presenterer et vindu med kalender
Bruker velger hvilken dato hen ønsker værvarslet for
Kalenderen lukkes, brukerens valg presenteres på knappen for dato
Brukeren drar kartet til ønsket lokasjon.
Brukeren trykker på et sted i kartet
Systemet henter informasjon fra de ulike API-ene.
Systemet presenterer informasjonen for brukeren i et pull-up-vindu.*

Alternativ flyt 1, steg 4:

- A4.1. Brukeren velger en dato lengre enn 7 dager frem i tid.
- A4.2. Systemet viser en toast der det forklarer at datoer er for langt frem i tid.

Alternativ flyt 2, steg 7:

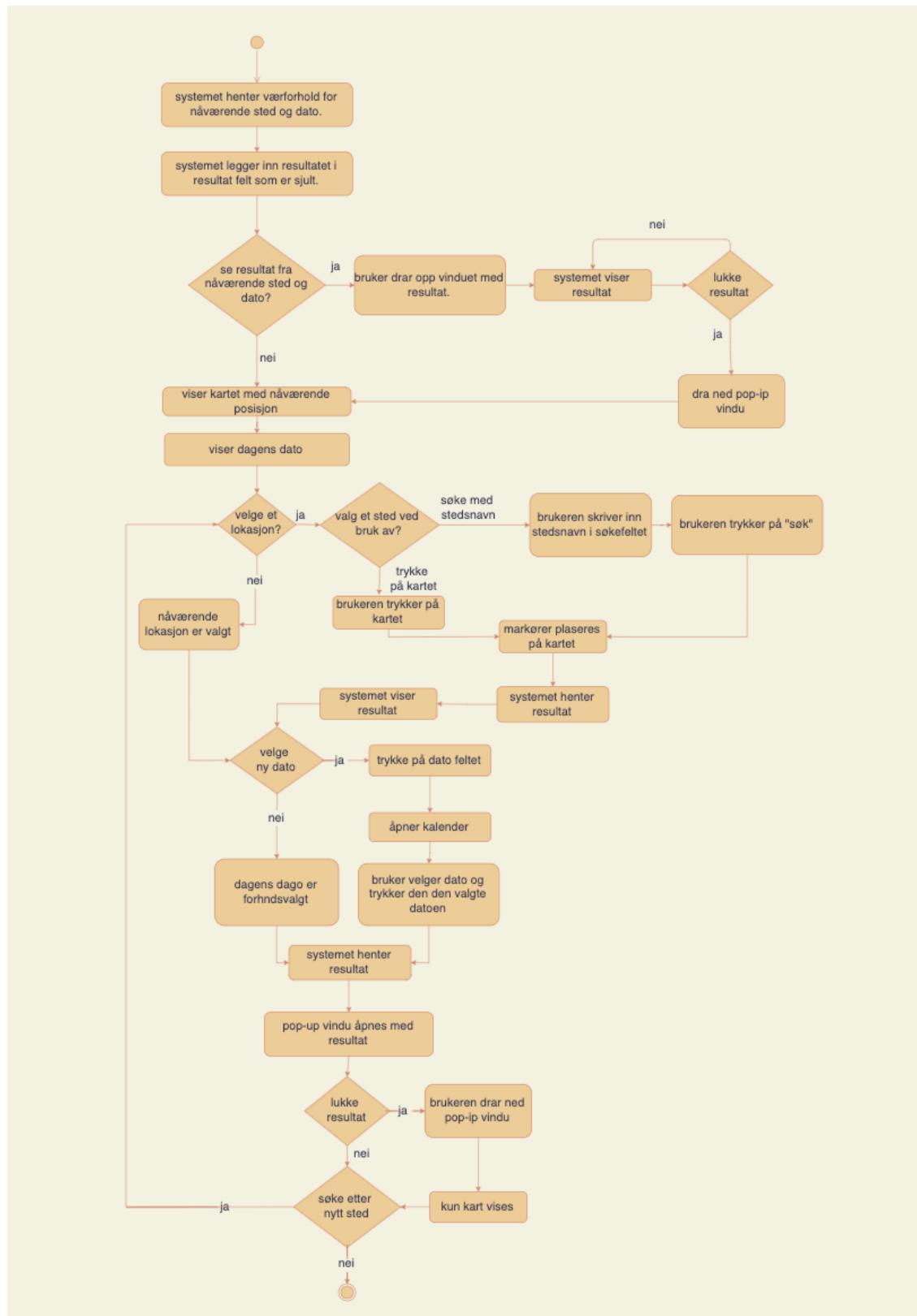
- A7.1. Brukeren trykker på et sted utenfor målområdet for NILU.
- A7.2. Systemet henter informasjon fra API-ene bortsett fra NILU.
- A7.3. Systemet presenterer informasjonen for brukeren i et pull-up-vindu, luftkvalitetsverdien vises som " - ".



Figur SEQ Figur 1* ARABIC 2: Use case diagram for "Sjekk værvarsel for en annen dato ved trykk i kart".

3.3 Aktivitetsdiagram

Dette delkapitlet presenterer systemet til Soleklart ved bruk av aktivitetsdiagrammet vist i *Figur 3*. Dette diagrammet er ment for å gi et overordnet innblikk i sammenhengen mellom handlinger, valg og systemets aktiviteter på en grafisk og strukturert måte.



Figur SEQ Figur 1* ARABIC 3: aktivitetsdiagram som viser kontrollflyten i systemet hvor brukeren skal søke opp værforhold.



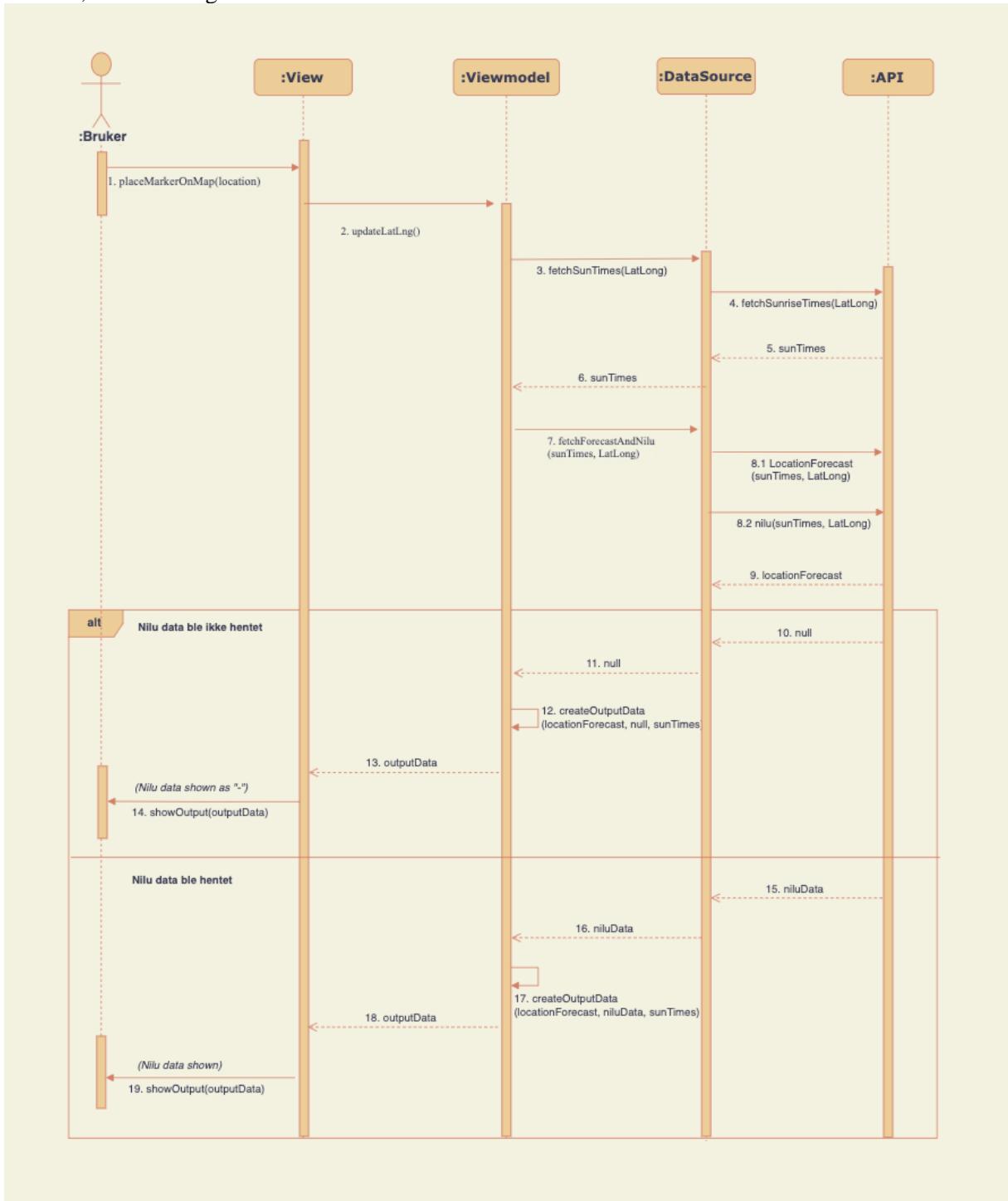
Diagrammet illustrerer kontrollflyten og kronologisk sammenheng i systemet hvor en bruker har som mål å hente informasjon om værforhold. En mer utfyllende tekstlig beskrive av systemflyten kan beskrives som følgende:

Systemets aktivitet starter med henting av værforhold for nåværende posisjon og sted. Nåværende posisjon og dagens dato vises for brukeren. Deretter har brukeren mulighet til å finne en ny lokalisasjon, enten ved å trykke på kartet eller søke med ord. Hvis det er valgt, vil systemet, uansett metode, oppdatere kartet med ny lokasjon, hente værforhold og vise resultat. Videre vil brukeren ha et valg om å endre dato – hvis valgt vil kalenderen vises, og etter at en dato er valgt, vil systemet igjen hente ny data og presentere det ved et pop-up vindu. Deretter kan brukeren velge å lukke resultatvinduet ved å dra det ned. Dersom brukeren ønsker å velge en ny lokasjon, kan det gjøres. Hvis ikke, avsluttes flyten.



3.4 Sekvensdiagram

Programmets flyt for brukstilfellet “Sjekk værvarsel for en annen dato ved trykk i kart” har blitt modellert ved bruk av sekvensdiagram (Figur 4). Diagrammet viser kronologisk interaksjon mellom hovedkomponenter i systemet. Figuren viser en hovedflyt, samt én alternativ flyt som er avhengig av vellykket API-kall på NILU API. Komponentene presentert i dette diagrammet er abstraksjoner av klassene, metodene og API-er.



Figur SEQ Figur 1* ARABIC 4: Sekvensdiagram for use case «søke værforhold ved trykk i kartet»



3.5 Ikke-funksjonelle krav

Tabell 4 viser ikke-funksjonelle krav fordelt på kravtyper, samt på hvilken måte disse kravene kan verifiseres og hvorvidt de er implementert eller ikke.

Tabell 4: Tabell med ikke-funksjonelle krav sortert etter type.

Kravtype	Kravspesifikasjon	Verifisering	Implementert
Pålitelighet	Systemet skal ikke ha en nedetid på over 1%, så lenge systemet har tilgang til internett	Løsningen er implementert uten avhengigheter av eksterne databaser.	Ja
	98% av API kall skal ikke vare lengre enn 15 sekunder	Testes via å kjøre automatisert statistisk test	Har ikke kjørt automatiserte statistiske tester, men få ytterlige tester har resultert i API kall som har vart lengre enn 15 sek.
	Applikasjonen skal ikke krasje på API nivå 29	Applikasjonen har blitt utviklet og testet med API 29, og fungerer fint.	Ja
	Applikasjonen skal ha minst 1 integrasjonstest.	Brukte Instrumented tests i Android studio	Ja, 5 integrasjonstester er implementert
	Applikasjonene skal ha minst 5 enhetstester.	Brukte JUnit i Android Studio	Ja, 10 enhetstester er implementert
	Applikasjonen skal ha kort opplæringstid.	Onboarding-instruksjoner skal gi tilstrekkelig opplæring i bruk av systemet, hvor mange % av brukere kan bruke systemet etter å ha lest instruksjoner.	Testet Onboarding med fem brukere, hvorav 100% evnet å bruke applikasjonen etter å ha fullført Onboarding
Bruknernavn	90% av brukere skal ikke bruke mer enn 2 min på å lese Onboarding instruksjoner.	Onboarding skal ha enkle beskrivelser og lite, men godt forklarende tekst. Brukertest gjennomføres for å kvalitetssikre	Ingen av de fem brukere som gjennomførte brukertestning brukte mer enn 2 min på Onboarding
	Applikasjonens utseende skal være responsiv og uavhengig av skjermstørrelse og skjermopløsning	Testet under utvikling med emulatorer i forskjellig skjermopløsning og skjermstørrelse	Dette gjelder for alle fragmenter i hoved-use case. Dimensjonene er definert ved bruk av SP og DP, i stedet for å angi PX
	Fargene i Applikasjonen skal være fargeblindvennlige (tilstrekkelig kontrast)	Brukte Coblis fargesimulator for å finne farger som ville gi tilstrekkelig kontrast	Alle farger som er brukt i appen er iht. Coblis fargesimulator
GU			



S i k k e r h e t s k r a v	Systemet skal ikke lagre personopplysninger om brukere.	Applikasjonen utvikles slik at ingen personopplysninger håndteres under bruk av applikasjonen	Ja, applikasjonen håndterer ingen personopplysninger, men bruk av Google Maps API medfører at data knyttet til bruksenhets lokasjon lagres av Google.
	Applikasjonen skal ikke lagre kryptert data	Applikasjonen er utviklet slik at ingenting lagres	Ja
	Leveransedato. Applikasjonen skal være ferdig 8.mai 2022	Utviklingsteamet setter intern frist som er tilpasset eksterne frister (eksamen)	Ja
	Systemet skal utvikles i Android studio	Android Studio har vært eneste utviklingsverktøy gjennom helle prosessen	Ja
	Applikasjonene skal kodes ved bruk av Kotlin og XML	Kotlin og XML har vært de eneste programmeringsspråkene gjennom helle prosessen	Ja
	Systemet skal ha eget navn og logo.	Teamet har gjennomført idémyldring og brukerundersøkelser for å finne passende navn og logo	Ja
	Applikasjonen skal benytte proxy-server ved bruk av API-er fra Meteorologisk institutt.	Proxy-server er benyttet	Ja
	Applikasjonen skal benytte minst ett API fra Meteorologisk Institutt	Bruker ett API fra Meterologisk institutt	Ja
	Koden skal være dokument ved bruk av KDoc	Utviklet ved bruk av KDoc	Ja
	Applikasjonen skal være utviklet ved bruk av MVVM design pattern	MVVM er implementert	Ja
e k r a v	Applikasjonen skal ha teknisk dokumentasjon.	Inkludert i rapporten.	Ja
	Applikasjonen skal ha brukerdokumentasjon.		Ja
	UI-komponenter synlige for brukeren skal inneholde norske ord (bokmål).	Norsk er implementert som språk i applikasjonen	Ja
	Variablene og kommentarene i koden skal være skrevet på engelsk.	Utviklet ved bruk av engelsk.	Ja
	Alle tekstrenger skal ekstraheres til String resources.	All tekst er ekstrahert.	Ja



4 Produktdokumentasjon

4.1 Grunnleggende informasjon

4.1.1 Oversikt

Applikasjonen Soleklart er utviklet i forbindelse med et skoleprosjekt i faget *IN2000 Software engineering med prosjektarbeid*, ved Universitetet i Oslo. Ideen til Soleklart er utviklet av teamet selv, men utviklingen har foregått i henhold til kravene gitt i faget. Utviklingen har foregått over en 10 ukers periode, fra 27. februar 2022, til 20.mai 2022. *Tabell 5* viser en oversikt over grunnleggende informasjon om Soleklart.

Tabell 5: Grunnleggende informasjon om applikasjonen Soleklart.

Navn	Soleklart
Filstørrelse	83MB
API-nivå	29
Programmeringsspråk	Kotlin, XML
Utviklingsmiljø	Android Studio Chipmunk 2021.2.1
Testverktøy	Android Studio og JUnit 4
Eksterne API-er	Locationforecast NILU Sunrise Goole Maps SDK Geocoder

Koden for solkartet er utviklet i QGIS. Solkartet er en del av funksjonaliteten i applikasjonen, men utviklet som et frittstående element, basert på en annen teknologi enn resten av applikasjonen. Solkartet er ikke et fragment i appen, men åpnes gjennom en nettleser. Av disse årsakene er solkartet utelatt fra diagrammer.

4.1.2 Teknologier

4.1.2.1 Utvikling

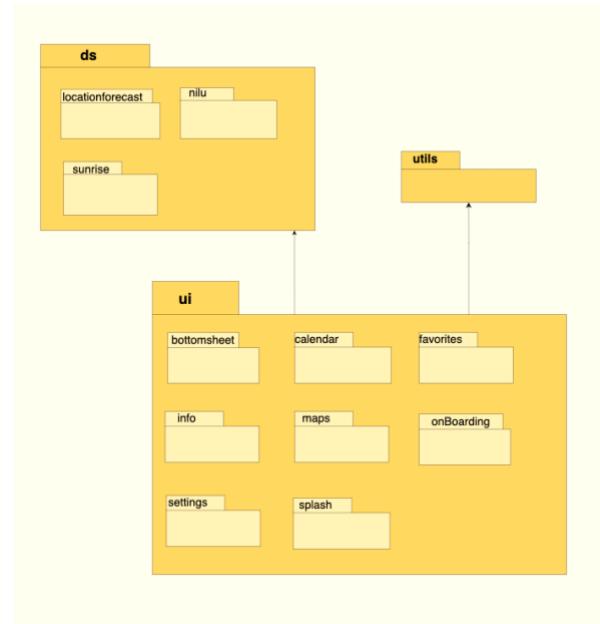
Programvare utvikling av applikasjonen Soleklart har blitt utført ved bruk av en rekke teknologier. Noen av de viktigste teknologiene er presentert i *Tabell 5*. HTTP-forespørsler ble utført ved bruk av Fuel 2.3.1 bibliotek. Deserialisering fra JSON til Kotlin ble utført ved bruk av GSON 2.9.0 bibliotek. JSON-filer til de ulike kartstilene ble laget ved bruk av *Google Maps Styling Wizard*.

4.1.3 Mappediagram

Simplifisert struktur og avhengigheter mellom klassene i applikasjonen

Soleklart presenteres ved bruk av UML

pakkediagram vist i *Figur 5*. Diagrammet viser logisk relasjon mellom de tre pakkene i høyeste nivå: *ds*, *ui* og *utils*, hvor det sees at pakken *ui* er avhengig av både *ds* og *utils* pakkene. Diagrammet viser også pakke innhold i *ds* og *ui* pakkene. Detaljert innhold i pakkene på alle nivåer er tilgjengelig i vedlegg 9.2 *Mappeoversikt*.



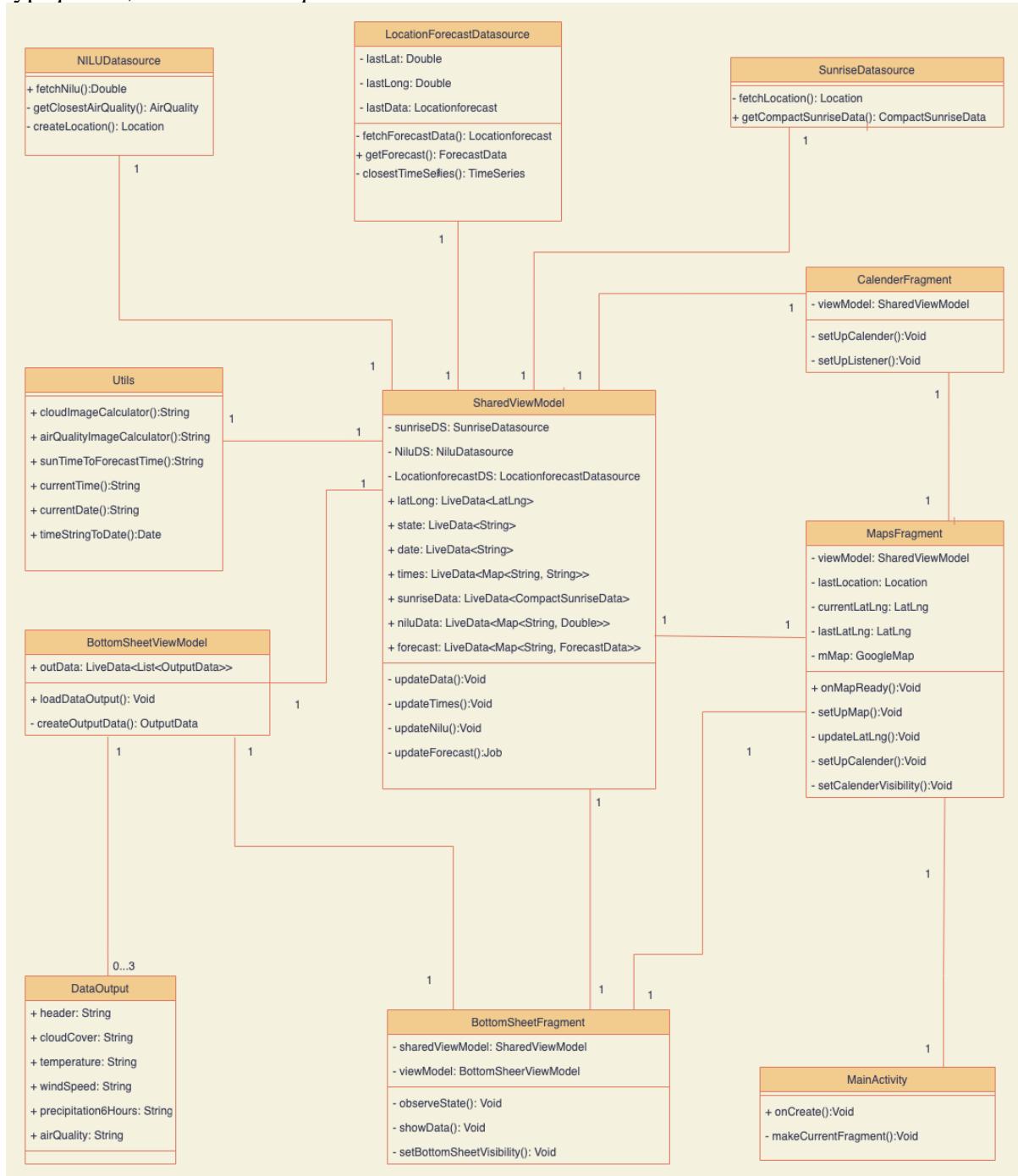
Figur 5: Mappediagram som viser mappehierarki i applikasjonen Soleklart.



4.1.4 Klassediagram

Figur 6 klassediagram inneholder alle de viktigste komponentene som er relevante for hovedbrukstilfellet "Sjekk værvarsel for en annen dato ved trykk i kart". Diagrammet viser den statiske strukturen av programmet og forhold mellom de ulike klassene i systemet. Den eneste relasjonen som presenteres i diagrammet er assosiasjon, vist med en enkel strek.

Hver klasse i diagrammet har navnefelt, variabelfelt og metodefelt. Mindre signifikante metoder og variabler er ikke med for å holde fokus på de sentrale komponentene som gir best innsikt i programmets struktur. Metodenes og variablene synlighet indikeres med "+", av type *public*, mens "-" viser *private*.



Figur 6: Klassediagram av klassene involvert i hoved-use case.



4.1.5 Beskrivelse av eksterne API-er

Soleklart sin kartfunksjonalitet er basert på Google Maps SDK. Maps SDK brukes i flere funksjoner: til visuell fremstilling av kartet, geokoding av koordinater gitt ved trykk i kartet, og visning av kartmarkør. Geokoding av stedsnavn (gitt som brukerinput) til geografiske koordinater utføres ved bruk av Geocoder-klassen som bruker Geocoding API fra Android.

Data knyttet til værforhold er basert på informasjon hentet fra Locationforecast og NILU API-ene. Locationforecast er et API utviklet av Meteorologisk institutt ved Universitet i Oslo, og har blitt benyttet til henting av verdiene temperatur ($^{\circ}\text{C}$), skydekke (%), vindhastighet(m/s) og nedbør (mm). NILU API er utviklet av Norsk institutt for luftforskning og brukes for henting av svevestøv ($\mu\text{g}/\text{m}^3$).

Verdiene som soloppgang og solnedgang er hentet fra Sunrise API (Meteorologisk institutt, UiO). Tidspunktene blir hentet i UTC offset (ISO 8601) format og analyseres av applikasjonen.

4.1.5.1 Geografiske informasjonssystemer

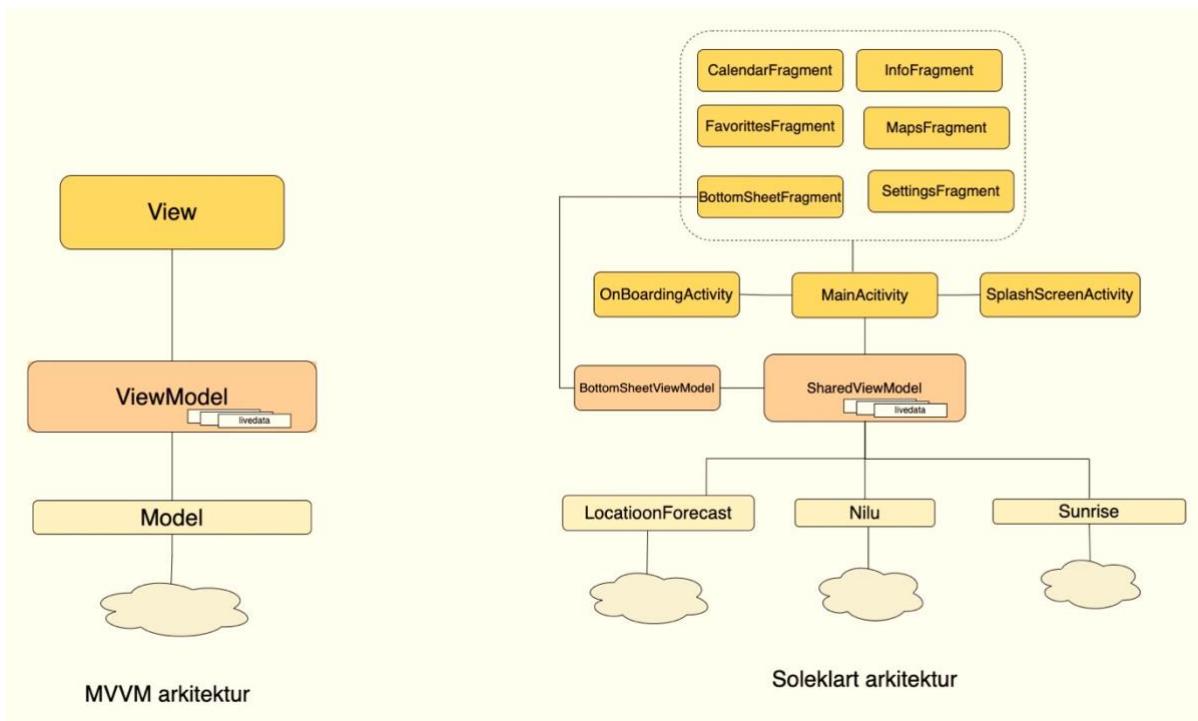
Solkartet er laget ved bruk av programmet QGIS. Bakgrunnskartet er *Kartverkets N50 topo*, og fylkesgrensen for Oslo er hentet fra Kartverkets "Administrative enheter fylker". Det er gjort en Viewshed-analyse basert på Kartverkets digitale overflatemodell (DOM), som inkluderer vegetasjon og bygninger, kombinert med gjennomsnittlige posisjoner for solen ved soloppgang og solnedgang sommerstid. For å forenkle lesingen av kartet, gjennomgikk rasterlagene en naboanalyse. Dette betyr at man i bygater direkte bak bygninger også vil få en markering som om man har direkte siktlinje til solen, til tross for at det kun gjelder når man står ved siden av eller foran bygningen. Det blir noe upresist, men større flater med fargekoding vil gi økt lesbarhet. Kartet er publisert på GitHub, i dimensjoner som passer de fleste smarttelefoner.

4.2 Systemarkitektur og objektorienterte prinsipper

Arkitekturen til applikasjonen Soleklart (*Figur 7*) er basert på *Model-View-ViewModel* (MVVM) design pattern. Arkitekturen ble utviklet med hensyn til prinsippene lav kompleksitet, skallerbarhet og enkelt vedlikehold, samt målene om lav kobling og høy kohesjon.

Lav kobling oppnås ved å separere komponentenes ansvarsområder. I Soleklart består *View*-komponenten av elementene som er av type *Activity* og *Fragment*. Disse elementene definerer grensesnitt for bruker interaksjon og inneholder ikke applikasjons logikk. *Model*-komponenter består av tre ulike klasser som har ansvar til å hente data fra API-ene – Locationforecast, NILU og Sunrise. Klassene *BottomSheetViewModel* og *SharedViewModel* danner komponenten *ViewModel*, som lenker sammen *Model* og *View*. *ViewModel* fungerer som en felles kommunikator ved å eksponere og konvertere data objekter fra *Model* og presentere disse til *View*.

MVVP prinsippene i Soleklart sin arkitektur støtter også høy kohesjon. Komponentenes inndeling i tydelig definerte ansvarsområder gjør at enhetene innen hver komponent, har



Figur 7: Model over MVVM arkitektur og diagrammet for Soleklart sin arkitektur. Tilsvarende komponenter i hvert diagram er markert med like farger.

begrenset antall oppgaver som den enheten har ansvar for. En mer detaljert beskrivelse av arkitekturdesignprosessen er tilgjengelig i kapittel 6.5 Valg av arkitektur.

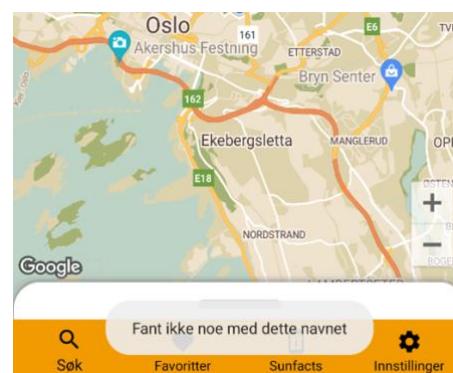
4.3 Kvalitetsegenskaper

4.3.1 Inndatavalidering og feilhåndtering

Siden brukerens input er et aspekt som ikke kan kontrolleres i særlig grad, er dette programmet designet med fokus på at eventuelle feil vil forårsake minst mulig konsekvenser.

Applikasjonen Soleklart har to funksjoner i grensesnittet hvor brukeren kan oppgi inndata. Det første stedet er stedssøkefeltet i kartet, hvor brukeren kan gi input i form av fritekst, og bruke det til å finne et sted i kartet. I dette tilfellet er "ugyldig input" definert som en input som ikke resulterer i vellykket oppslag av sted. Input valideres ikke før den blir sendt til Geocoder (Geocoding API, Android), men håndteres i etterkant dersom oppslaget ikke er vellykket på grunn av ugyldig input. Applikasjonen detekterer oppslag som ikke ga resultat på grunn av ugyldig input, og presenterer en *Toast*-melding. Det vil si en tilbakemelding som informerer brukeren om ugyldig input. Programmet tillater uendelig forsøk på input. Figur 8 viser en del av skjermbildet med denne *Toast*-feilmeldingen.

Den andre funksjonen er hvor brukeren velger en dato i kalenderen. Her er inputvalidering utført ved å begrense brukerens mulighet til å velge en dato som ikke er innenfor 7-dagersperioden frem i tid. Her vil det ikke skje inputvalidering,



Figur 8: Feilmelding som gis til bruker etter et mislykket søk på grunn av ukjent input.



men input er begrenset til godkjente datoer. Trykk på ugyldige datoer vil ikke føre til en hendelse. Ugyldige og gyldige datoer er markert ved ulik skriftstil (*Figur 9*).

Ved begge tilfellene vil ugyldig input ikke stoppe programmet, men la brukeren forsøke på nytt.

4.3.2 Pålitelighet

Applikasjonens kvalitetsegenskaper som pålitelighet og tilgjengelighet er i stor grad avhengig av kvaliteten til de eksterne tjenestene applikasjonen bruker. Tilgjengelighet av værvarsel brukeren får via Soleklart er direkte knyttet til tilgjengelighet og kvalitet på tjenestene som benyttes.

NILU har tilgjengelighet hver dag i hele året (NILU, 2022), noe som oppfyllet kravet om at luftkvalitetsdata skal være tilgjengelig året rundt. NILU har derimot ikke publisert data som viser tilgjengelighet av data gjennom døgnet. Dermed er det ikke mulig å anslå størrelse på sannsynligheten om at tjenesten vil være fraværende. Applikasjonen Soleklart stiller krav om at NILU luftkvalitetsdata som vises skal være hentet fra den nærmeste målestasjonen som befinner seg innen 20 kilometer radius fra det ønskede stedet. Dermed vil i noen tilfeller luftkvalitetsdata fra NILU være utilgjengelig, og i andre tilfeller gi irrelevant data, da distansen til målestasjonen er veldig stor. Det er ukjent for hvor stor dekning av landet vil denne dataen være utilgjengelig.

Sunrise API-tjenesten tilbyr heller ikke noen garantier for kvalitet, men beskriver seg selv som assosiert med høy kvalitet og pålitelighet og skriver at de kan prosessere rundt 50 000 forespørsel per time (Tveter, 2021). Locationforecast oppdateres en gang hver time for data fra nordiske regioner utenom arktisk område (Seierstad, 2020). Bortsett fra det, er det ikke mye informasjon om påliteligheten til tjenestene Locationforecast tilbyr. Meteorologisk institutt skriver at trafikken kan optimaliseres ved å sette maksimal trafikkgrense på 20 forespørsler per sekund (Aalberg, 2020). Dermed kan det antas at trafikk under denne grensen mest sannsynlig vil føre til en pålitelig kvalitet.

Geocoder (Geocoding API, Android) gir ikke garanti for tilgjengelighet og nøyaktighet (Android Developers, 2022). Det ble konkludert med at i skopet av dette prosjektet vurderes Geocoders kvalitet til å være god nok. Grunnlaget for dette er at applikasjonen Soleklart ikke er ment til å benyttes i kritiske funksjoner som har et formål om å ta vare på helse, forhindre skade på mennesker og miljø, eller opprettholde normale samfunnsfunksjoner på andre måter. Dermed vil Soleklart som en mulig feilkilde ikke føre til betydelig skade.

4.3.3 Vedlikeholdsvennlighet

Applikasjonens vedlikeholdsvennlighet oppnås gjennom begrenset kompleksitet, testbare enheter, dokumentasjon og tydelig arkitekturdesign.

Programmet er skrevet med et fokus på minimal kompleksitet, lav kobling og høy kohesjon, noe som oppnås ved at komponenter som har tydelige ansvarsområder og arkitekturen som følger det klassiske MVVM design pattern. Applikasjonens arkitektur er designet for å tillate



Figur 9: Input begrensning i kalenderen. Gyldige og ugyldige datoer er skrevet med ulike skrift.



utbytting av komponenter og skalering. Arkitekturen er beskrevet og dokumentert ved bruk av diagrammer (kapittel 4.1 *Grunnleggende informasjon* og 4.2 *Systemarkitektur og objektorienterte prinsipper*) som vil fungere som utgangspunkt for vedlikehold og videre utvikling. En mer detaljert beskrivelse av arkitekturen og designprosessen er tilgjengelig i kapittel 6.5 *Valg av arkitektur*.

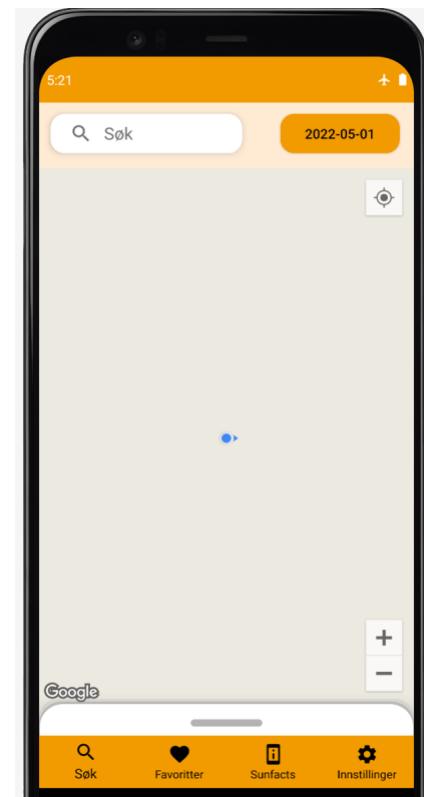
Tilgjengelighet og lesbarhet av koden oppnås ved bevisst bruk av mapper (engelsk: *package*) og tydelig mappehierarki. Applikasjonens mappestruktur separerer komponenter med like ansvarsområder og oppgaver i egne mapper. Det finnes tre øvrige mapper – *ds*, *ui* og *utils* som gir en grov inndeling. Mappene *ds* og *ui* inneholder flere mapper som gir en finere oppdeling av funksjoner.

Applikasjonens kode er dokumentert ved bruk av KDoc og følger standard KDoc syntaks med kommentarblokker som starter med `/**` og ender med `*/`. Første del av kommentaren er en oppsummering, mens det som kommer etter er detaljert beskrivelse. De fleste funksjoner har også tags (@). Tilgjengelighet av skriftlig dokumentasjon oppnås ved at koden, kommentarene og dokumentasjonen er skrevet på engelsk.

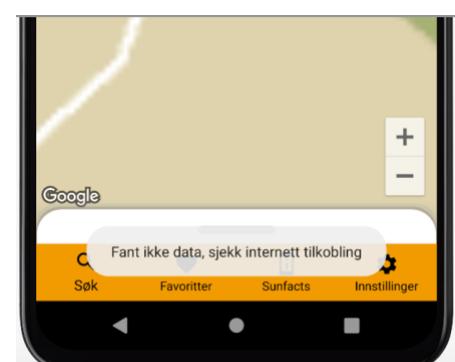
De kritiske og viktigste komponentene av systemet er testet ved bruk av enhets- og integrasjonstester. Testene er tilgjengelig i mappen *app/src/test*. Denne mappen vil kunne brukes for fremtidige tester.

4.3.4 Svartid og internetttilkobling

Applikasjonens tilgjengelighet og funksjonalitet er i stor grad avhengig av internetttilkobling. Hovedfunksjonen til Soleklart – henting av tidspunkt for soloppgang og solnedgang, og varsel for disse tidspunktene, er avhengig av maskinens tilkobling til internett. Kartfunksjonen er også avhengig av internett. Dersom tilkobling er fraværende fra oppstarten av applikasjonen, vil kartet ikke vises. Et allerede opplastet kart vil være tilgjengelig og synlig med lavere oppløsning dersom internett forsvant etter oppstart. Manglende tilkobling vil også gjøre det umulig å skifte kartstil, fordi kartet ved skift av stil vil lastes opp på nytt, noe som er avhengig av internett. Det har ikke blitt implementert funksjonalitet for å varsle brukeren om at kartet er utilgjengelig. Det utilgjengelige kartet vises i form av en grå bakgrunn, noe som er en fremstilling laget av Google Maps API. Figur 10 viser hvordan et utilgjengelig kart ved manglende nettverk vil se ut for brukeren. Manglende varsling i dette tilfellet ses på som en faktor som kan redusere både brukervennlighet og mulighet til fremtidig testing.



Figur 10: Applikasjonens GUI ved utilgjengelig kart (Google Maps API) som et resultat av fraværende nettverkstilkobling.



Figur 11: Toast melding som vises ved fraværende internetttilkobling.



Applikasjonene varsler om manglende internett-tilkobling som en respons på trykk i kartet eller endring av dato. Varslingen skjer ved at brukeren får en Toast-melding med tekstu "Fant ikke data, sjekk internett-tilkobling" (*Figur 11*). Dersom internett er fraværende ved stedssøk ved bruk av tekst, vil brukeren få den samme meldingen som ved tilfellene hvor ugyldig tekst ble sendt inn (se *Figur 8*, kapittel 4.3.1 *Inndatavalidering og feilhåndtering*). Det at programmet ikke skiller mellom fraværende internett og ugyldig input ved tekstsøk er et kjent aspekt innen teknisk gjeld i denne applikasjonen.

Fremstilling av allerede hentet data, GUI, onboarding samt Sunfacts er ikke avhengig av internett-tilkobling og kan fungere som normalt uten at maskinen er tilkoblet. Solkartet ligger på en nettside, og er dermed også avhengig av nettilkobling.

Ved normal internett-tilkobling vil programmet forsøke å hente inn data i 55 sekunder. Mislykket henting vil resultere i den samme Toast-meldingen som vist i *Figur 10*.

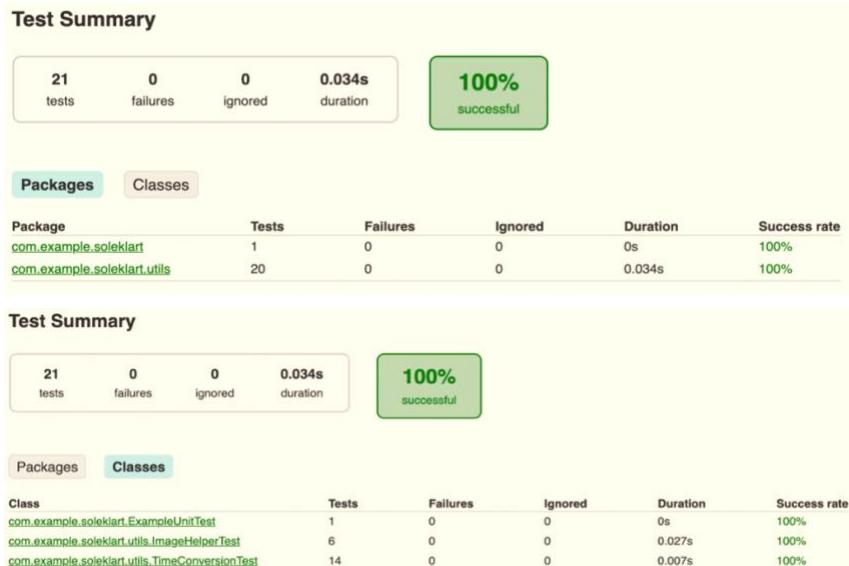
5 Testdokumentasjon

5.1 Testmetoder

Underveis i prosjektet har gruppen hovedsakelig kjørt manuelle tester for å teste funksjonalitet i appen. Det er også laget enhetstester, spesielt for å teste enkelte *utility*-metoder for tidskonvertering og henting av *drawables* basert på data (*Figur 12*). Integrasjonstester er også blitt laget for å teste kall på ulike API-er (*Figur 13*), og for å sjekke at data blir oppdatert etter nye koordinater er satt. Gruppen har også hatt som mål det skal kjøres manuelle tester før kode pushes til hoved *branch*, for å sikre at denne forblir så feilfri som mulig. Hovedflyten i appen har vært hovedfokuset ved denne testingen. I tillegg til testing av at data vises som forventet etter søker på sted og valg av kart har vært sentralt ved manuell testing.

5.2 Formål

Formålet med testingen har vært å kontrollere forventet oppførsel fra appen og dets metoder. Testing har i visse tilfeller også blitt brukt underveis i utviklingen av noen metoder, for å få bedre oversikt over program-flyten. Under utviklingen av metoder for konvertering av tidsstempeler og datoer, har testing vært spesielt nyttig for å sjekke at metoder gir forventet resultat. Disse metodene er godt egnet for å testes isolert fra andre deler av appen, da det ikke er noen



Figur 12: resultat av enhetstester.



avhengigheter å måtte ta høyde for. Testingen har hjulpet med å finne feil som ellers kunne vært vanskelig å feilsøke, slik som feil med å få hentet neste dag i kalenderen ved månedsskifte.

Denne type testing kan dermed ha spart tid for gruppen senere i utviklingsprosessen, da det har gjort det enklere å finne slike feil tidlig, for så å løse dem.

Status		5 passed	5 tests, 13s 931ms
Filter tests:			
Tests		Duration	Pixel_2_API_29_Oblig
✓ Test Results		317 ms	5/5
✓ ExampleInstrumentedTest		18 ms	1/1
✓ useApplicationContext		18 ms	✓
✓ LocationforecastDSTest		1 ms	1/1
✓ getForecast_isNotNull		1 ms	✓
✓ NiluDataSourceTest		196 ms	1/1
✓ fetchNilu_isNotNull		196 ms	✓
✓ SunriseDataSourceTest		102 ms	1/1
✓ getCompactSunriseData_isNotNull		102 ms	✓
✓ BottomSheetFragmentTest		0 ms	1/1
✓ outputDataNotEmptyOnLatLongChange		0 ms	✓

Figur 13: resultat av integrasjonstester.

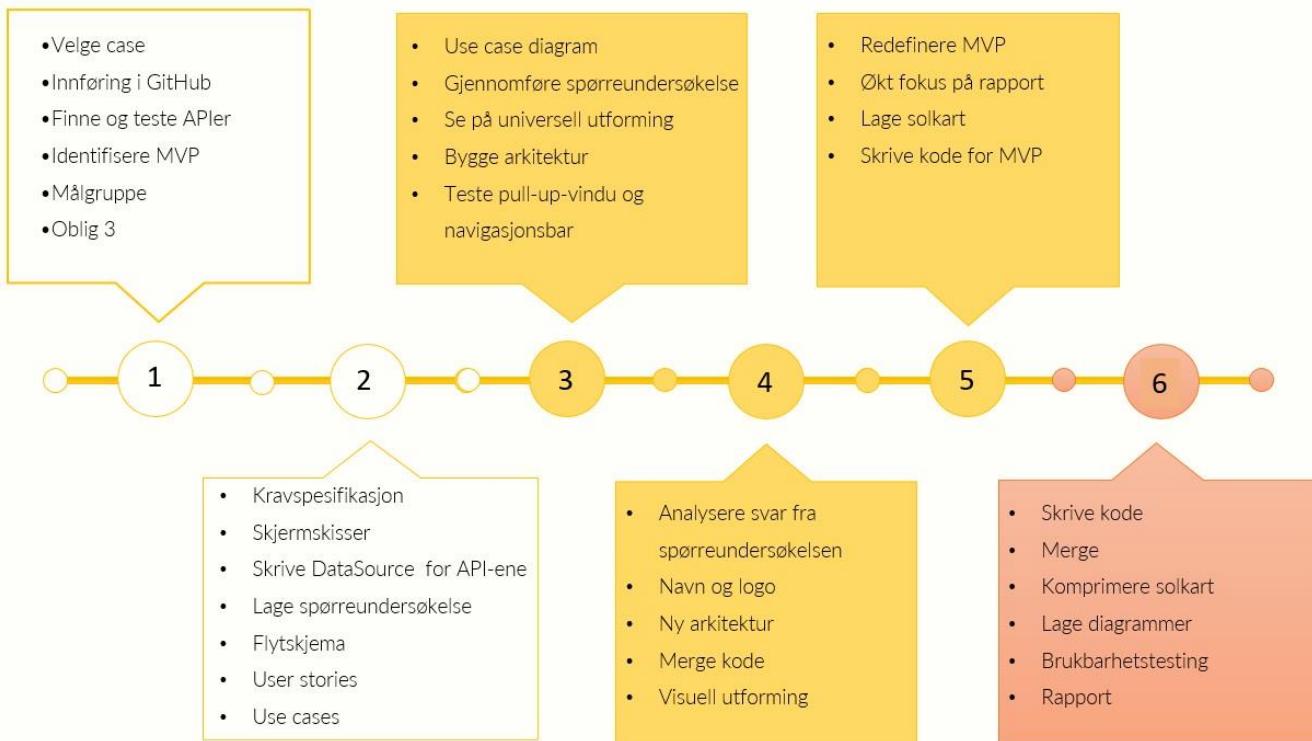
5.3 Verktøy

Verktøy benyttet til testing har vært Android studio og JUnit. JUnit er blitt brukt til å lage og kjøre enhetstester og integrasjonstester. Emulator i Android studio er blitt brukt for manuell testing, både for funksjonalitet og testing av utseende på forskjellige skjermstørrelse og oppløsning. Verktøy fra emulatoren, som blant annet testing med forskjellige internetsignal og ulike GPS koordinater har også blitt benyttet under testing av applikasjonen. Proto.io har gruppen brukt under tidlig brukbarhetstesting av designelementer i appen.



6 Prosessdokumentasjon

I dette kapittelet presenteres mange aspekter ved utviklingsprosessen. Prosessen beskrives ikke kronologisk, men er oppdelt i temaer som arbeidsmetode, verktøy, produktvisjon, MVP, arkitektur, utforming og undersøkelser. Det er delt opp på denne måten fordi gruppen mente det ville være mer oversiktlig å finne all informasjon knyttet til ett tema samlet, i stedet for at alle temaene presenteres samlet. *Figur 14* viser en oversikt over tidsboksene gruppen har jobbet i.



Figur 14: En oversikt over tidsboksene gruppen har jobbet i. Ulike farger for innledende fase, hovedfase og avsluttende fase.

6.1 Smidige utviklingsmetoder

I denne delen forklarer det hvorfor gruppen valgte Scrumban som smidig utviklingsprosess under prosjektet, hvordan det ble praktisert i en tidlig fase, og hvordan det utviklet seg gjennom prosjektet. Merk at med Scrumban, menes gruppens tilnærming til denne utviklingsmetoden. Dette beskrives i de kommende avsnittene.

6.1.1 Valg av smidige metoder

Gruppen diskuterte hvilken smidig metode som skulle bli praktisert i prosjektet. Prinsippene ved plandrevne utviklingsmetoder ble ansett som uegnet, mens Scrum og Kanban ble oppfattet som mer aktuelle.

Gruppen valgte Scrumban som utviklingsmetode for hele prosjektet. Årsaken skyldes av fordelene som gruppen kan få nytte av gjennom Scrum og Kanban. I Scrum, har man “sprinter” hvor man har som mål å utvikle noe som kan lanseres, og legges til som et



inkrement til den endelige løsningen (Sjøberg, 2021). Det var enighet i gruppen om å jobbe i ukesintervaller, og kalle disse sprinter, til tross for at de ikke nødvendigvis resulterte i et inkrement som kunne lanseres. Ukentlige sprinter, med nye vurderinger av hva som måtte gjøres, og evaluering av hvordan det hadde gått, ble oppfattet som passe hyppig for dette prosjektet, ved at det ville gi gruppen mulighet til å gjøre erfaringer og endre arbeidsmetoder før det var gått for lang tid.

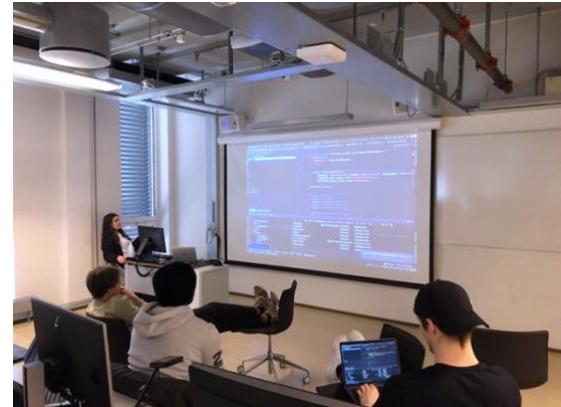
I Kanban har man fordeler som ikke kan oppnås gjennom Scrum. Blant annet prioriteres såkalte “kritiske problemer” i Kanban hvor det kan oppstå et problem som fører til at teamet må prioritere å få løst dette problemet. Årsaken er at slike problemer kan hindre videre utvikling og kan koste dyrt hvis man ikke løser dem raskt når de oppdages. En annen grunn til å inkludere Kanban, skyldes økt fleksibilitet. Det kan oppstå tilfeller under sprinten hvor gruppen kommer til enighet om at det er flere ting som kan bli gjort under sprinten, men som ikke er i backloggen. Kanban tillater at man kan jobbe med andre viktige oppgaver om man nylig ble ferdig med en oppgave (Stray, 2022). Dette syntes gruppen kunne være nyttig for å være fleksible og være i forkant i forbindelse med utviklingen av appen. Slik er gruppens tilnærming til Scrumkanban hvor man har blandet de ulike prinsippene fra både Scrum og Kanban som er best egnet for gruppen.

6.1.2 Innledende fase

Den innledende fasen besto av de to første sprintene i prosjektet. Prinsippet om “kontinuerlig prosessforbedring”, som handler om at gruppen skal reflektere over hvordan ting kan forbedres og justere aferden etter dette (Stray, 2022), ble viktig i denne fasen.

Ettersom dette var startfasen i prosjektarbeidet, og de fleste i gruppen ikke kjente hverandre så godt, var det felles enighet om at møtene hovedsakelig skulle være fysisk. Årsaken skyldes at ved smidig utvikling er “ansikt-til-ansikt”-kommunikasjon mest effektiv for å formidle informasjon til og innad i et utviklingsteam (Stray, 2022). Almås (2021, s.7) skriver også at flere intervjuobjekter, studenter i emnet *IN2000*, fortalte at da de ble nødt til å jobbe digitalt da pandemien kom, førte det til at kommunikasjonen ble merkbart dårligere, og dette gikk igjen ut over både samholdet og samarbeidet. Ved dårlig kommunikasjon ble terskelen for å ta kontakt med de andre, eller spørre om hjelp, høyere. Gruppen mente at fysisk samarbeid ville bidra til å bli bedre kjent med hverandre, styrke mellommenneskelig kjemien, kommunikasjonen og samarbeidet. Figur 15 er et bilde fra et av møtene.

Alle kjente noen av de andre fra før, noe som kan være både en fordel og en ulempe. Det vil si at enkelte allerede hadde en gjensidig tillit, mens det kan være mer krevende å etablere nye bånd på tvers når det allerede er en slags skjevhetsrelasjonene. Det ble utarbeidet en samarbeidsavtale, og gjennomført en forventningsavklaring med tanke på hvordan man ønsket at gruppen skulle fungere.



Figur 15: Gruppens første sprint-review møte.



Til retrospektiv-gjennomgangen i den andre sprinten, hadde et gruppemedlem laget et spørreskjema hvor alle kunne svare anonymt. Det kom fram at det var litt ulikt syn på flere aspekter ved arbeidsmetodene. Resultatet fra undersøkelsen er tilgjengelig i vedlegg 9.3 *Rapport fra Sprint Review 2 nettskjema*. Blant annet var “lengden på møtene” og “strukturen på møtene” de viktigste punktene som ble diskutert. Selv om noen var fornøyde, var alle enige i at møtene kunne bli litt for lange. Årsaken var at det manglet struktur i møtene. Frem til nå hadde gruppen planlagt møter uten en agenda, noe som førte til at man brukte tid på å lage en slik agenda og å utføre den i samme møte. Det ble bestemt at agendaen skulle fastsettes på forhånd for å gjøre møtene mer effektive og strukturerete.

Det kom også noen tanker om at måten teamet hadde jobbet på ble for kaotisk. Det ble klart at ingen var visste om noen hadde bestilt rom til møter, hva gruppen skulle gjøre på de neste møtene, hvilke saker som var aktuelle og ikke minst om noe av det gruppen gjorde ble dokumentert. Alle disse oppgavene ble utført, men med mye synsing og usikkerhet. Det ble innført en fasilitator-rolle som skulle ha ansvar for å ordne det praktiske med rombestilling, skrive innkalling til møtet på kommunikasjonskanalen, opprette saksliste, skrive en kort oppsummering av møtet, og ikke minst - å hjelpe til med å holde struktur i møtene. Denne rollen skulle gå på omgang, så man unngikk at noen skulle oppfattes som en leder, og fordi sykdom og andre forpliktelser gjorde at enkelte i perioder ikke var tilgjengelige.

I begge sprintene innså gruppen hvorfor det smidige prinsippet “kontinuerlig prosessforbedring” er utrolig viktig i prosjektutvikling. Muligheten for at problemer kan oppstå under prosjektutviklingen vil alltid være der, men muligheten til å endre og tilpasse prosessen er en av grunnene til at smidig utvikling er så utbredt. Gjennom retrospektive-møtene ble det mulig å vurdere arbeidsmetodene, gjøre vurderinger og endringer. Uten refleksjon rundt prosessen, kunne strukturen ha fortsatt å være et problem for både fremgang, arbeidsmoral og samhold.

6.1.3 Mellomfasen

Mellomfasen omfatter sprintene 3 – 5, som kan sees i vedlegg 9.4 *Prosjektkalender*.

I den tredje sprinten, møtte gruppen på et “kritisk problem” som kunne hindre videreutvikling av appen. I sprint-planningen ble det valgt arbeidsoppgaver til backloggen, men gruppen skulle ha et veiledingsmøte med veilederne etter planleggingsmøtet. Dette møtet sørget for at gruppen måtte ha et nytt planleggingsmøte senere i uken, da det dukket opp et kritisk problem. Under møtet hadde veilederne spørsmål angående app-arkitekturen, hvor gruppen ikke hadde et presist svar på hvordan arkitekturen ville være. Dette blir utdypet i kapittel 6.5 - *Valg av Arkitektur*. Det førte til at hovedfokuset ved sprinten måtte bli å få løst dette problemet ettersom arkitekturen er essensiell for at appen skal fungere som planlagt. *Figur 16*



viser et bilde fra dette møtet. Dermed ble det holdt et nytt planleggingsmøte for å sette fokus på dette problemet, og håndteringen gikk bra.

Under sprint-review hadde man en løsning av app-arkitekturen. Gruppen var fornøyde over hvordan man håndterte det kritiske problemet, og hadde så erfaring til eventuelle fremtidige kritiske problemer. I retrospektiv diskuterte gruppen om å eventuelt vurdere å ha flere veileddingssamtaler siden det var gjennom veiledningsmøtet at problemet ble oppdaget. Eventuelt skulle gruppen reflektere enda grundigere etter hvert sprint-review for å avdekke eventuelle kritiske problemer. Her kan man se et eksempel på hvor nyttig prinsippet fra Kanban kan være. Gruppen var i stand til å håndtere sitt første kritiske problem og hvordan man skulle løse dette. Det kunne gått dårlig i senere utvikling hvis arkitekturen ikke var bestemt ettersom det er utrolig viktig å få på plass. Siden gruppen hadde planleggingsmøte og diskuterte hvordan dette problemet skulle løses, gikk dette ganske bra.

I den påfølgende sprinten hadde gruppen hybride møter, ettersom to medlemmer ble syke. I en slik situasjon ble løsningen å holde møter fysisk, men også bruke digitale verktøy slik at de syke medlemmene blir inkludert. Dermed ble det ikke mye ansikt-til-ansikt kommunikasjon mellom alle i gruppen, men det førte ikke til komplikasjoner ettersom informasjonen når det gjaldt arbeidsoppgaver til sprinten ble godt formidlet til alle.

I den siste sprinten, ble det smidige prinsippet om å være “positiv til endringer” benyttet, ettersom kravspesifikasjonen måtte endres. Da gruppen skulle vurdere hvilke funksjonelle krav som skulle settes i sprint-backloggen, førte dette til at man måtte endre en del av kravspesifikasjonen. Gruppen måtte endre kravspesifikasjonen da det ble tydelig at tidsestimatet for å innfri alle de ønskede kravene ikke stemte overens med tiden til rådighet. Dette står det mer om i kapittel 6.4 *MVP og Kravspesifikasjon*. Men når man jobber smidig, er det greit å “ønske endringer i krav velkommen, selv sent i utviklingen” (Stray, 2022). Dette er en av fordelene ved å jobbe smidig, og noe som var nyttig for gruppen å erfare ettersom mye av arbeidspresset i fremtidige sprinter ble lettere.

Gruppen var enig i at den femte sprinten skulle være i to uker, ettersom påskeferien nærmet seg. Det var noen i gruppen som ville jobbe i påskeferien, mens andre tok ferie og reiste ut av landet. Dermed konkluderte gruppen om å ha en lang sprint med mindre arbeid, mot at man var klar for å ha flere hyppige møter i sluttfasen for å ferdigstille prosjektet.



Figur 16: Et øyeblikk fra møtet hvor gruppen samarbeidet for å løse det kritiske problemet.

Her kan man se et eksempel på hvor nyttig prinsippet fra Kanban kan være. Gruppen var i stand til å håndtere sitt første kritiske problem og hvordan man skulle løse dette. Det kunne gått dårlig i senere utvikling hvis arkitekturen ikke var bestemt ettersom det er utrolig viktig å få på plass. Siden gruppen hadde planleggingsmøte og diskuterte hvordan dette problemet skulle løses, gikk dette ganske bra.



6.1.4 Avsluttende fase

Da påskken var over, gikk prosjektet inn i en avsluttende fase. Gruppen vurderte det til at det ikke lenger var praktisk å jobbe i ukesintervaller på samme måte, da det var kort tid igjen av prosjektet, og flere gruppemedlemmer hadde en hjemmeeksamen å ta hensyn til. Kanban-metodikken ble dermed veldig tydelig, med en liste av oppgaver som måtte fullføres innen prosjektets slutt. Møtene ble dessuten hyppigere, slik at man i større grad enn før jobbet sammen på oppgaver, og kommunikasjonen gikk raskere, for eksempel ved diskusjoner og tilbakemeldinger på arbeid. Flere av diagrammene ble laget i fellesskap, for å sørge for at hele gruppen hadde forståelse for det tekniske, og som en repetisjon av diagrammer før eksamen. *Figur 17* viser et bilde tatt under en av møtene i den avsluttende fasen.



Figur 17: Ett av de mange møtene som ble holdt i den avsluttende fasen.

Gjennom hele prosjektet hadde alle jobbet i ulike *branches* som ble testet og godkjent før den ble slått sammen med hovedkoden. Dette hadde sørget for at man unngikk *spaghettikode* og sammenslåingskonflikter, og tiden kunne i stedet brukes på å fullføre MVP, samt gjøre små endringer, parallelt med rapportskriving. Gruppemedlemmene har ulike bakgrunner og erfaringer, og det var derfor litt ulike syn på hvordan en prosjektrapport skulle bygges opp, og hvordan de ulike kapitlene skulle være. Dette ble diskutert med både veiledere, sjefsgruppelærere og faglærer, men alle ga ulike svar. Det var derfor vanskelig å vite om rekkefølgen i rapporten hadde en god og forventet flyt, eller om enkelte av retterne ville mene at underkapitlene i kapittel 6 skulle stokkes om på. Rapportskrivingen hadde pågått gjennom hele prosjektet, noe som gjorde at det var gode notater og utkast å ta utgangspunkt i til den avsluttende rapporten.

6.2 Verktøy

Her følger en beskrivelse av verktøyene gruppen valgte å bruke. *Tabell 6* presenterer en kort oversikt over verktøyenes bruksområder i dette prosjektet. De fleste verktøyene har konkurrerende verktøy med tilsvarende egenskaper. For eksempel er Figma et mer utbredt verktøy for skjermkisser, men gruppen valgte likevel å bruke Proto.io. Dette er fordi ingen på gruppen hadde erfaring med Figma, og gruppen da foretrakket å jobbe med et kjent program. Dette ville kunne spare gruppen for tid og frustrasjoner ved å skulle sette seg inn i for mye nytt på en gang.

**Tabell 6:** oversikt over verktøy og deres bruksområder i dette prosjektet.

	D i s c o r d	M e s s e n g	T e a m s e r	G i t H u b	A n d o i d s t u d i o	D r a w i d s t u d i o	M i r o t o i o	P r o t o .	N e t s k j e m a	T r e l l o	G o o g l e m a	M i o c r e s D r i v e	Q G I S	
Kommunikasjon	X	X	X											
Planlegging	X		X	X				X			X	X		
Koding						X								
Versjonskontroll				X										
Design							X							
Modellering								X						
Prototyping									X					
Brukbarhetstesting					X				X					
Brukerundersøkelser										X				
Dataanalyse												X		
Rapportskriving											X	X		
Geovisualisering				X									X	

6.2.1 Planlegging, dokumentasjon og kode

6.2.1.1 Trello

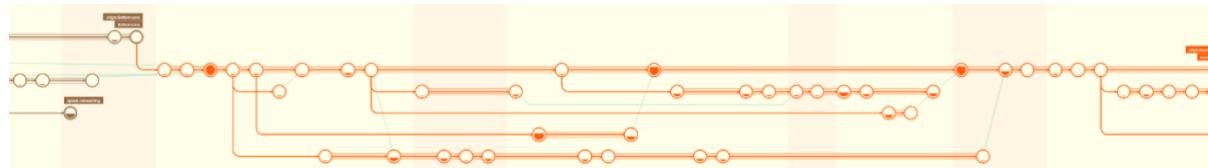
I Trello hadde gruppen en Kanban-tavle med en oversikt over prosjekt-backlog, sprint-backlog, hvilke oppgaver man jobbet med, og hva man hadde jobbet med i tidligere sprinter. Dette ga en visuell oversikt over prosessen, og ble brukt mye i starten av prosjektet. Etter hvert, føltes tavlen noe overflødig ut, da kode-oppgavene var på en egen Kanban-tavle i GitHub, og det ikke var så mange oppgaver å holde oversikt over i Trello.

6.2.1.2 Android Studio

Det var en selvfølge å bruke Android Studio til å skrive kode, i og med at det er det programmet alle har kjennskap til gjennom de obligatoriske oppgavene i faget. Her ble også programmet kjørt i en emulator.

6.2.1.3 GitHub

Når det skal utvikles et større program, eller mange personer skal jobbe på samme kode, er det viktig med god versjonskontroll. Dette sørget gruppen for gjennom å bruke GitHub. Et felles prosjekt ble opprettet, og grener ble opprettet for hver seksjon av kode (*Figur 18*). Disse kunne da trygt jobbes i, uten risiko for å ødelegge fungerende kode. Ny kode kunne så testes før den flettes med felleskoden. I GitHub førte gruppen også en Kanban-tavle med koderelaterte oppgaver, med en oversikt over både prosjekt-backlog og sprint-backlog.



Figur 18: Visualisering av noen av grenene i utviklingen.

GitHub ble også benyttet for å publisere solkartet. Muligheten for å publisere en nettside på GitHub var en til da ukjent mulighet for flere i gruppen. Det å kunne gjøre det på denne måten føltes veldig praktisk, i og med at teamet da slapp å betale for et domene.

6.2.1.4 Google Drive

Alle filer ble delt på en felles Google Drive. Hovedhensikten med å bruke skyløsning for dokumentoppbevaring, var å sikre at alle medlemmer i teamet hadde lett tilgang til nyeste versjon av dokumenter. I tillegg til det, tilbyr Google Drive back-up, som var en ønsket sikkerhetsfunksjon. Teamet hadde allerede erfaring med Google Drive, og tilgang gjennom universitetet, og det var dermed et naturlig valg.

6.2.1.5 Microsoft Office

Rapporten ble først påbegynt i en Google Docs. Etter hvert oppsto det frustrasjon over at dokumentet manglet funksjonalitet som finnes i Microsoft Word, som automatisk nummerering av kapitler, tabeller og figurer. Dermed ble rapporten flyttet over til et Microsoft Word-dokument. Dokumentet lå delt på nettet, slik at alle til enhver tid hadde en oppdatert versjon. Dessverre var heller ikke Microsoft Word problemfritt, da bildene ofte forsvant (*Figur 19*). Dette var et gjentagende problem under rapportskrivingen, og det ble enda viktigere at bildene var lagret på Google Drive, så de lett kunne limes inn på nytt.

 Bildet kan ikke vises.

Figur 19: Eksempel på forsvunnet bilde.

6.2.2 Kommunikasjon

6.2.2.1 Discord

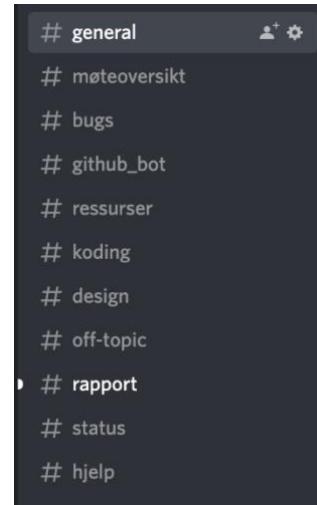
Dette var gruppens hovedkanal for kommunikasjon og koordinering av arbeidsoppgaver. *Figur 20* viser en oversikt over gruppens kanaler på Discord. Man kan opprette en egen server for kun inviterte, og inne i den opprette tekst-kanaler og lyd/video-rom. Ved å ha flere



kanaler med ulike temaer, kunne gruppen ha flere parallelle diskusjoner uten at det ble uoversiktlig.

Lyd/video-rommet var alltid tilgjengelig, og man kunne se hvem som er der, uten å måtte åpne det. Det var ikke behov for en vert, og hvem som helst kunne gå inn i eller forlate rommet når de selv ville. Alle kunne å dele skjerm, også samtidig. Dette var en fordel i de digitale og hybride møtene, da det ga alle mulighet til å selv velge hvilken skjermvisning man ønsket å følge til enhver tid. Eventuelle kommentarer i chatten forsvant heller ikke da videokonferansen ble avsluttet, men ble laggende i serveren.

Teamet hadde også installert en egen bot som varslet hver gang det skjedde en endring på GitHub, noe som gjorde det enkelt å holde seg oppdatert på kode-progresjonen.



Figur 20: Skjermbilde med alle gruppens kanaler på Discord.

6.2.2.2 Messenger

Messenger ble brukt for viktig kommunikasjon som trengte raske svar. Fordelen her var at man kunne se om meldingen var blitt lest, eller om man måtte ringe for å være sikker på at alle fikk med seg beskjeden.

6.2.2.3 Teams

Kommunikasjonen med veilederne foregikk på Teams, da dette er den offisielle kommunikasjonskanalen i IN2000. Utover det, har teamet ikke brukt Teams, til tross for en del overlappende funksjonalitet med Discord.

6.2.3 Modellering og design

6.2.3.1 Draw.io

Draw.io ble brukt til modellering av diagrammer, som use case-, sekvens- og klassediagrammet. Draw.io ble benyttet både til endelige diagrammer fremstilt i denne rapporten og som et verktøy i plenumsdiskusjoner.

6.2.3.2 Miro

Brainstorming og visuell formidling skjedde i Miro, et digitalt tavle med Post-it-lapper, der man også kan lime inn bilder og figurer. Dette ble brukt både til idémyldring rundt navn på appen, samt å samle alle skjermkisser, visuelle elementer og fargekoder på ett sted. Miro har også mange andre funksjoner, og da gruppen skulle avgjøre appnavnet, foregikk dette i en anonym avstemming i Miro, der resultatene genereres automatisk. Dette var et nytt verktøy for Prosa-studentene, og et verktøy de kunne tenke seg å bruke igjen. *Figur 23* i kapittel 6.3.4 *Valg av navn* er et eksempel på bruk av Miro-tavle.

6.2.3.3 Proto.io

Skjermkisser ble laget i Proto.io. Prosjektet lå da på nett, slik at alle hadde tilgang til det samme prosjektet, selv om det kun var Design-studentene som brukte det. Ved å koble



elementene fikk gruppen en klikkbar prototype som kunne testes internt uten å måtte skrive kode. Dette hjalp teamet med å gjøre raske endringer i oppsettet, og lettere forklare hverandre hvordan man så for deg interaksjonen.

6.2.4 Datainnsamling og analyse

6.2.4.1 UiO Nettskjema

UiO sitt Nettskjema ble brukt for innhenting av anonymisert kvantitativ data i den første spørreundersøkelsen, samt i en sprintreview. Løsningen fungerte svært bra for den kvantitative undersøkelsens formål, da den sørger for automatisk anonymisering av respondentene, genererer en automatisk sluttrapport av undersøkelsenes resultater, samt sørget for at de innhentete dataene blir oppbevart og håndtert på en forskningsmessig forsvarlig måte, i motsetning til mange konkurrerende løsninger.

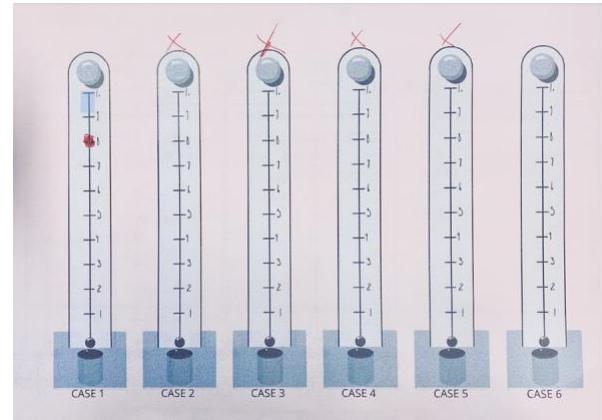
6.2.4.2 Geovisualisering

QGIS ble brukt for å lage solkartet. Dette er et *open source*-program for geovisualisering og romlige analyser. UiO-studenter har tilgang til ArcGIS gjennom UiO, men studenten med GIS-erfaring hadde kun jobbet i QGIS, og valgte derfor dette.

6.3 Produktvisjon og valg av case

6.3.1 Valg av case

Gruppens generelle mentalitet gjennom hele prosjektet, har vært et fokus på effektivitet. Det var ønskelig å sette i gang utviklingen med en gang, noe som avhenger av at man har valgt en case. Med inspirasjon fra veilederne, gjennomførte gruppen termometerøvelsen på det første møtet, for å måle hvor aktuell hver case er. *Figur 21* viser det samlede resultatet fra denne øvelsen. Gruppen startet med å eliminere uaktuelle caser. Til slutt gjensto case 1, og to selvvalgte caser: himmelsikt og strikking av temperaturpledd. Etter flere diskusjoner falt valget på himmelsikt.



Figur 21: Resultat fra en termometer-øvelsen hvor det måles engasjement for hver eneste case.



6.3.2 Utvikling av visjon

Utvikling og arbeidet med visjonen startet ikke med en undersøkelse av markedet, men heller kunnskap og interesse for domenet samt et eget personlig behov for en slik produkt.

Den første versjonen av visjonen for Soleklart baserte seg på å gi brukeren mulighet til å oppleve himmelfenomener under perfekte værforhold. Denne visjonen er ganske abstrakt og ble skrevet på en følgende uformell måte:

En app som viser bruker siktforhold mot himmelen, slik at man kan se vakre solnedganger eller stjernehimmelen. Appen skal la bruker søke opp en lokasjon i et kart, og skal så gi bruker informasjon om siktforhold (skydekke, vær, luftforurensning, soloppgang og solnedgang) i den ønskede lokasjonen.

Gjennom den tidlige fasen av arbeidet har visjonene fått to tydelige vinklinger. Den ene vinklingen har fokus kun på soloppganger og solnedganger og værforhold på disse tidspunktene. Den andre vinklingen har i tillegg synlighet av stjernehimmel og andre himmelfenomener som viktige aspekter av tjenesten. *Tabell 7* presenterer et kort sammendrag av disse vinklingene.

Tabell 7: Oversikt over to mulige vinklingen av visjonen.

Vinkling 1	Vinkling 2
Soloppgang, solnedgang Fokus kun på værforholdene under soloppgang, solnedgang og noen timer etter solnedgang.	Soloppgang, solnedgang, stjernehimmel Fokus på både soloppgang, solnedgang og stjernehimmel. Stjernehimmel sees på som et verdifullt tillegg for et større spekter av opplevelser.
Viser klokkeslett for soloppgang, solnedgang, værforholdene og luftkvalitet under disse tidspunkt.	Viser det samme som vinkling 1, men i tillegg viser informasjon om eventuelle himmelfenomener som meteoroider, nordlys og kart over dagens stjernehimmel.

Visjonen ble justert og endret seg litt gjennom review-møtene, kombinert med en bedre forståelse for kravene. Diskusjoner i plenum førte til en enighet om å starte med en vinkling 1 som utgangspunkt for MVP (*minimal viable product*) fordi dette er en visjon som er mer begrenset, krever mindre tid til å gjøre om til MVP og fremdeles tillater addisjon av andre aspekter senere i prosjektet. En mer formell definisjon av visjonen ble senere laget ved bruk av en modell beskrevet av Sommerville (2015, s.8).

TIL en privatperson SOM ønsker å oppleve vakre solnedganger og soloppganger under trivelige forhold. SOLEKLART er en APP til smarttelefon SOM tilbyr brukerne en mulighet til å få oppdatert informasjon om vær- og siktforhold i forkant av opplevelsen, slik at de kan skape optimale minner av solnedgang/soloppgang. I MOTSETNING til andre applikasjoner tilbyr VÅR løsning en enkel og brukervennlig presentasjon av akkurat det som trenges for en perfekt opplevelse.

Det at det ikke fantes en kunde eller en produkteier, kompliserte prosessen knyttet til definering og spesifisering av visjonen. Diskusjonen var basert på egne synspunktet og behov



fremfor et klart definert behov hos kunden, noe som gjorde at hvert aspekt måtte drøftes med hensyn til flere spørsmål. Disse spørsmålene var blant annet:

Hvor sannsynlig er det at det finnes behov for dette hos andre?

Har vi tid og ressurser til å gjennomføre en valgt visjon?

Vil en visjon fremdeles være aktuell dersom vi klarer ikke å fullføre alt?

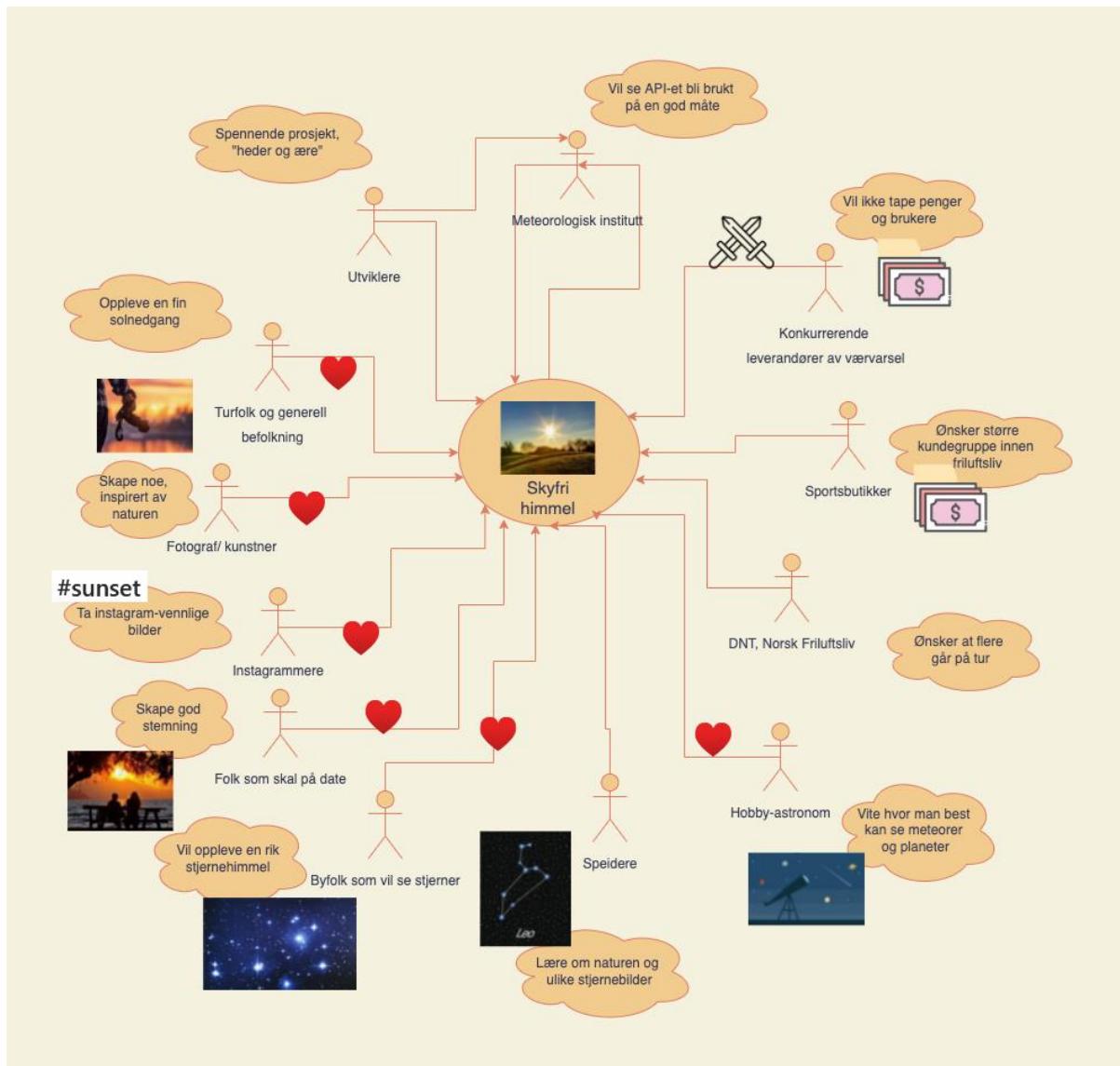
6.3.3 Interessenter og målgruppe

Arbeidet med å definere en målgruppe startet med å kartlegge interessenter i et rikt bilde (*Figur 22*). Denne informasjonen ble også samlet i en tabell (*Tabell 8*).

I dette prosjektet definerte man en generell målgruppe som var “myndige innbyggere i Norge med en gjennomsnittlig teknisk kompetanse, og en interesse for å oppleve soloppganger/solnedganger”. Denne generelle målgruppen kunne igjen deles inn i to undergrupper, nemlig én gruppe som ble estimert til å være *gjennomsnittlig* interessert i natur og friluftsliv, og én gruppe som ble estimert til å være *over gjennomsnittlig* interessert i natur og friluftsliv. Det var denne inndelingen av målgruppen som ble lagt til grunn for videre kvalitative og kvantitative undersøkelser knyttet til brukernes behov, krav og preferanser for prosjektet.

Tabell 8: En tekstlig oversikt over interessenter og deres

	Interessenter	Interesse
Individ	En som skal på date	Ønsker godt vær og god stemning
	Fotograf/ kunstner	Skape noe, inspirert av naturen
	Instagram-fotografer	Ta Instagram-vennlige bilder
	Byboer	Vil oppleve en rikere stjernehimmel enn det man kan se inne i byen
	Hobbyastronom	Vil vite hvor og når man kan se spesielle himmelfenomener
	Turfolk	Ønsker fine turopplevelser
Grupper	Speidere	Lære om stjerne-navigasjon og stjernebilder
	Utviklere (teamet)	Utvikle noe nyttig, som også kan brukes i jobbintervjuer
Organisasjoner	DNT, Norsk Friluftsliv	Ønsker flere aktive brukere innen friluftsliv i Norge
	Sportsbutikker	Vil tjene penger på større kundegrupper.
	Konkurrerende leverandører av værvarsler	Vil ikke tape penger og brukere
	Meteorologisk institutt	Vil se API-et bli brukt på en god måte



Figur 22: Et rikt bilde som viser interessenter og deres interesser

6.3.4 Valg av navn

Gruppen hadde 56 forslag til applikasjonens navn. Oversikt over alle forslagene vises i Figur 23. Det var naturlig å sortere forslagene etter kategori basert på ord opprinnelsen. Navnet ble valgt ved demokratisk avstemning hvor Soleklart var en soleklar vinner.



Figur 23: Oversikt over forslag til applikasjonens navn sortert etter kategori.



6.4 MVP og kravspesifikasjon

6.4.1 MVP – *minimum viable product*

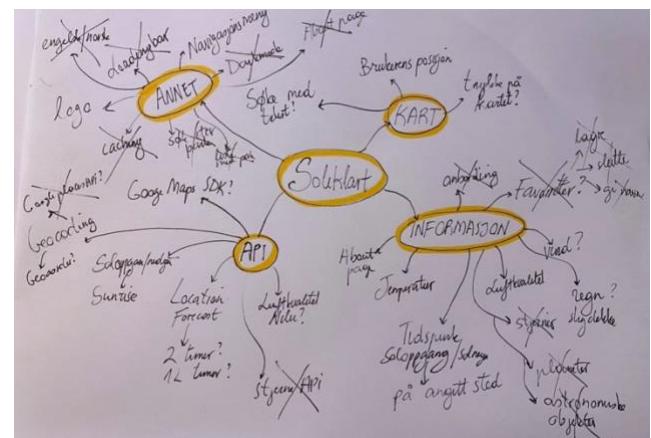
En av de viktigste og mest kritiske prosessene i en innledende fase, er å definere *minimal viable product* (MVP). Etter diskusjon med veiledere og internt i gruppen, var det enighet om at en tydelig definert MVP vil ha flere fordeler. Det vil føre til felles enighet i gruppen om hva som er målet med prosjektet, og en gjennomtenkt definisjon vil sikre at det som utvikles faktisk er noe som er "*viable*". Arbeidet for å definere en MVP gikk ut på å identifisere den mest essensielle delen av produktet som ville dekke behovet beskrevet i visjonen, og samtidig fungere som et fullverdig produkt.

Prosessene rundt definisjon av MVP ble utført i to faser – den første fasen var en uformell diskusjon, mens den andre fasen inneholdt en formell definisjon av funksjonelle krav (prosessen beskrevet i kapittel 6.4.2 *Kravspesifikasjon*). Figur 24 viser et øyeblikk fra denne diskusjonen.

Diskusjonen i den første fasen opplevdes som en naturlig videreføring av utviklingen av visjonen, men på dette tidspunktet ble fokuset konkretisert og tok utgangspunkt i spørsmålene:

1. Hva er den viktigste funksjonaliteten i dette programmet?
 2. Hvilke funksjonaliteter må programmet ha for at brukeren skal få dekket sitt behov?
 3. Hvilke funksjonaliteter er ikke nødvendige for å dekke dette behovet?
 4. Hvilke teknologier og API-er må implementeres og brukes for å lage en verdifull løsning?

Aspektene gruppen identifiserte som de viktigste, ble dokumentert i et tankekart (*Figur 25*). Figuren viser at det ble identifisert fire hovedaspekter med produktet: kart, informasjon som skal vises til brukeren, API-ene som trenges og "annet", som består av ulike funksjonaliteter som navigasjonsmeny. Tankekartet var ment som et dynamisk verktøy, og derfor er noen av funksjonalitetene senere krysset ut. Disse ble først identifisert som essensielle, men ble i løpet av prosessen vurdert til å være mindre viktige.



Figur 25: Tankekart laget under diskusjon angående applikasjonens MVP.



Både under og etter diskusjonen var det utfordrende å begrense seg, og skille det viktigste fra det mindre viktige. Alle medlemmene delte et ønske om å lage noe spennende og brukbart, og dermed var det et ønske og engasjement for å ta med flest mulig funksjonaliteter i MVP. Noen av gruppemedlemmene fungerte som djevelens advokat, og sørget for en realitetssjekk med tanke på hva som var realistisk å oppnå. Disse medlemmene balanserte diskusjonen med å dra inn eksterne faktorer som f.eks. hjemmeeksamen i andre fag, og påpeke hvordan det kan redusere kapasiteten i dette prosjektet. Dermed dreide diskusjonen seg ikke bare om hva man vil gjøre i dette faget, men også om hva er realistisk å få til, og hvilke andre forpliktelser teamet har. Diskusjonen sett ut ifra metaperspektiv dreide seg om å sikte høyt og muligens feile, eller sikte lavere og fullføre det man har forpliktet seg til.

Denne uformelle diskusjonen resulterte i både tydelig definerte funksjonelle krav som inngår i MVP, samt krav som er ikke nødvendigvis de viktigste, men som allikevel har en verdi. En slik liste med krav ledet naturligvis til neste trinn, som besto i å skrive et formelt dokument med funksjonelle krav til applikasjonen Soleklart. Denne prosessen er beskrevet i neste kapittel.

6.4.2 Kravspesifikasjon

Denne delen av prosessen tok utgangspunkt i en uformell liste som var resultatet av å definere MVP. På dette stadiet var det enighet om at kravspesifikasjon skulle være et formelt dokument som kan sees på en slags kontrakt for det som skal produseres. Dokumentet skal kunne være dynamisk og kunne endres, men det skal allikevel ha en struktur som viser tydelig hva som prioritert. Dokumentet skal inneholde entydige og korte setninger som beskriver en funksjonalitet.



Figur 26: Oversikt over funksjonelle krav og deres inndeling i "må", "bør" og "kan" kategorier.

Gruppen kom frem til to mulige metoder for å dele opp funksjonaliteter. Den ene er å liste alt i en liste nummerert etter prioritet, mens den andre er å dele opp kravene i kategoriene "må", "bør" og "kan" (Figur 26). Den siste metoden ble vurdert som mer fleksibel og tydelig. Kravene er inndelt etter kategorier som beskriver hvilke krav som er viktigst å implementere. Her vil krav knyttet til MVP plasseres i "må"-gruppen, mens de andre kravene vurderes til "bør" eller "kan".

"Må"-kravene er essensielle for at gruppen skal ha et ferdig produkt, og vil derfor være høyest prioritert i produkt-backloggen. Disse kravene er rangert i synkende rekkefølge, hvor de som er plassert øverst ansees som de viktigste, men denne rekkefølgen trenger ikke å bli fulgt, og kan være til vurdering etter behov. Kravene som er satt til "bør" er krav som burde implementeres, men som ikke er strengt nødvendig for å oppfylle visjonen. Det betyr at om gruppen skulle bli ferdig med å implementere kravene som er satt til "må", kan man begynne å implementere kravene som er satt til "bør". Til slutt har man kategorien "kan" som rommer



de kravene med minst prioritet. Disse krav sees på som mer aktuelle i fremtiden dersom prosjektet skal videreutvikles, fremfor å være aktuelle på dette tidspunktet av prosjektet.

Selv om gruppen har ferdigstilt en tabell med disse kravene, betyr det ikke at tabellen er uforanderlig. Tabellen er ment for å hjelpe gruppemedlemmer å holde samme retning, samtidig som det er åpent for diskusjon om kravene og hvorvidt noe kan leges til, fjernes eller flyttes. Særlig den siste kategorien "kan" har blitt laget med tanke på at her kan det være nesten hva som helst, og virker som en idébank som kan komme til nytte ved videre utvikling av applikasjonen.

6.4.3 Retrospektivt blikk på MVP

Gjennom utførelsen av prosjektet har gruppen fått innsikt i at gruppemedlemmer har ulik forståelse for hva MVP er, hvor det er to hovedtolkninger av MVP, og hva det betyr for dette prosjektet. Noen av medlemmene tolket MVP som bare en liten del av applikasjonen eller en slags begynnelse, og at målbildet inkluderte flere funksjonaliteter. Den andre tolkningen var at MVP er det som løser det definerte problemet på en minimal måte – det vil si alt det nødvendige er med, og at applikasjonen da kunne sies å være ferdig.

Denne misforståelse kan virke som en indikasjon på at gruppens arbeid mangler to viktige ting: 1. Gruppen forsikret seg ikke om at alle har samme forståelse av fagterminer som MVP, men i stedet for det antok at alle har lik tolkning. 2. Hver enkeltes forståelse av prosjektet, visjonen og ambisjoner ble ikke koordinert eller justert i forhold til hverandre.

Diskusjonen i etterkant har identifisert to mulige grunner til disse manglene. Den ene kan være det at gruppen var veldig engasjert for å starte å utvikle prosjektet, og dermed ikke har investert nok tid til å forsikre seg om at alle har lik forståelse. Den andre grunnen har vært kommunikasjonen mellom medlemmer har feilet, noe som i seg selv kan være et resultat av at medlemmene ikke kjenner hverandre, og er ikke vant til å kommunisere med hverandre, eller at manglende erfaring gjør at det er vanskelig å se tvetydigheter i kommunikasjon.

Det ble tydelig at det ikke ble brukt nok tid på brukerhistorier i fellesskap, der begrunnelser for brukerens behov ville avdekket flere misforståelser med tanke på funksjonaliteten. De første brukerhistoriene (vedlegg 9.5 *Brukerhistorier*) ble skrevet som et notat i etterkant av et møte, men ble aldri diskutert tydelig i fellesskap. Et eksempel er "Jeg som bruker ønsker å se når solen står opp og går ned", der noen tolket dette til hovedfunksjonaliteten, mens andre mente dette kun var en nødvendig detalj for at man lettere kunne beregne når det var mørkt nok for stjernekikking.

Misforståelsen ble til slutt oppdaget og oppklart. Funksjonelle krav og definisjon av MVP ble gjennomgått, alle tolkninger og utsydelige deler ble diskutert, og brukerhistorier ble skrevet i fellesskap. Etter dette, hadde hele gruppen samme tolkning av visjonene og MVP.



6.5 Valg av arkitektur

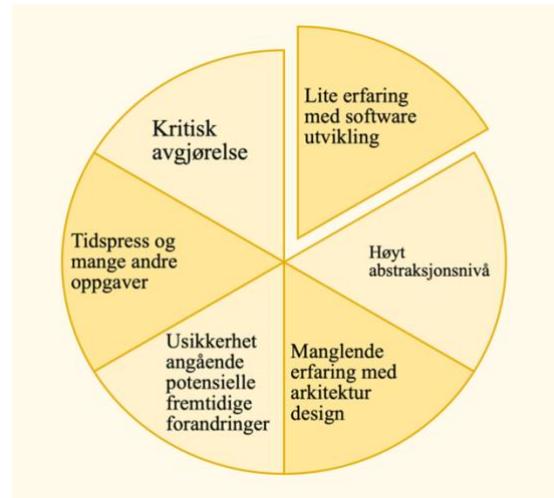
6.5.1 Diskusjon i tidlig fase

Applikasjonsarkitektur er et av de aspektene som har vært diskutert helt siden starten av prosjektet. På det første gruppemøtet 27. februar 2022 ble arkitektur utnevnt som en ting som må diskuteres og bestemmes tidlig og helt siden da frem til midten av mars har gruppen snakket om, tegnet og tenkt på arkitektur. Arkitekturarbeid og design foregikk da "internt" - gruppen nølte med å sette streken og bestemme seg for noe konkret. I tillegg til at arkitekturkonseptet virket omfattende og uklart, var det å forplikte seg til et arkitekturdesign en skremmende ting. *Figur 27* gir en kort oversikt over noen utfordringer som gruppen har opplevd i forhold til design av arkitektur.

Gruppen har stilt spørsmål ved om en forpliktelse til en arkitektur så tidlig i utviklingen strider mot agile prinsipper. Etter grundig undersøkelse kom bekreftelsen på at arkitekturen ikke skulle utvikles på en inkrementell måte, og at selv i smidig arbeid vil arkitekturen bestemmes tidlig (Sommerville, 2015, s. 168).

Det egentlige "oppmuntringen" til å sette seg i arkitekturen var møtet med veilederne. De engasjerte gruppen med å gjenta det gruppemedlemmene allerede visste, men trengte å høre en gang til - arkitektur må på plass nå - det er krevende å gå tilbake for å endre på en dårlig arkitektur. Det gir mening at arkitektur er rigid og ikke kan endres lett. Derfor må den bli implementert som noe stabilt og robust, og det må fungere for denne casen. På dette tidspunktet gruppen ble også klar over at en forpliktelse til en arkitektur er ikke noe som begrenser oss – tvert imot kan det bygge et godt grunnlag for videre inkrementelt arbeid. Gjennom diskusjonen ble det oppdaget at gruppens medlemmer deler samme mentalitet angående det å kunne bygge prosjektet slik at det være mulig å holde åpent for fremtidige endringer. Mulighet for skalering og smidig addisjon av inkrementer avhenger av at man har en solid basis som tåler å bære vekten av det som kommer med hvert inkrement. Ikke minst vil en solid arkitektur kunne virke som et grunnlag for videreutvikling av arkitekturen.

Veilederne foreslo at kun en eller to personer i gruppen fikk ansvar for arkitektur - disse ville ha ansvar for å samle inn mest mulig kunnskap, formidle det, og passe på at alle bygger og konstruerer prosjektet etter den valgte arkitekturen. Teamet identifiserte flere fordeler med denne strategien - en person vil ha rikelig med tid og energi til å fordype seg i dette, tid og energi er investeringer som kan sannsynligvis garantere at solid kunnskap vil bli oppnådd. Samtidig kan det virke som en tidsbesparelse - kun en person bruker masse tid på dette, mens de andre medlemmene kan "gå videre" med andre oppgaver. Det siste er ikke helt sant, det kan virke mot sin hensikt - man kan ikke bare "gå videre" da mange funksjonaliteter er



Figur 27: Kort oppsummering over gruppens utfordringer knyttet til arkitektur design i tidlig arbeidsfase.



avhengig av arkitekturen. Det ville faktisk vært bortkastet tid - alt de andre hadde gjort hadde kanskje ikke passet inn på det tidspunktet en arkitektur ble presentert.

Gruppen har tatt en annerledes tilnærming til dette spørsmålet: alle skal undersøke om arkitektur og skal samtidig bruke litt tid på å sette seg inn i noen andre mindre temaer, som f.eks. kalender-implementasjon eller pull-up bar. Da fikk alle en sjanse til å lære om arkitektur og leke i sin egen "sandkasse" ved å bygge om prosjektet etter en valgt arkitektur. Dette er en tilnærming som kan potensielt kreve mer tid og forpliktelse fra hvert enkelt medlem, men teamet så på dette som en fornuftig investering - hvis alle lærer om arkitekturs så det fremtidige arbeidet og diskusjoner være basert på felles forståelse.

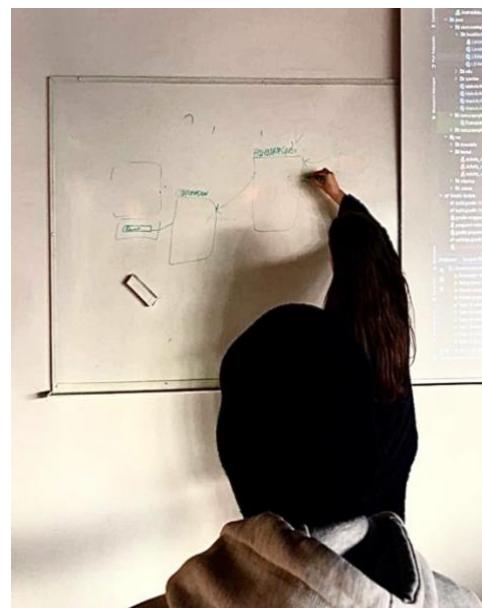
6.5.2 Valg av design pattern

Model-View-ViewModel (MVVM) var det eneste pattern gruppen kjente til og har implementert tidligere. Det er nok ikke tilfeldig at MVVM er kjent, det er et veldig populært og mye brukt pattern (Kouraklis, 2016). Gruppen har likevel ønsket å undersøke andre patterns og se om tenkemåter fra disse kunne appliseres i prosjektet. Tabell 9 i vedlegg 9.6 *Diskusjon av design patterns* presenterer et sammendrag av patterns som ble studert samt en kort analyse om hvor aktuell hver av disse er i dette prosjektet.

Sommerville (2015, s.171) skriver at for å kunne bygge en god arkitektur, må man kjenne til systemet sitt og dets krav. Gruppens idé bak applikasjonen var blitt ganske moden, kravene hadde blitt diskutert, og det var enighet om hva som måtte til for at produktet skulle bli vellykket. Det at gruppen hadde brukt tid på definisjon av MVP, funksjonelle og ikke-funksjonelle krav, var avgjørende for en lett diskusjon av design patterns.

Resultatet av diskusjonen ble, ikke overraskende, at gruppen valgte MVVM. En av de største årsakene er de eksterne begrensningene i prosjektet. Prosjektet er kortvarig, og hvert medlem har begrenset med tid og kapasitet. Implementeringen av et helt nytt og ukjent design pattern ville krevd mer tid. I tillegg til det, er MVVM det design pattern som ble undervist, og dermed har både faglærere og veiledere god kjennskap til det. Det å velge et kjent arkitekturdesign som alle kan, gir en stor fordel i det å kunne å ha en effektiv kommunikasjon både med veiledere og med hverandre i teamet. I tillegg til det har gruppen diskutert at prosjektet bringer med seg mange ukjente aspekter som må læres og dermed er det et ønske om å velge noen aspekter, som MVVM, som er allerede kjent.

Selv om design pattern var valgt, gjenstod det en del arbeid som ville være knyttet til dekomponering av systemet og design av arkitekturen. På dette tidspunktet var en del av de grunnleggende komponentene allerede blitt utviklet, og disse dannet grunnlaget for den videre designprosessen. Fasen bestod av et eget møte med tegning på tavlen, design ved bruk av verktøy som draw.io og undersøkelse av andre lignende applikasjoner.



Figur 28: Et bilde fra en interaktiv sesjon som var en del av arkitekturdesignsprosessen. Bildet viser tegning av arkitekturdiagram i en tidlig fase.

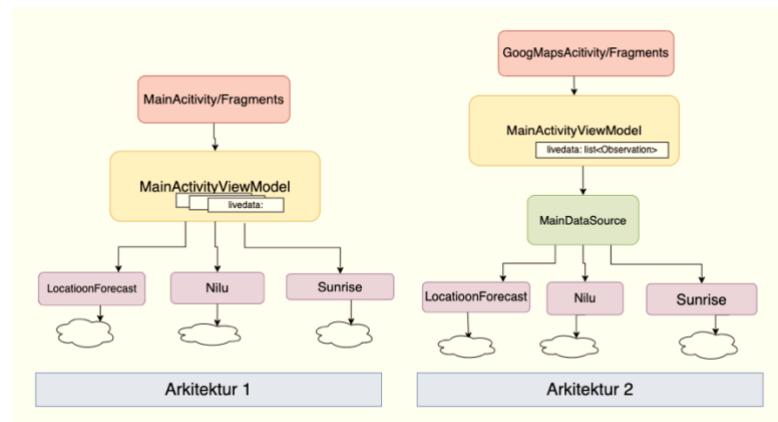


Figur 28 viser et slik scenario, hvor et av gruppemedlemmene tegner og forklarer en mulig arkitektur, mens resten av gruppen stiller spørsmål og diskuterer fordeler og ulemper. I løpet av dette møtet presenterte flere av gruppemedlemmene sin versjon. Ikke alle medlemmer var vant til å tegne datastrukturer og forklare det, men teamet ble enig om at alle prøver det siden den visuelle presentasjonen virket som den mest effektive kommunikasjonsmåte i dette tilfellet. Etter litt øvelse til og med de minst "kunstneriske" medlemmene har blitt ganske komfortable med tegning på tavlen. Neste kapittel beskriver resultatene av dette møtet, og presenterer det endelige arkitekturdesignet.

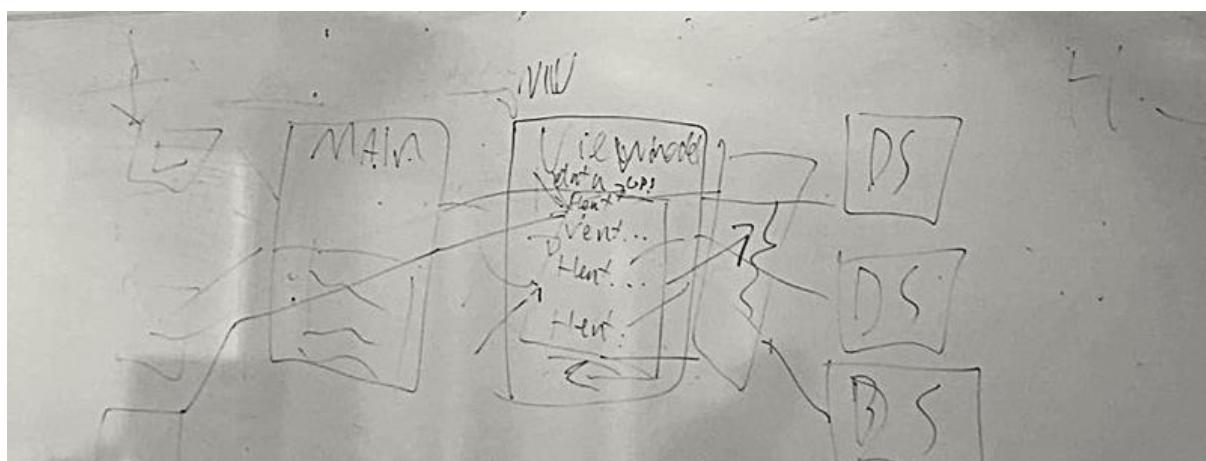
6.5.3 Arkitekturdesign

I denne delen presenteres det to konkrete MVVM arkitektur-implementasjoner: Arkitektur 1 (A1) og Arkitektur 2 (A2), hvor hver av disse ble konstruert av gruppen. *Figur 29* presenterer A1 og A2 på et overordnet nivå. A1 ble vurdert til å passe best for applikasjonens visjon, kravene og mulige begrensninger. Det presenteres likevel to arkitekturen, fordi det er lettere å begrunne det endelige valget i sammenligning med en annen løsning.

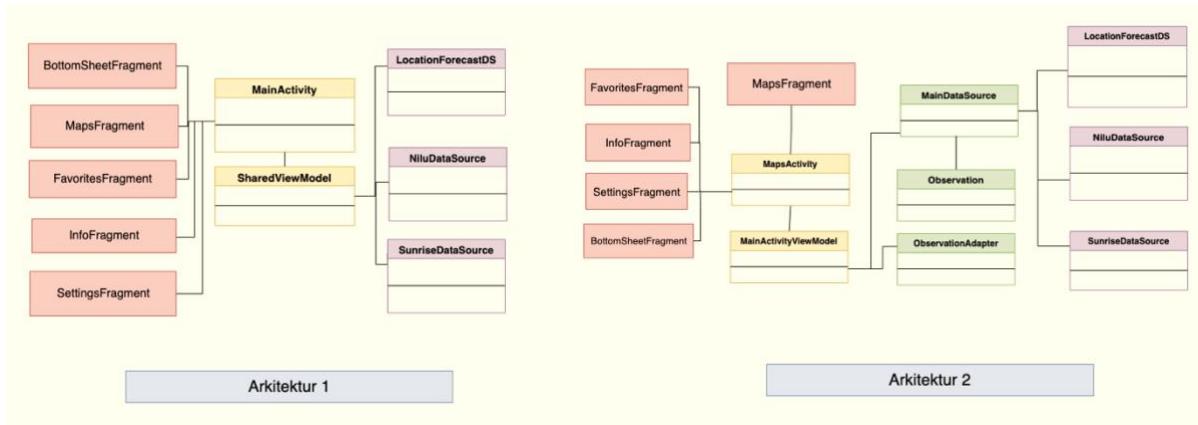
Etter at de ulike arkitekturene ble presentert, ble det satt i gang en prosess for å konstruere noe som passet best for dette prosjektet. Konstruksjonen foregikk ved å tegne opp den nåværende strukturen på en tavle, og endre det underveis ved å legge til nye komponenter. Diskusjonen som foregikk samtidig, var basert på hvilke designprinsipper gruppen vil ta hensyn til, hva som er realistisk å gjøre innen den gitte tiden, og hvilke potensielle fallgruver som finnes. *Figur 30* viser en tegning på en tavle som et resultat av en slik sesjon. Arkitekturen i figuren kan sees å ha komponenter som DS (Data Source), ModelView og Main.



Figur 29: Sammenligning av A1 og A2 på et overordnet nivå.



Figur 30: Midlertidig resultat fra dekomponering av applikasjonens struktur utført i plenum.



Figur 31: Detaljert arkitektur-diagram for A1 og A2.

Resultatet av sesjonen ble de to allerede nevnte arkitekturer A1 og A2. Det finnes flere hovedforskjeller mellom disse arkitekturene. For det første har A2 en felles komponent *MainDataSource* som interagerer med alle de andre mindre ressursklassene som kaller på API, mens A1 har ikke et slikt mellomledd mellom *ViewModel* og nettressursklassene. I tillegg til det vil A2 sin *ViewModel* interagerer kun med en dataressursklasse og har kun livedata av en type. På overordnet nivå er valget mellom A1 og A2 valget mellom å ha en delt datastruktur (A2) og ikke ha delt data struktur, men heller sende data direkte til der dataene er bestilt (A1). *Figur 31* viser en mer detaljert modell for arkitekturene A1 og A2.

Gruppen endte opp med å velge modellen i A1, fremfor A2 da denne ble oppfattet som enklere å jobbe med i framtiden, og mer fleksibel. Hvis man vil kunne gjøre endringer i hva som skjer underveis, og mellom API kallene, ville dette vært mindre fleksibelt og måtte gått gjennom flere ledd med en *MainDataSource* klasse. Selv om det var fint å kunne hente en dataklasse med all nødvendig data, ville man da måtte kalle på alle API-ene hver gang man skulle hente noe fra noen av dem. I fremtiden ville man også måttet legge inn metoder i enda en klasse, hvis man skulle legge til nok et API. For å følge prinsippene med lav kobling, høy kohesjon og fleksibilitet, endte det derfor opp med A1-modellen.

Selv om at A2 ble valgt har gruppen allikevel blitt enig om at A1 er også en arkitektur som kan være nyttig i dette tilfellet. Generelt sett den siste diskusjonen om arkitektur opplevdes som ganske krevende fordi disse arkitekturene er veldig like og dermed det er vanskelig å kunne skille små aspekter og forstå deres innvirkning for fremtidig applikasjon.

6.5.3.1 Retrospektivt blikk på arkitekturarbeid

Gjennom prosjektet har gruppen hatt relativt stort fokus på arkitektur, og har brukt mye tid på å lære om og designe arkitektur. I etterkant kunne det kjennes som om at gruppen har brukt unødvendig mye tid på å lete etter den perfekte arkitekturen når teamet allikevel innerst inne visste at MVVM er det beste valget.

Til og med etter endt prosjekt, synes gruppen at tiden brukt på arkitekturdesign er ikke bortkastet. Det har blitt diskutert at undersøkelser og felles diskusjon førte til at alle medlemmer var klar over hvordan prosjektet er bygget opp. I tillegg ga deltagelsen i arbeidet med arkitekturdesign medlemmene større eierskap over prosjektet. Teamet har diskutert at én arkitekturansvarlig person også hadde ført til god nok arkitekturdesign, men da ville ikke



resten av teamet hatt en følelse over å være med og bygge skjelettet til applikasjonen. I tillegg vil arkitekturkunnskapen til resten av teamet være avhengig av den ene personens evne til å overføre og utveksle kunnskap.

Teamet har også diskutert at det å takle et større spørsmål som arkitekturdesign sammen var nyttig for fremtidig arbeid. Det å presentere, tegne og forklare ideer i felleskap var en nyttig øvelse. I minst var det å lete etter fordeler og ulemper en perfekt måte å øve seg på å se prosjektet fra flere perspektiver.

6.6 Visuell utforming

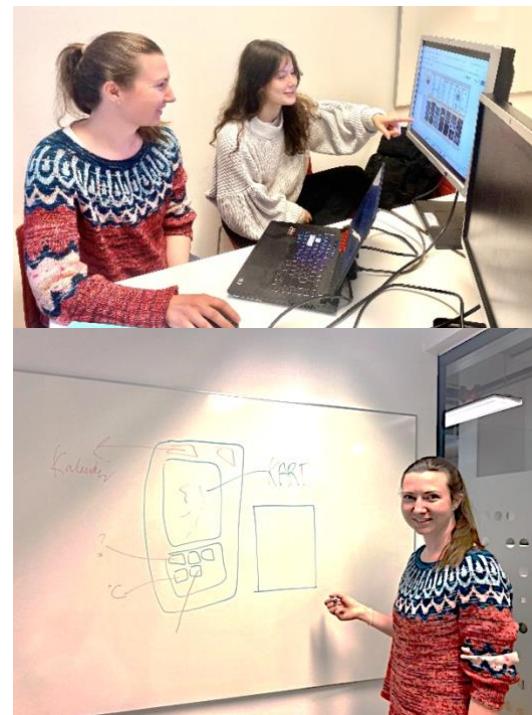
Den visuelle utformingen gjennomgikk flere iterasjoner, med ulike mål og opplosninger, og endret seg basert på både gruppens ønsker og innsikt fra brukerundersøkelsene. Skissene finnes som vedlegg 9.7 *Visuell utforming*. Disse skissene ble laget i Proto.io, og var klikkbare. Det vil si at man kan simulere bruk av appen uten å implementere kode. Dette legger til rette for å oppdage svakheter som ikke oppdages like lett på papirskisser. *Figur 32* viser et bilde fra en slik arbeidssesjon.

Det første utkastet hadde som mål å danne en felles forståelse for konseptet innad i gruppen, og var enkle skisser med plassholdere. På dette punktet i prosjektet var appens hovedfokus å informere om forhold for stjernekikking, og gi informasjon om ulike stjernebilder. Disse stjernebildene kunne filtreres basert på hvor i verden og under hvilken årstid de er mulige å observere. Tanken var at resultatene for en lokasjon kunne vises som flere tidspunkter per døgn, eventuelt kun forhold om natten over flere døgn, slik at det skulle være lett å planlegge en stjernekikkingstur.

Andre utkast viser en annen mulighet for visning av resultater. I denne versjonen ville resultatene komme som en boks som kan trekkes opp og ned, og kun resultater for soloppgang og solnedgang ville vises.

Tredje utkast hadde en høyere oppløsning. På dette tidspunktet var fokuset flyttet til soloppgang og solnedgang. Utkastet har farger og en mer detaljert fremstilling av skjermbildene, og utforsket ulike måter å presentere værvarsel for favoritter. Dette kunne vises ved at:

- listen med favoritter utvidet seg ved klick på et element
- favoritten åpnes i nytt vindu
- alle favoritter er åpne hele tiden
- brukeren sendes til kartet, og resultatet vises på samme måte som ved søk



Figur 32: Arbeid med skisser.



Gruppen var enige om at en liste som utvidet seg ville være å foretrekke med tanke på brukeropplevelsen, mens det ville være raskest å implementere ved å gjenbruke koden for å vise søkeresultat. I dette prosjektet ble ikke kode for å kunne lagre favoritter implementert, men det ville være høyt prioritert dersom utviklingen skulle fortsette.

Fjerde utkast ble brukt som grunnlag for kodingen. Appen er likevel ikke identisk med skissene, som for eksempel mulighetene på *Innstillinger*. Enkelte detaljer er heller ikke med i appen, slik som korrekte farger på fabrikk-ikonet i nattmodus

6.7 Universell utforming

Universell utforming handler om å utforme samfunnet, arkitektur, tjenesteutvikling, design etc. på en slik måte at så mange som mulig kan delta uavhengig av funksjonsevne (*Figur 33*). Det er lett å glemme at midlertidige og situasjonelle faktorer også kan ha behov for tilrettelegging. I kapittel 2 *Brukerdokumentasjon* er en kort beskrivelse av hvordan Soleklart stemmer med WCAG 2.1, mens dette kapittelet vil beskrive litt av arbeidet med universell utforming.

6.7.1.1 WCAG 2.1-krav

I et prosjekt som dette, må det hele tiden gjøres prioriteringer, og kravene knyttet til skriftstørrelse, skjermleser og alternativ formidling av visuell informasjon (kart) ble som nevnt ikke oppfylt. Teamet begrunner dette med at appen skal benyttes for å tilrettelegge for visuelle opplevelser, og at brukeren derfor forventes å ha godt syn. Men det finnes også svaksynte mennesker som strever med mobilskjermer, som likevel kan ha glede av lys og farger ved soloppgang og solnedgang. Det har ikke blitt prioritert å finne en bedre løsning på dette problemet.



Figur 33: En løsning som fungerer for noen med en permanent funksjonsnedsettelse, vil også fungere for andre med midlertidig eller situasjonsbetinget funksjonsnedsettelse (www.curioresearch.net)

6.7.1.2 Brukbarhet

Brukervennlig design bygger på brukerens forventninger og ferdigheter, og bør utformes basert på systemer brukeren kjenner fra før. Gruppen har i dette prosjektet benyttet kjente oppsett for skjermbilder, menyer og ikoner. Men det finnes utallige design for værikoner, og ingen godt etablerte ikoner for luftkvalitet. Dermed falt det naturlig å gjennomføre en spørreundersøkelse for å finne ut hvilken stil potensielle brukere mente egnert seg best. Resultatet av denne undersøkelsen finnes som *9.8 Spørreundersøkelse knyttet til visuelle ikoner*. Se også kapittel 6.8 *Brukerundersøkeler*.

Det er forbedringspotensialet ved input i tekstsøkefeltet. Slik appen er nå, er man nødt til å skrive stedsnavnene korrekt. Ved å bruke Google Places API sin funksjon for automatisk fullføring av tekst, ville brukeren fått en liste over forslag dersom stavemåten ikke var helt



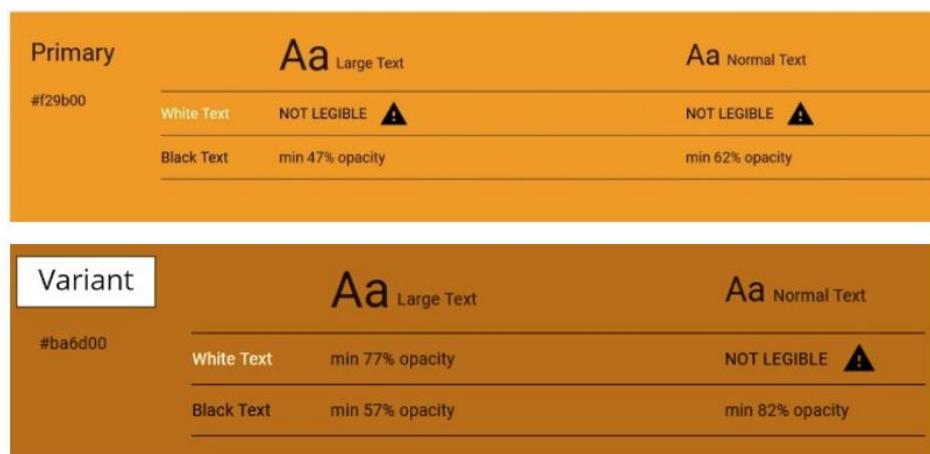
korrekt. Det ville hjulpet mange med å unngå frustrasjon, men er altså ikke implementert i den nåværende løsningen. Dette ville være naturlig å inkludere i en senere iterasjon.

6.7.1.3 Nattmodus

Appen skal kunne brukes av et stort spekter av brukere med ulike behov og preferanser. Det kan være brukeren helt enket foretrekker et mørkt skjerm bilde, er lyssensitiv, eller at appen skal brukes i mørke omgivelser, der man ønsker å bevare nattsynet. Man kan se ganske godt i mørket, men det krever at øyet får tid til å tilpasse seg de nye lysforholdene. Nattmodus gir et utseende der bakgrunnen er mørkere enn innholdet (positiv kontrast). Dette bevarer nattsynet bedre, og er mildere for øynene enn med en lys bakgrunn (negativ kontrast). Derfor er appen utviklet slik at den tilpasser seg dersom brukeren setter telefonen i nattmodus.

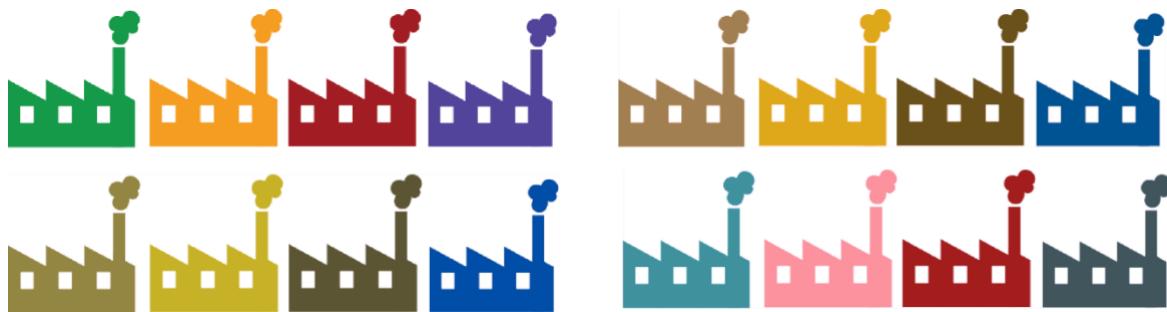
6.7.1.4 Farger og fargeblindhet

For å teste om fargekontrastene i appen var gode nok, gikk de gjennom en kontrast-analyse. *Figur 34* viser et slikt resultat. Analysen ble gjennomført ved bruk av *Color Tool* (Material Design, 2022)



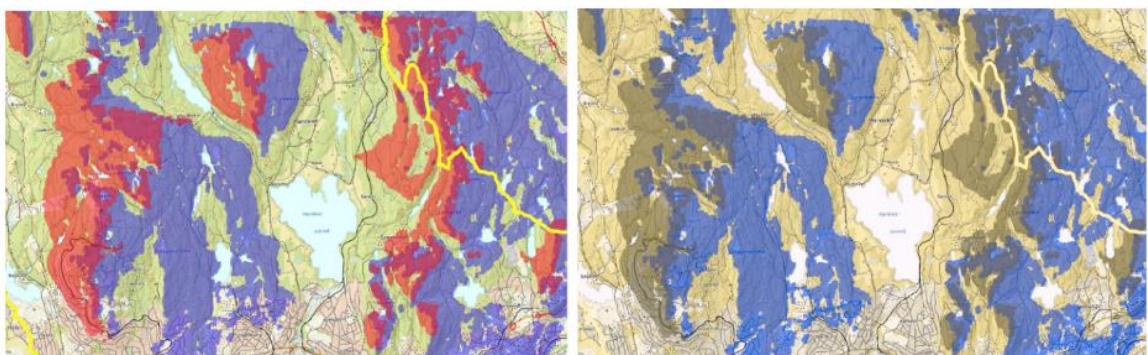
Figur 34: Eksempel på resultat av en analyse av fargekontraster.

Det er mange typer og grader av fargeblindhet. Total fargeblindhet (monokromasi) innebærer at en person kun ser gråtoner, men dette er ekstremt sjeldent. De mest vanlige formene er rødblindhet (protanopi) og grønnblindhet (deutanonpi), men det finnes også blåblindhet (tritanopi). Fargeblindhet og fargesvakhet oppstår når tappene i øynene har ingen eller nedsatt følsomhet for lys av en spesiell bølgelengde. Dette rammer mellom 5 og 8% av alle menn, og omtrent 0,3% av alle kvinner (Norges Blindeforbund, 2022). Mange er ikke klar over at de har nedsatt fargesyn, og lever helt fint med det. Problemene oppstår når visuell informasjon ikke er utformet med hensyn til denne gruppen mennesker, som vist i *Figur 35*, der rød og grønn oppfattes som lik farge av rød-/grønn-fargeblinde. Derfor stilles det krav om at informasjon ikke skal formidles gjennom kun farge, og resultatene for luftkvalitet presenteres med både tall og farger. Miljødirektoratet informerer om luftkvaliteten, og opererer med fargeskalaen grønn, gul, rød og lilla. For å ikke bryte kjente designmønstre noen brukere kjenner fra før, har gruppen valgt å bruke de samme fargene, selv om det kunne være hensiktmessig å introdusere en ny og fargeblindvennlig skala.



Figur 35: Symbol for luftkvalitet sett med fullt fargesyn, rød-, grønn- og blåblindhet. Simulert ved bruk av Coblis (Colblindor, 2022).

I solkartet (*Figur 36*) er soloppgang visualisert i blått, og solnedgang i rødt. Selv om rødt kan være lite egnet sammen med grønn i mange sammenhenger, fungerer det i dette tilfellet. Det er fordi det her ikke er selve fargetonen som er viktig, men at lagene kan skilles fra hverandre. Gruppen testet dette ved å simulere rød-fargeblindhet, og gjennom en evaluering med en rød/grønn-fargeblind bruker.



Figur 36: Solkartet sett med fullt fargesyn og rødblindhet. Simulert ved bruk av Coblis (Colblindor, 2022).

Dataene for svevestøv kan forstås som mengde per kubikkmeter eller ordnede nivåer (for dem med fullt fargesyn; grønn, gul, rød, lilla). Men informasjonen om hvilke tiltak som er nødvendig ved ulike nivåer, er ikke godt kjent. Å gi brukeren en informasjon, uten å samtidig sørge for at de forstår hva den skal brukes til, er uheldig. Derfor ønsket gruppen å gjøre det lett tilgjengelig ved å lenke til dette i appen.



6.8 Brukerundersøkelser

Det hjelper lite å utvikle et produkt dersom ingen vil bruke det. Gruppen gjennomførte flere undersøkelser for å sikre at både konseptet, den visuelle utformingen og interaksjonsdesignet hadde støtte hos brukerne.

6.8.1 Kartleggende spørreundersøkelse

I den første undersøkelsen var målet å kartlegge mulige brukere, deres mål, behov og preferanser, slik at gruppen hadde et mer korrekt bilde løsningens målgruppe, som igjen ville gi et mer treffende sluttprodukt.

6.8.1.1 Metode

Gruppen ønsket å kartlegge målgruppens demografi, interesser, samt holdninger til mulige funksjonaliteter. Siden undersøkelsen hadde til hensikt å kartlegge et bredt spekter av preferanser og interesser hos målgruppen, men ikke skulle undersøke målgruppens følelser og adferd i dybden, var det naturlig at den første undersøkelsen ble en kvantitativ undersøkelse i form av en spørreundersøkelse.

En spørreundersøkelse er nyttig når man ønsker å innhente store mengder med data på en rask og ressursbesparende måte. Siden gruppen enda ikke hadde gjort undersøkelser av målgruppen, var det derfor hensiktsmessig å gjennomføre en spørreundersøkelse, da det trolig ville gi flere respondenter enn ved bruk av andre, mer ressurskrevende undersøkelser.

6.8.1.2 Gjennomføring

Undersøkelsen var en spørreundersøkelse på nett, der deltakeren var anonym. Gruppen hadde gjennom det innledende arbeidet avdekket flere målgrupper. For å kunne kartlegge disse separat, ble to identiske undersøkelser sendt ut; én til en generell befolkning, og én rettet spesifikt mot personer med interesse for friluftsliv. Lenker til undersøkelsene ble postet i relevante grupper på Facebook, etter klarering fra administratorene av gruppene.

Spørreundersøkelsen ble utformet via UiO sin nettskjemaløsning. Under utforming av spørreundersøkelsen var det viktig å begrense mengden spørsmål. Om spørreundersøkelsen har for mange spørsmål, kan respondentene gå lei av å svare, og man kan risikere at dataene blir påvirket av dette. Spørreundersøkelsen besto derfor av 17 spørsmål med radioknapper, samt et avsluttende spørsmål hvor respondentene ble bedt om å komme med eventuelle tips eller innspill til konseptet for løsningen.



Figur 37: Konseptet har støtte hos målgruppen.



For å kunne undersøke hvorvidt antagelsene rundt målgruppen var treffende, var det hensiktsmessig å hente inn informasjon om hvem de mulige brukerne ville være. Derfor var de syv første spørsmålene i undersøkelsen sentrert rundt målgruppens demografiske fordeling. Videre var det hensiktsmessig å undersøke hvorvidt målgruppen hadde interesse i å bruke forskjellige tiltenkede funksjonalitetene for løsningen. Respondentene ble bedt om å rangere mulige funksjonaliteter, slik som lagring av favoritter, nattmodus, informasjon om parker og liknende, slik at gruppen kunne gjøre bedre prioriteringer i forhold til hvilke funksjonaliteter som burde vektlegges i løsningen.

6.8.1.3 Resultater

Respondentene fra den generelle befolkningen svarte at de oftest gikk på tur i urbane områder, og foretrakk solnedganger fremfor stjernehimmel. Dette kan ha noe med at de fleste respondentene bor i Stor-Oslo, der stjernehimmelen er vanskelig å se, og det kreves litt ekstra innsats og reisetid for å oppleve stjerner. Den friluftsinteresserte befolkningen var mer spredt i landet, og hadde en bredere aldersfordeling. Turen deres gikk som regel i skogen, men de ønsket også å gå turer i fjellet eller på vidda. I denne gruppen var andelen som ønsket å se stjerner flere enn blant den generelle gruppen, men det var fortsatt et klart flertall som ønsket å oppleve solnedganger. Gruppens utgangspunkt hadde primært vært stjernehimmel, men ble etter dette mer rettet mot brukernes ønske; solnedganger.

En respondent uttrykte et ønske om å kunne lagre favorittplassene sine, uten at disse var synlige for andre brukere av appen. En annen respondent ønsket seg et kart der man kan se optimale steder for hvor sola treffer, slik at hen kunne våkne til en flott soloppgang på hengekøyetur. Dette var to behov teamet ønsket å se nærmere på. Respondenter ble også spurtt om de kunne tenke seg å bruke denne applikasjonen til planlegging av turer. Flesteparten - 81,7% svarte ja (*Figur 37*).

6.8.2 Etnografisk undersøkelse

Den kartleggende undersøkelsen hadde avdekket noen interessante idéer. For å lære mer om målgruppen, og om funnene i spørreundersøkelsen var representative for målgruppen, ble det gjennomført en etnografisk undersøkelse.

6.8.2.1 Metode

Etnografiske undersøkelser har sin opprinnelse fra antropologiske studier. Et essensielt aspekt under etnografiske undersøkelser er graden av deltagelse hos de som undersøker et fenomen, der det er ønskelig at man tilnærmer seg undersøkelsen ved å dykke ned i dybden av det man undersøker, gjennom for eksempel å delta i en aktivitet som om man selv er en del av den gruppen som undersøkes, deltagende observasjon er et konkret eksempel på en metode som kan brukes for å oppnå dette.

Hensikten med etnografiske undersøkelser er å oppnå detaljert og nyansert deskriptiv data av fenomenet som undersøkes. Derfor kan etnografiske undersøkelser være særlig nyttige om man ønsker å undersøke fenomener knyttet til kontekst, miljø, brukerbehov og krav.

Når man gjennomfører etnografiske undersøkelser er det særlig viktig at man er bevisst mulig bias og etiske forhold, da en stor grad av deltagelse hos de som gjennomfører undersøkelsen



kan føre til skjevheter i dataene, samt at man i større grad knytter relasjoner med subjektene under slike undersøkelser (Lazar et al, 2017, s.230).

6.8.2.2 Gjennomføring

To av medlemmene i teamet meldte seg inn i ulike Facebook-grupper for friluftsinteresserte, og gjennomførte en etnografisk undersøkelse som passive observatører.

6.8.2.3 Resultater

I alle gruppene, både for generelt friluftsliv og for hengekøyeliv, var det mange innlegg om opplevelser (*Figur 38*), med bilder av godt vær og solnedganger, og spørsmål om steder med gode solforhold. Dette underbygget resultatene fra spørreundersøkelsen, der en respondent ønsket en oversikt over solforhold.

Et av gruppemedlemmene hadde litt erfaring med geografiske informasjonssystemer (GIS), noe som gjorde det mulig å utvikle et kart med ulike lag som visualiserer optimale steder for å se soloppgang eller solnedgang. En tidligere versjon av kartet ble lagt direkte inn i appen, men det var tungt og tidkrevende å åpne. Det gikk raskere å åpne i en nettleser, så i den nåværende versjonen av appen, åpnes det i en ekstern nettleser. For å sikre at det ikke tar for lang tid å laste inn, er solkartet også forenklet. Det dekker kun Oslo, og viser kun solens posisjon på sommeren. Dette er litt snevert, men denne versjonen av appen er kun ment som et *proof of concept*.

6.8.3 Spørreundersøkelse knyttet til visuelle elementer

Gruppen ønsket å avdekke hvilke preferanser brukerne har til visuell utforming, og om symbolene er lette å oppfatte og tolke.

6.8.3.1 Metode

Spørreundersøkelser kan gjennomføres både digitalt og fysisk. Å rekruttere passivt (invitere til deltagelse i sosiale medier) kan være tidkrevende, i og med at man må vente på at respondenter selv oppsøker undersøkelsen. Oppsøkende rekruttering kan oppleves som ubehagelig for noen, men det er lett å takke nei til å delta. Samtidig kan motivasjonen for å interagere med et menneske du har en tilknytning til (medstudent på samme fakultet) være større enn motivasjonen for å fylle ut et anonymt skjema på nett.

6.8.3.2 Gjennomføring

Det ble laget en rekke utkast til app-logo, og samlet flere ulike stiler på ikoner for værvarsel. Disse ble skrevet ut på A4-ark, og vist til studenter ved Instituttet for Informatikk. Respondentene ble bedt om å velge en eller flere favoritter, og en eller flere de absolutt ikke likte. For å unngå bias, fikk de ikke vite hva andre respondenter hadde svart, med unntak av

Tenkte å henge ute i natt på en topp et sted i Hurum eller kanskje mellom Spikkestad og Lier for å få med meg solnedgangen.

Meg, kåya og havet ❤️livet ❤️magisk og kunne våkne opp til soloppgangen og fugle kvitter 🙏😊

Deilig å ligge og sveve i hengekøya. Se på solnedgang og nyte å ligge i en varm og god sovepose, mens jeg kjenner frisk luft stryker meg mot ansiktet.

Flott utsikt, og ny kommune! Men det ble overraskende kaldt her oppover Drammen. Slik kan det gå når man absolutt må i høyden og er ute etter utsikt 😊.

Dette er heller ikke stedet for kveldssol oppdaget jeg, men soloppgangen var flott!

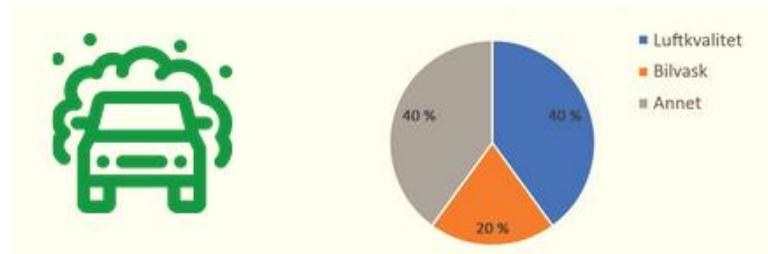
Figur 38: Eksempel på innlegg knyttet til solforhold.



dem som satt sammen og hørte hva sidemannen svarte. De ble også bedt om å tolke et ikon tenkt for å vise luftkvalitet.

6.8.3.3 Resultater

Undersøkelsen viste klare trender, og det kunne kåres en favorittlogo og en favorittstil for værikonene. Symbolet som var tenkt å vise svevestøv, som vist i *Figur 39*, førte til mye usikkerhet blant respondentene. Gruppen diskuterte, og ble enige om å bytte det ut. En mer utfyllende beskrivelse av undersøkelsen finnes som vedlegg 9.8
Spørreundersøkelse knyttet til visuelle ikoner.



Figur 39: Kun 40% av respondentene koblet dette ikonet til «luftkvalitet».

6.8.4 Brukbarhetstesting

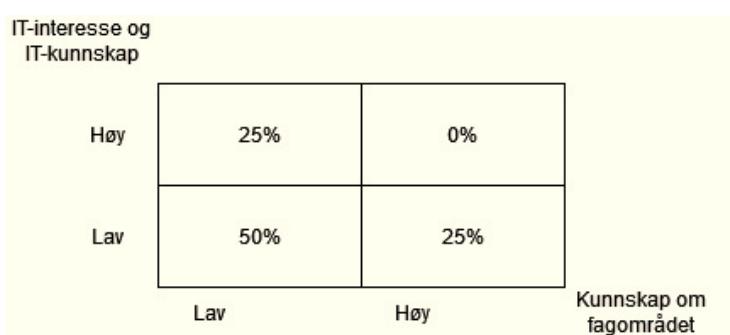
6.8.4.1 Metode

Brukbarhetstesting simulerer reell bruk, og avdekker svakheter og bruksproblemer ved systemet. Dette sikrer at avgjørelser baseres på bedre informasjon enn egne meninger, og dermed resulterer i et mer solid produkt. Hvor stor og omfattende en brukbarhetstest bør være, avhenger helt av prosjektets størrelse, kompleksitet, budsjett og tidsramme. Det kan gjøres så enkelt som en test med kun én bruker og analoge skjermskisser i en lab, eller med 15 brukere og en fungerende prototype i en naturlig setting. Noen svakheter kan man kun oppdage ved bruk i naturlig setting, der situasjonsavhengige faktorer påvirker bruken.

Som utvikler blir man ofte blind for svakheter i sitt eget produkt. Derfor var det viktig at testerne ble hentet utenfra, og at disse var representative for sluttbrukeren. Erfarne brukere vil stort sett kunne jobbe seg forbi svakheter og logiske brister, mens disse blir veldig tydelige når uerfarne brukere deltar. *Figur 40* viser en anbefalt fordeling av deltakere. Selv med en smal målgruppe, vil det alltid være varierende ferdigheter, og systemet skal kunne fungere for alle.

6.8.4.2 Gjennomføring

Soleklart har en bred målgruppe, og det ville være gunstig å gjennomføre flere brukbarhetstester med sluttbrukere med ulik demografi og funksjonsnedsettelser. Av praktiske årsaker har gruppen i dette prosjektet valgt å gjennomføre en test med 5 brukere, der de har gjennomført forhåndsbestemte oppgaver i en emulator på en datamaskin eller på



Figur 40: Gunstig fordeling av deltakere til brukbarhetstesting. (Toftøy-Andersen og Wold, 2011, s.32).



en Android mobil. Kunnskapen deres om IT og fagområdet ble ikke kartlagt, men ble anslått til å være middels. En av deltakerne var rød/grønn-fargeblind, og en annen hadde dysleksi.

Deltakerne ble rekruttert gjennom bekjentskapskretsen til gruppemedlemmene. Dette gjorde både rekrutteringen og gjennomføringen av testingen mye enklere, men det kan samtidig ha gitt andre resultater enn om disse var rekruttert gjennom for eksempel plakater i en matbutikk eller innlegg i sosiale medier.

Testleder og deltaker satt sammen, slik at begge kunne se skjermen godt, og begge parter hadde mulighet til å stille den andre spørsmål. *Figur 41* viser brukbarhetstesting med en av fem brukerne. Ved å gjennomføre testen fysisk unngikk teamet tekniske problemer knyttet til videosamtaler, skjermdeling og nedlastning av programvare. Brukerne ble informert om prosjektet, og bedt om å gjennomføre oppgaver i appen. De ble også oppfordret til å fortelle hva de tenkte og hvordan de navigerte. Etter at oppgavene var gjennomført, var det en kort samtale om hvordan deltakeren opplevde systemet.



Figur 41: Bilde tatt under brukbarhetstesting.

Brukerne ble bedt om å utføre følgende oppgaver:

1. *Onboarding*
2. *Søk etter værvarsle ved trykk i kart*
3. *Søk etter værvarsle med tekstlig input*
4. *Endre dato for søket*
5. *Skru på nattmodus på telefonen, og så velge en ny lokasjon i kartet.*
6. *Åpne solkartet og finne et egnet sted for å se både soloppgang og solnedgang*

6.8.4.3 Resultater

Undersøkelsen avdekket følgende problemer og irritasjonsmomenter:

- 1. Onboarding**
 - a. Bildene i onboardingen er litt små, og teksten for generell. Værikonene blir ikke forklart, som for eksempel hva slags tall "fabrikken" viser.
- 2. Søk etter værvarsle ved trykk i kart**
 - a. En bruker ønsket at symbolet og "Søk" på menyen ble byttet ut med "Home" og hus-symbol.
 - b. Ikonet for luftkvalitet vises i korrekte farger på flere telefoner, men de blir alltid grå på enkelte mobiler, til tross for at den mottar data om mengden svevestøv.



- c. Retro-kartet (det som kommer automatisk) er uvant, og kan være litt vanskelig å lese.
- d. Ikonene var lette å forstå, men resultatene ga lite mening. Hva betyr for eksempel 72% under et vær-ikon? At det er 72% sannsynlighet for at dette varselet stemmer? Luftfuktighet? Skyer?
- e. Teksten i resultatet blir presset tett inntil hverandre på enkelte skjermstørrelser.
- f. Symbolene for vær etter solnedgang bør inkludere en måne, ikke en sol.
- g. Tallene for luftkvalitet betyr ingenting for brukeren. Flere brukere, spesielt den fargeblinde, uttrykte et ønske om at dette var forenklet til ordnet nivå ("Lite", "Moderat", "Høyt", "Svært høyt").
- h. Hva slags luftkvalitet? Gasser eller svevestøv?
- i. "+2 timer" burde vært i en mer brukervennlig tekst som "2 timer etter solnedgang". Dette ble fikset.
- j. Tidspunktet for soloppgang osv. burde kun vise timer og minutter, ikke sekunder.
- k. En bruker søkte værvarsel for Bangkok, og resultatene viste at sola står opp kl.00:58, og går ned kl.13:33. Her ville det være nødvendig å ta hensyn til tidssoner, og vise tidspunktene i lokal tid. Det vil også oppstå problemer med områder med mørketid eller midnattssol.
- l. Det burde være tekst over resultatene, som fortalte hvor resultatene var fra ("Vinteren", "Hovedøya" etc.)

3. Søk etter værvarsel med tekstlig input

- a. Ved trykk på input-vinduet, ble alt på skjermbildet skjøvet oppover. Det førte til at resultatvinduet la seg oppå boksen der man skrev. Brukeren måtte selv dra resultatet ned, så hen kunne se hva hen skrev.
- b. Brukeren med dysleksi fikk ikke opp ønsket lokasjon pga. en skrivefeil.

4. Endre dato for søker

- a. Det burde stått kalender, eller vært et kalender-ikon, kombinert med "i dag", "i morgen" eller "4.mai" i stedet for datoformatet "2022-05-04"
- b. Kalenderen hadde litt liten skrift.
- c. Kalenderen dukker opp oppå resultatene. Det resulterer i to hvite bokser oppå hverandre, noe som blir vanskelig å se på.
- d. Alt skjedde for brått når man valgte en dato: kalenderen lukket seg, og det kom opp resultater. Brukeren ønsker å trykke på en knapp for å bekrefte valg av dato.

5. Skru på nattmodus på telefonen, og så velge en ny lokasjon i kartet.

- a. Fargekontrasten for fabriksymbolet mot bakgrunnen er for lav. Ikoner med gode kontraster for nattmodus er laget, men ikke lagt inn i appen.

6. Åpne solkartet og finne et egnet sted for å se både soloppgang og solnedgang

- a. Brukerne ville helst at dette kartet var en del av appen, og at man ikke trengte å åpne det eksternt.

7. Innstillinger

- a. En bruker forsøkte også å endre kartstil. Det ble påpekt at hen måtte "avvelge" kartstiler før hen valgte en ny kartstil. Dette er noe som burde skje automatisk.



Mange av manglene var gruppen allerede klar over, men det var ikke tid til å rette dem opp før brukbarhetstesting. Systemer bør uansett alltid brukbarhets-testes av eksterne brukere, for å avdekke så mange svakheter som mulig. Teamet hadde andre eksamener å måtte ta hensyn til, så det ble dessverre ikke prioritert å utbedre disse punktene, men heller fokusere på dokumentasjon og rapportskriving.

6.9 Videre utvikling

6.9.1 Funksjonalitet og brukeropplevelse

I dette prosjektet har gruppen produsert en fungerende app som oppfyller kravene for MVP. Dersom prosjektet hadde vart over lengre tid, er det flere ting gruppen ville implementert. Det ble blant annet avdekket flere svakheter ved brukeropplevelsen (se kapittel 6.8.4 *Brukbarhetstesting*). I den nåværende versjonen av appen er det ikke skrevet kode for å lagre favorittstedene sine, og solkartet som ble produsert, akkasseres gjennom en nettleser, og ikke som et fragment i appen. Solkartet er statisk, og vil ikke tilpasses seg ulike årstider, og WCAG 2.1 burde oppfylles så godt som mulig. Alt dette er punkter det ville være naturlig å utbedre.

Deretter burde det gjennomføres nok en runde med brukbarhetstesting, der man rekrutterte målrettet for å få et bredere spekter av kompetanse og behov for tilrettelegging, der man inkluderte brukere som har behov for tastaturnavigasjon og skjermleser. Dette ville kunne øke bruksverdien for et stort spekter av brukere, uansett om de er avhengige av slik tilrettelegging, eller om de ønsker det basert på personlige preferanser.

Det ble også oppdaget en "bug" gruppen ikke klarte å finne ut av. Resultatene for luftkvalitet skal vises som tall og et fargekodet ikon. I emulatoren og enkelte mobiler fungerte dette som normalt, mens en annen mobil (Samsung, API-level 30) alltid presenterte ikonet i grått, til tross for at det var data tilgjengelig.

6.9.2 Teknisk gjeld

Underveis i prosjekt har gruppen etter beste evne forsøkt å følge formelle og uformelle standarder og god kodeskikk for å minimere teknisk gjeld. Gruppen har likevel blitt noen valg de mener kan føre til teknisk gjeld. Grunnet tidsbegrensningen i prosjektet, har blant annet testing av egne metoder blitt mindre prioritert, noe som er grunnen til lav prosentandel testedekning av metodene. Dette har ikke ført til problemer i prosjektet, men det kan være en potensiell kilde til teknisk gjeld. De manglende testene må enten bli skrevet senere, eller så må man fortsette å gjøre testene i emulatoren, som kan være tidkrevende. En eventuell videreutvikling bør ha større fokus på både enhets- og integrasjonstesting. Videreutvikling vil sannsynligvis føre til at nye funksjonaliteter blir introdusert, noe som øker sannsynligheten for kodekonflikter i applikasjonen. For å minimere disse konfliktene og samtidig minimere sjansen for fremtidige konflikter, må testingen utføres rutinemessig. Gruppen har diskutert at den optimale løsningen for testingen er at teamet har et felles mål for dekningsprosent, for eksempel 80%, og at hvert enkelt teammedlem oppfyller dette kavet før koden sammenslås med felles kode.



Gruppen har underveis i utviklingen diskutert flere alternative verktøy som kunne potensielt benyttes under prosjektet. Noen eksempler på slike verktøy er Dagger, som kan benyttes for å håndtere avhengigheter, og Jetpack Compose, som er et alternativ til XML. De har derimot ikke blitt tatt i bruk, da gruppen mente de oppdaget dette for sent, prosjektet var godt i gang. Dette anser gruppen til en viss grad som teknisk gjeld, da verktøyene er anerkjent og brukes i økende grad, noe som sannsynligvis vil gjøre at teknologien gruppen benyttet snart vil bli utdatert, og applikasjonen Soleklart må oppdateres. Det ble også diskutert om å sette seg inn i verktøyene kunne bli for tidkrevende for gruppen. Samtidig ville det også trolig blitt vanskeligere å finne ressurser på nettet, da disse er nyere teknologier som har færre kilder og brukseksempler tilgjengelig.

Applikasjonens ikke-funksjonelle krav har i liten grad fokusert på responstid på API-kall. Det har ikke blitt utført målinger som viser forventet responstid fra API-ene. Denne informasjonen kan tenkes å være nyttig både for brukeren og som de som har teknisk interesse eller ønsket til å videreutvikle applikasjonen. Den eventuelle API-responstidtestingen kan ha flere mulige variabler som for eksempel antall brukere som sender forespørslar, brukerens lokasjon, og styrke av nettverkstilkobling. En kompliserende faktor i responstidtesting av Soleklart, er det at applikasjonen bruker flere API-er samtidig, som til slutt produserer et sammensatt resultat. En helhetlig informasjon om responstid kan fås dersom den eventuelle testingen også undersøker variasjoner i responstid mellom de ulike API-ene. Slik informasjon vil kunne identifisere eventuelle flaskehalsar i systemet, noe som kan være en betydelig faktor i fremtidig forbedring og redesign av systemets arkitektur. Det finnes flere verktøy som kan benyttes til en enkel og automatisert API-responstidtesting, hvor de mest kjente verktøyene er f.eks. *Katalon*, *Postman* og *JMeter* (Hamilton, 2022).



7 Refleksjon

7.1 Felles situasjonsforståelse

Under en mob-koding-økt i sprint 3, oppdaget gruppen at det hadde oppstått en divergens mellom flere av gruppas situasjonsforståelse i forhold til prosjektets visjon. Dette ble oppdaget som følge av at gruppas designere mellom seg diskuterte veien videre for appen, mens resten av gruppa arbeidet med sammenslåing av kode. Det viste seg at designerne hadde en forståelse av at den endelige løsning skulle involvere funksjonaliteter som hadde blitt diskutert i tidligere fase, mens andre gruppemedlemmer hadde en forståelse av at disse funksjonalitetene ikke lenger var en del av gruppens målbilde.

Da gruppen oppdaget denne misforståelsen knyttet til løsningens visjon, ble det med en gang tydelig at det var nødvendig å håndtere denne misforståelsen før man kunne fortsette med videre arbeid på prosjektet. Diskusjonen som fulgte som følge av oppdagelsen av denne misforståelsen var en hel essensiell diskusjon. En felles forståelse av løsningens visjon ville være avgjørende for arbeidet videre og for å sikre at gruppen jobbet mot det samme målet. Gruppen ble derfor nødt til å skifte fokus den siste delen av arbeidsøkten, hvor målet ble å identifisere hvor misforståelsen hadde oppstått, samt å bli enige om hva som skulle være visjonen for applikasjonen.

Denne misforståelsen løste gruppa på en meget god måte. Gruppen diskuterte de motstridende forståelsene i felleskap, samt hvorfor disse hadde oppstått. Deretter ble gruppen enige om hvilken visjon det var som skulle legges til grunn, slik at alle medlemmene igjen ble samkjørte. Etter denne hendelsen lærte gruppen at det ville være hensiktsmessig om man i større grad diskuterte det kontinuerlige arbeidet i forhold til det store bildet, og veien videre under sprintstarter fremover, for å unngå at en liknende misforståelse skulle oppstå igjen.

7.2 Sykdom og covid-19

I starten av prosjektet ble gruppemedlemmene kjent med at det var et punkt i rapportkravene som oppfordret til refleksjon rundt påvirkningen covid-19 har hatt på prosjektet. Dette opplevdes som overraskende, siden samfunnet på det tidspunktet i stor grad var gjenåpnet og de aller fleste restriksjonene fjernet. Det ble derfor sendt en melding for å spørre om denne delen kunne utelates fra rapporten, da gruppen antok at det ikke ville være noen utfordringer for prosjektet knyttet til sykdom. Det var altså ingen plan for hva som skulles skje om noen i gruppa ble syke over lengre tid.

Noen uker inn i prosjektet viste det seg at gruppen tok feil. I løpet av prosjektet har 4/6 av gruppas medlemmer vært hjemmeværende som følge av egen sykdom, samt egne barns sykdom, fra bl.a. covid-19 (*Figur 42*). Dette førte til at enkelte av gruppens medlemmer ikke hadde mulighet til å møte fysisk i flere perioder. Gruppens løsning på dette ble å la det hjemmeværende gruppemedlemmet delta digitalt, mens resten av gruppen gjennomførte møtet fysisk. Dette viste seg å ikke være en optimal løsning, da det medlemmet som deltok digitalt, naturlig nok ikke fikk deltatt i samtalen i like stor grad som resten av gruppa som var fysisk til stede. Medlemmet som deltok digitalt, ble ofte sittende å lytte til varierende lydkvalitet fra det fysiske møtet. Likevel var en hybrid løsning bedre enn at den smittede



møtte fysisk før vedkommende følte seg komfortabel med dette, for å unngå å smitte andre i gruppen.

Om det hadde vært planlagt for sykdom på forhånd kunne man muligens hatt klare løsninger som var bedre tilpasset en situasjon hvor bare deler av gruppa kan møtes fysisk, men denne læringen fikk man heller ta i det situasjonen oppsto. I etterkant har gruppen lært at det er hensiktsmessig å ha et ekstra fokus på å være kort og konsis under hybride møter, da det gir frihet til de som deltar digitalt slik at de heller kan bruke tiden sin på andre ting enn å lytte til en samtale som de ikke helt får deltatt i.

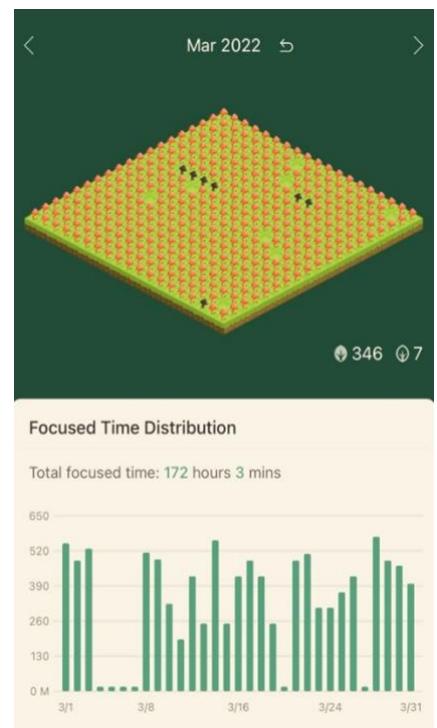


Figur 42: Positiv hurtigtest for covid-19.

7.3 Tidsbruk

Parallelt med arbeidet for dette prosjektet, hadde alle medlemmer i prosjektgruppen andre emner ved universitetet, samt andre forpliktelser som jobb og familie. Dette gjorde at gruppen var nødt til å håndtere utfordringer knyttet til tidsbruk og planlegging tidlig. Siden alle hadde forskjellige timeplaner, og forskjellige frister for innleveringer og eksamener, ble det tidlig klart at det var nødvendig å sørge for en god oversikt over medlemmernes egne frister i kombinasjon med de fristene som var felles for prosjektet i IN2000. Denne utfordringen ble løste ved å etablere en tidslinje som tok utgangspunkt i siste frist for innlevering av prosjektet. Deretter jobbet gruppen seg bakover i tid, og plottet inn frister, eksamensdatoer samt ferier for alle gruppens medlemmer, slik at teamet hadde en god forståelse av tiden som sto til disposisjon i løpet av semesteret. Dette gjorde at gruppen fikk en god situasjonsforståelse over hvilke perioder man ville være nødt til å arbeide mer intenst med prosjektet i IN2000. På denne måten ble det også enklere å håndtere utfordringer knyttet til tidsbruk og ressurser som oppsto underveis, da det fantes en oversikt over gruppens timeplaner allerede i starten av prosjektet (se vedlegg 9.4 *Prosjektkalender*).

Gruppen opplevde at noen perioder i prosjektet var mer travle enn de andre. Begynnelsen av prosjektet, særlig mars måneden var intens fordi mye tid måtte brukes på å bli kjent med visjonen, verktøy og måten teammedlemmer jobber på. Et av gruppemedlemmene har logget timene som gruppen har brukt på prosjektet ved bruk av Forest App (*Figur 43*). Her vises det at det ble totalt brukt ca. 172 timer i mars på arbeidet med Soleklart. Gruppen har i etterkant innsett at det har blitt lagt inn mye arbeid i dette prosjektet og at resultatene viser det.



Figur 43: Logg over timer brukt på prosjektet i mars 2022 som vist i applikasjonen Forest App. Hver fullført arbeidsøkt resulterer i en rød sopp, mens avbrutt arbeidsøkt blir til en brun



7.4 Kommunikasjons- og arbeidskanaler

Under første fase av prosjektet ble det definert hvilke verktøy, samt kommunikasjons- og arbeidskanaler som skulle benyttes, samt på hvilken måte de forskjellige verktøyene skulle brukes (se kapittel 6.2 *Verktøy*). Dette fungerte i hovedsak godt, og gruppa evnet å holde seg til de føringene som ble satt i forhold til hvilke verktøy som skulle brukes, og på hvilken måte de skulle brukes. Likevel opplevde gruppen at det kunne være utfordrende å få med seg alt av beskjeder, spørsmål og produkter ettersom man kom lengere ut i prosjektet. Dette var trolig en konsekvens av at flere personer arbeider med forskjellige ting parallelt, og at det i slike situasjoner kan være utfordrende å holde seg oppdatert på andres arbeid når man selv jobber med andre ting samtidig, noe som er en relativt naturlig utfordring i en samarbeidssituasjon for et større prosjekt. For å møte denne utfordringen ble det tydelig for gruppen at man måtte være flinke til å informere hverandre om eget arbeid under sprint-review, slik at alle gruppens medlemmer hadde en god situasjonsforståelse, også for prosessene og problemstillingene til resten av gruppa.

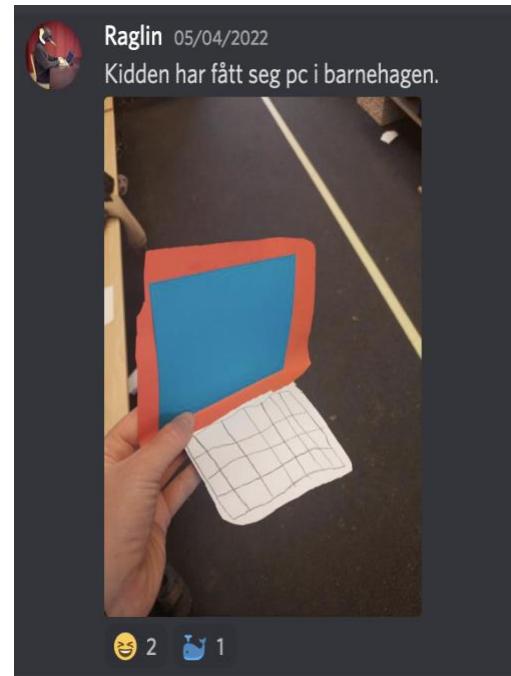
Selv om kommunikasjonskanaler var primært ment for arbeidet knyttet til prosjektet, ble disse allikevel benyttet til å ha litt moro. "Off-topic"-kanalen på Discord ble mye brukt til å dele morsomme bilder og hendelser og ble et godt plattform for litt uformell lagbygging. Figur 44 viser en av mange morsomme ting som ble delt i kanalen.

7.5 Å være interaksjonsdesigner i IN2000

Et aspekt ved prosjektarbeidet som var et gjentagende tema i flere samtaler mellom gruppemedlemmer, var hvordan man skulle implementere tverrfaglighet under prosjektets varighet. Denne problemstillingen var særlig til stede hos de gruppemedlemmene som hadde bakgrunn fra design.

Under introduksjonen av emnet IN2000 ble det opplyst om at prosjektarbeidet skulle være en mulighet til å øve seg på godt tverrfaglig samarbeid ved hjelp av smidige arbeidsmetoder. I starten av prosjektet var nettopp dette med å tilpasse graden av tverrfaglighet en aktuell problemstilling for å kunne oppnå en god forståelse av rollene innad i gruppen, samt tilnærmingen til utviklingsprosessen.

Som designstudent lærer man egne tilnærninger til utvikling, som ofte inkluderer å tilrettelegge for brukernes behov og krav i så stor grad som mulig. I flere av designemnene man tar før IN2000 legges det særlig vekt på å gi brukerne reell medvirkningskraft, og gjensidig læring, samtidig som man i høy grad prioriterer brukernes ønsker. Dette fører gjerne til prototyper hvor de tekniske funksjonalitetene er abstrahert over og prototypes med "Wizard of Oz"-teknikk, som da kan imøtekommne flere krav hos brukerne. Dette er en god



Figur 44: Et eksempel på at kommunikasjonskanaler gjennom prosjektet ble brukt både til moro og arbeid.



tilnærming om man ønsker å bruke mye tid på å vurdere bredden av ulike funksjonaliteter, men det resulterer sjeldnere i et faktisk fungerende sluttprodukt.

I emnet IN2000 er det å vektlegge brukernes behov og krav også en viktig del av prosjektarbeidet, men her skal man også etterstrebe og følge smidig utviklingsmetodikk, samtidig som man implementerer objektorienterte prinsipper og god kodeskikk, slik at man til slutt ender med et godt designet og fungerende sluttprodukt. Det er med andre ord et større emne, der man må prioritere mellom flere hensyn enn i de emnene designstudentene har vært gjennom til nå.

Som følge av de ekstra hensynene som var nødvendige under utviklingsprosjektet, opplevde designstudentene noen uker inn i prosjektet at det var nødvendig å reinstille sine forventninger i forhold til hvordan tilnærmingen til utviklingen måtte foregå. Dette ble også klart etter å ha hørt med gruppelæreren i emnet, da de på spørsmål om balansen mellom designfaget og IN2000 svarte at man gjerne kunne tenke på IN2000 som en forlengelse av emnet IN1030 *Systemer, krav og konsekvenser*, og at omfanget av designarbeidet er en del mindre enn det designstudentene er vant til.

7.6 Avsluttende refleksjon

Gjennom dette prosjektet har gruppen fått læringsutbytte som dekker ulike fagområder. Teamet har diskutert at arbeidet med Soleklart har gitt oss en mulighet til å oppleve hvordan en utvikling av et større prosjekt utføres. Det innebærer at teamet fikk prøvd ulike smidige metoder og har opplevd fordeler og ulemper med de arbeidsmetodene. Teknisk kunnskap og ferdigheter er også en stor del av læringsutbyttet. Dette prosjektet ga mulighet til å både arbeide med konkrete teknologier, og lære å finne relevant informasjon.

Et av områdene med det største læringsutbyttet er personlig kompetanse. Gruppemedlemmer har i etterkant av prosjektet diskutert at en av de viktigste tingene gruppen har fått fra prosjektet er trening i grupperarbeit. Dette innebærer både å kunne utvikle sin egen selvinnsikt, empati, se gruppens mål fremfor egne mål og ikke minst øve på å takle mindre og større problemer i felleskap.



8 Kilder og referanser

Aalberg, G. (2020). Optimizing Traffic. Hentet 11.04.2022 fra <https://github.com/metno/weatherapi-docs/blob/master/doc/OptimizingTraffic.md>

Almås, S., Lindsjørn, Y., Stray, V. (2021). *A case study of teamwork and project success in a comprehensive capstone course.*

Android Developers. (2022). *Geocoder*. Hentet 10.04.2022 fra <https://developer.android.com/reference/android/location/Geocoder>

Bratteteig, T. (2021). *Design for, med og av brukere*. Universitetsforlaget.

Colblindor (Producer). (2022, 19.05.2022). Coblis – Color blind simulator. Hentet 19.05.2022 fra <https://www.color-blindness.com/coblis-color-blindness-simulator/>

Dagger. (2022). Dagger. Hentet 19.05.2022 fra <https://dagger.dev/dev-guide/>

Dziedzic, K. (2021). What is Jetpack Compose and why do we need it? Mood Up Team. Hentet 19.05.2022 fra <https://moodup.team/blog/what-is-jetpack-compose-and-why-do-we-need-it/>

Hamilton, T. (2022). What is Response Time Testing? How to Measure for API, Tools. Hentet 03.05.2022 fra <https://www.guru99.com/response-time-testing.html>

Kouraklis, J. (2016). MVVM as Design Pattern.

Lazar, J., Feng, J.H., Hoccheiser, H., (2017) *Research Methods in Human-Computer Interaction* (2nd ed.). Morgan Kaufmann.

Lindsjørn, Y. (2021, 24.mars). *Modellering av krav*. Hentet 14.05.2022 fra https://www-int.uio.no/studier/emner/matnat/ifi/IN1030/v21/forelesninger-modul-b/in1030_2021.03.24_use-case-modellering.pdf

Material Design (Producer). (2022, 13.05.2022). Color Tool. Hentet 14.05.2022 fra <https://material.io/resources/color/#!/?view.left=0&view.right=0>

NILU. (2022). Om tjenesten. Hentet 10.04.2022 fra <https://luftkvalitet.nilu.no/om>

Norges Blindeforbund. Fargeblind/ fargesvak. Hentet 15.04.2022 fra <https://www.blindeforbundet.no/oyehelse-og-synshemninger/fargeblindhet>

Seierstad, I. A. S. (2020). *Locationforecast: data sources*. Hentet 11.04.2022 fra <https://github.com/metno/weatherapi-docs/blob/master/doc/locationforecast/>

Sjøberg, D. (2021, 10. mars). *Introduksjon til systemutvikling*. Intro_systemutvikling. Hentet 15.05.2022 fra <https://www->



int.uio.no/studier/emner/matnat/ifi/IN1030/v21/forelesninger-modulb/in1030_2021.03.10_intro_systemutvikling.pdf

Sommerville, I. (2015). *Software Engineering*. Harlow, England: Addison-Wesley. ISBN: 9781292096131

Sommerville, I. (2021). *Engineering software products: An introduction to modern software engineering*.

Stray, V. (2022, 02 09). *Smidige praksiser, autonome team og koordinering i stor-skala smidig*. 09_02_22_foiler. Hentet 11.04.2022 fra https://www.uio.no/studier/emner/matnat/ifi/IN2000/v22/forelesninger/09_02_22_foiler.pdf

Tveter, F. T. (2021). Sunrise. Hentet 11.04.2022 fra <https://docs.api.met.no/doc/sunrise/sunrise>

Wengaard, R. (2022). Forenklet Solkart. Hentet 19.05.2022 fra https://rawengaard.github.io/Forenklet_solkart/#9/59.9832/10.6828



9 Vedlegg

Dette vedlegget er et supplerende dokument som skal leses sammen med hovedrapporten. Innholdet i vedlegget er referert i de ulike kapitlene i rapporten. Vedlegget består av følgende delkapitler:

- 9.1 WCAG 2.1
- 9.2 Mappeoversikt
- 9.3 Rapport fra Sprint Review 2
- 9.4 Prosjektkalender
- 9.5 Brukerhistorier
- 9.6 Diskusjon om design patterns
- 9.7 Visuell utforming
- 9.8 Spørreundersøkelse knyttet til visuelle ikoner



9.1 WCAG 2.1

Det er totalt 78 suksesskriterier.

Gruppen har ikke gått i dybden på kriteriene, men forsøkt å få et generelt overblikk.

Oppfyller	43
Usikker/ delvis	13
Ikke	28

Percievable	Operable
<ul style="list-style-type: none"> ● Guideline 1.1: Text Alternatives <ul style="list-style-type: none"> ○ 1.1.1 ● Guideline 1.2: Time-based Media: <ul style="list-style-type: none"> ○ 1.2.1 ○ 1.2.2 ○ 1.2.3 ○ 1.2.4 ○ 1.2.5 ○ 1.2.6 ○ 1.2.7 ○ 1.2.8. ○ 1.2.9 ● Guideline 1.3: Adaptable <ul style="list-style-type: none"> ○ 1.3.1 ○ 1.3.2 ○ 1.3.3 ○ 1.3.4 ○ 1.3.5 ○ 1.3.6 ● Guideline 1.4: Distinguishable <ul style="list-style-type: none"> ○ 1.4.1 ○ 1.4.2 ○ 1.4.3 ○ 1.4.4 ○ 1.4.5 ○ 1.4.6 ○ 1.4.7 ○ 1.4.8 ○ 1.4.9 ○ 1.4.10 ○ 1.4.11 ○ 1.4.12 ○ 1.4.13 	<ul style="list-style-type: none"> ● Guideline 2.1: Keyboard Accessible <ul style="list-style-type: none"> ○ 2.1.1 ○ 2.1.2 ○ 2.1.3 ○ 2.1.4 ● Guideline 2.2 Enough Time <ul style="list-style-type: none"> ○ 2.2.1 ○ 2.2.2 ○ 2.2.3 ○ 2.2.4 ○ 2.2.5 ○ 2.2.6 ● Guideline 2.3 Seizures and Physical Reactions <ul style="list-style-type: none"> ○ 2.3.1 ○ 2.3.2 ○ 2.3.3 ● Guideline 2.4 Navigable <ul style="list-style-type: none"> ○ 2.4.1 ○ 2.4.2 ○ 2.4.3 ○ 2.4.4 ○ 2.4.5 ○ 2.4.6 ○ 2.4.7 ○ 2.4.8 ○ 2.4.9 ○ 2.4.10 ● Guideline 2.5 Input Modalities <ul style="list-style-type: none"> ○ 2.5.1 ○ 2.5.2 ○ 2.5.3 ○ 2.5.4 ○ 2.5.5 ○ 2.5.6

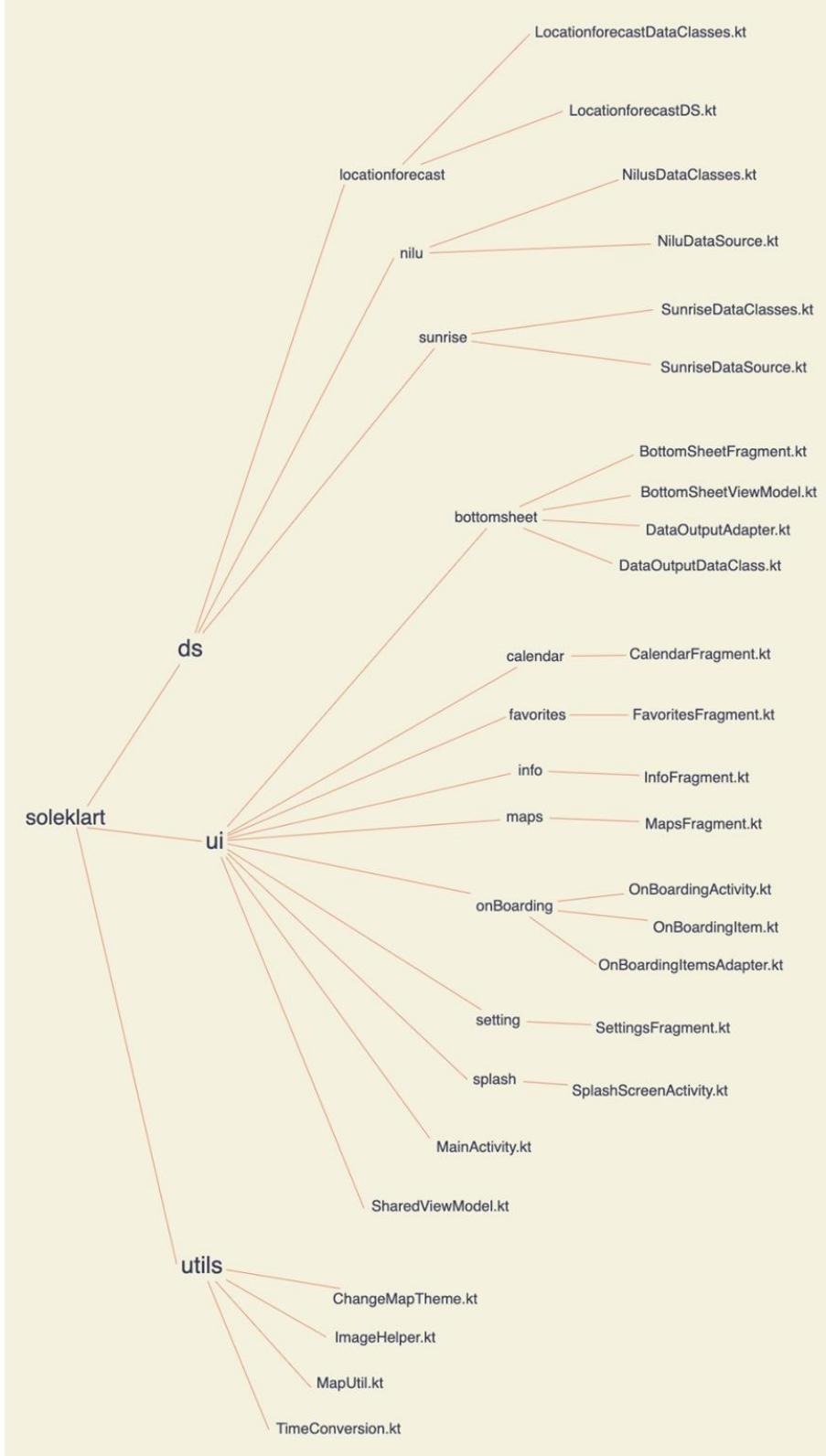


Understandable	Robust
<ul style="list-style-type: none"> ● Guideline 3.1 : Readable <ul style="list-style-type: none"> ○ 3.1.1 ○ 3.1.2 ○ 3.1.3 ○ 3.1.4 ○ 3.1.5 ○ 3.1.6 ● Guideline 3.2 Predictable <ul style="list-style-type: none"> ○ 3.2.1 ○ 3.2.2 ○ 3.2.3 ○ 3.2.4 ○ 3.2.5 ● Guideline 3.3: Input Assistance <ul style="list-style-type: none"> ○ 3.3.1 ○ 3.3.2 ○ 3.3.3 ○ 3.3.4 ○ 3.3.5 ○ 3.3.6 	<ul style="list-style-type: none"> ● Guideline 4.1: Compatible <ul style="list-style-type: none"> ○ 4.1.1 ○ 4.1.2 ○ 4.1.3 <p data-bbox="652 518 837 552">Conformance</p> <ul style="list-style-type: none"> ● Guideline 5.1: Interpreting Normative Requirements ● Guideline 5.2: Conformance Requirements <ul style="list-style-type: none"> ○ 5.2.1 ○ 5.2.2 ○ 5.2.3 ○ 5.2.4 ○ 5.2.5 ● Guideline 5.3: Conformance Claims (Optional) <ul style="list-style-type: none"> ○ 5.3.1 ○ 5.3.2 ● Guideline 5.4: Statement of Partial Conformance - Third Party Content ● Guideline 5.5: Statement of Partial Conformance - Language



9.2 Mappeoversikt

Figur 45 viser tre diagram over alle filene og mappene i applikasjonen Soleklart.



Figur 45: Mappe- og filoversikt for applikasjonen Soleklart.



9.3 Rapport fra Sprint Review 2 nettskjema

Rapport fra skjema besvart 01.03.2022

Rapport fra «Sprint review 2»

Innhentede svar pr. 5. april 2022 10:17

- Leverte svar: **6**
- Påbegynte svar: **0**
- Antall invitasjoner sendt: **0**

Med fritekstsvar

Hva gikk veldig bra i denne sprinten? *

- vi fikk gjort masse, hadde mange ideer
- Team spirit, motivasjon, selvdrevet team
- Gruppen fikk utført målene ved sprinten som var å lage datasource for de ulike klassene, viewmodel og startet på mainactivity. I tillegg til at oppgavefordelingen under planleggingsmøte gikk ganske bra hvor gruppen var flinke til å tilordne arbeidsoppgaver til alle i teamet.
- Samarbeidet når vi først jobber sammen, altså etter møter og slike. Oppmøte og innsats på møter.
- Alle hadde jobba hardt med APIene og kjente til disse godt, dette gjorde at de som ikke hadde jobbet med et API kunne få gode forklaringer av de som hadde tatt ansvar for egne oppgaver, bra jobba folkens!
- Fikk gjort mye, det er bra😊

Hva gikk ok? *

- planlegging, drøfting av ideer
- Hybride møter
- Syntes alt gikk ganske bra, kommer ikke på noe som ansees som "ok".
- Kunne vært litt effektive på møter, men ser ikke på dette som en så dårlig ting. Da man har det gøy og hyggelig, selvom det tar litt lengre tid.
- At folk jobber på tvers av fagene sine, men idk, har vi det som mål egentlig? For hvis ikke, så har det jo funka kjempebra
- Merging av kode

Hva kan forbedres? *

- kommunikasjon, for lange møter, mye digresjoner
- .
- Kanskje så burde gruppen fokusere på å vite hvordan rapporten skal fylles inn. Men dette er noe gruppen kan diskutere om. Ellers var det ikke noe som kunne forbedres siden gruppen var flinke med å beskrive arbeidsoppgaver og fikk utført dem.
- Kan ha flere oppgaver hver uke.
- kanskje vi burde kjøre en liten fordeling av hvem skriver hva i rapporten?
- Kode struktur?

Hvor smidige er vi? *

0 - ikke smidig, 10 - veldig smidig

- 5
- 7
- 10
- 10
- 9
- 10

Hvor fornøyd er du med:

Svar fordelt på antall

	ikke fornøyd	kunne vært bedre	heilt ok	veldig fornøyd
Kommunikasjonskanaler *	0	0	1	5
Faglige diskusjoner *	0	1	2	3
Arbeidsmengde *	0	0	3	3
Foreløpig produkt *	0	0	3	3
Antall møter *	0	0	3	4
Fordeling av arbeidsoppgaver *	0	1	3	2
Lengden på møtene *	1	1	3	2
Strukturen på møtene *	1	2	3	1
Kommunikasjon utenfor møtene *	0	1	2	3
Veiledere *	0	1	3	2
Verktøyene *	0	0	2	4

**Svar fordelt på prosent**

	ikke fornøyd	kunne vært bedre	heilt ok	veldig fornøyd
Kommunikasjonskanaler *	0 %	0 %	16,7 %	83,3 %
Faglige diskusjoner *	0 %	16,7 %	33,3 %	50 %
Arbeidsmengde *	0 %	0 %	50 %	50 %
Foreløpig produkt *	0 %	0 %	50 %	50 %
Antall møter *	0 %	0 %	50 %	66,7 %
Fordeling av arbeidsoppgaver *	0 %	16,7 %	50 %	33,3 %
Lengden på møtene *	16,7 %	16,7 %	50 %	33,3 %
Strukturen på møtene *	16,7 %	33,3 %	50 %	16,7 %
Kommunikasjon utenfor møtene *	0 %	16,7 %	33,3 %	50 %
Veiledere *	0 %	16,7 %	50 %	33,3 %
Verktøyene *	0 %	0 %	33,3 %	66,7 %

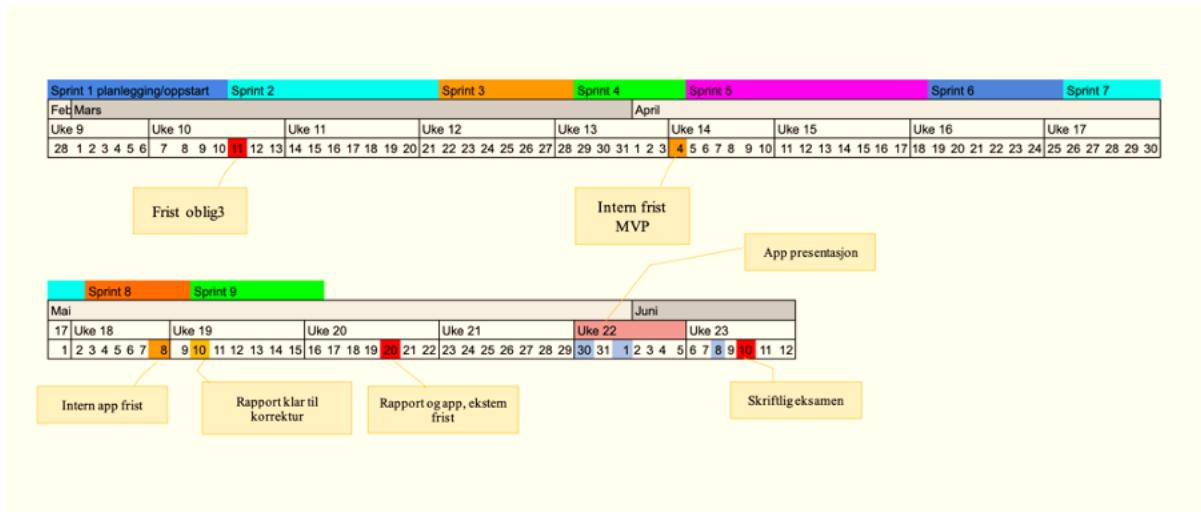
Noe mer på hjertet?

- Er glad i banan :)
- :)
- sikkert nice å se om ikke vi kan ta en liten runde med emil og emil, ikke at vi trenger å ha noe spess å snakke med de om, men bare for å sjekke at vi fortsatt er på skiva.



9.4 Prosjektkalender

Gruppen lagde i starten av prosjektet en tidslinje for prosjektet.



Figur 46: Kalender med oversikt over alle sprints som ble utført gjennom prosjektet. Kalenderen viser også både interne og eksterne frister for prosjektet samt frister og viktige hendelser i andre fag som medlemmene tar.



9.5 Brukerhistorier

Brukerhistorier for å beskrive brukerens behov. Skrevet i format :

Som <bruker> ønsker jeg å <gjøre noe> (for å <oppnå et mål>).

User Stories 1.0

- *Som bruker ønsker jeg å se korttidsvarsel for der jeg er, så jeg vet hvordan sikten er de neste timene.*
- *Som bruker ønsker jeg å se værvarsler for et selvvalgt sted, for å kunne vurdere om jeg skal dra et annet sted enn der jeg er nå.*
- *Som bruker ønsker jeg å kunne lagre favorittstedene mine, så jeg slipper å søke opp stedene hver gang.*
- *Som bruker ønsker jeg å kunne se værvarsler frem i tid, så jeg kan planlegge når jeg skal på tur.*
- *Som bruker ønsker jeg å vite når sol og måne går opp/ned, for å vite om det er mørkt nok til å se stjerner.*
- *Som bruker ønsker jeg å kunne vite hvilke stjernebilder jeg kan se, for å lære noe nytt.*
- *Som bruker ønsker jeg å kunne velge nattmodus for å ikke ødelegge nattsynet.*

User Stories 2.0

- *Som bruker ønsker jeg å få en enkel innføring i hvordan jeg bruker appen, slik at jeg unngår frustrasjon og forvirring under senere bruk.*
- *Som bruker ønsker jeg å kunne velge dato for henting av informasjon via en kalender, slik at jeg enklere kan se at jeg velger riktig dato.*
- *Som bruker ønsker jeg å vite når sola går opp og ned, for å kunne se på dette.*
- *Som bruker ønsker jeg opplysninger om skydekke, luftkvalitet, temperatur, vind og nedbør når det er solnedgang og soloppgang, for å kunne oppleve disse under komfortable forhold.*
- *Som bruker ønsker jeg å kunne velge en lokasjon for informasjon om siktforhold og vær, ved å søke i fritekst.*
- *Som bruker ønsker jeg å kunne velge en lokasjon for informasjon om siktforhold og vær, ved å markere stedet på et kart.*
- *Som bruker ønsker jeg å kunne se min nåværende lokasjon i kartet til appen.*



9.6 Diskusjon om design patterns

Dette vedlegget presenterer en tabell som oppsummerer en diskusjon gruppen utførte angående valg av *design pattern*. Tabellen viser noen av de viktigste design patterns som ble diskutert, samt en kort beskrivelse av faktorer som bestemmer hvor aktuelle et bestemt pattern er.

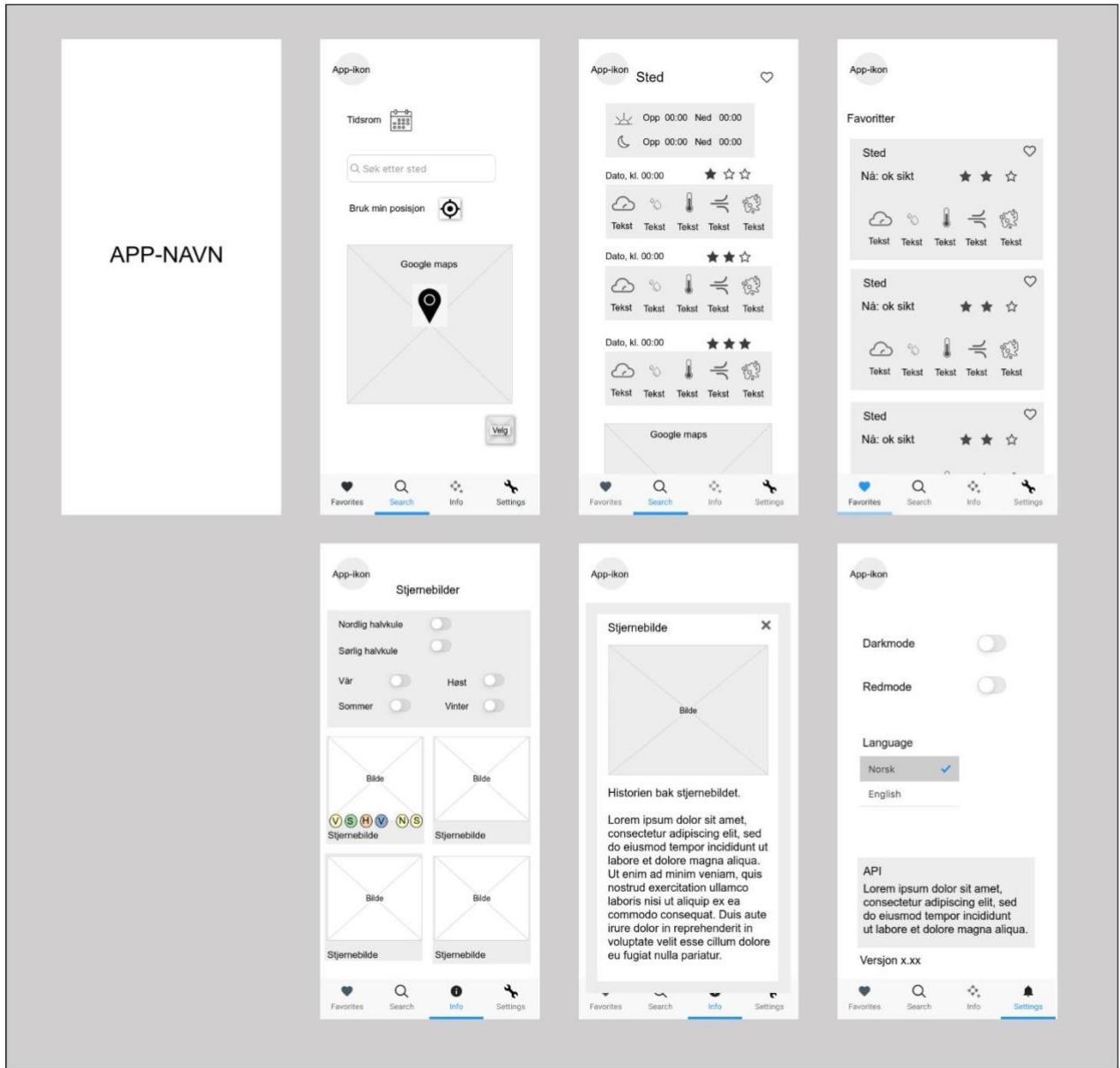
Tabell 9: oppsummering av gruppens diskusjon angående valg av arkitektur design pattern.

Pattern	Analyse
Layered	Tilbyr det som er ønskelig - å dele opp funksjonalitet i lag, hvor den øverste laget kan være UI, deretter presentasjon, data ressurser og API-kall. Det virker tungvint hvis det er ønskelig å hente noe direkte fra et annet lag som er ikke rett under. Det er ikke alltid mulig å separere all funksjonalitet uten overlapping.
Pipe-filter	Funksjonelle og ikke funksjonelle krav for Soleklart ser ikke ut til å kreve slik prosessering. Det finnes generelt få "rå" brukerinputs som prosesseres. Det finnes heller ikke stort behov for sikkerhet siden appen har ikke registrerte brukere og ingen personlig informasjon skal bli overført mellom komponentene.
Repository architecture	Denne ble diskutert til å ikke være aktuell for denne applikasjonen på grunn av to aspekter. "Repository" i denne applikasjonen er på en måte API-er som brukes og som behøver ikke å lagres midlertidig. I tillegg til det gjøres det få utregninger i applikasjonen - det meste av data hentet er brukt og presentert direkte.
MVVM	Hovedfordelen er muligheten til å endre data uavhengig av andre deler. Samtidig finnes det en ulempe som er at denne arkitekturen er tidkrevende å implementere - de ulike komponentene må bestemmes og implementeres. Gruppen har allerede god kjennskap til denne arkitekturen.
Event bus pattern	Prinsippene med publisher og subscribers godt kjent for oss. En ulempe gruppen opplever er at subscribers som "hører" etter events i bakgrunnen vil gjøre appen tregere og dermed mindre attraktiv for brukeren, noe som er en viktig faktor i dette tilfellet.



9.7 Visuell utforming

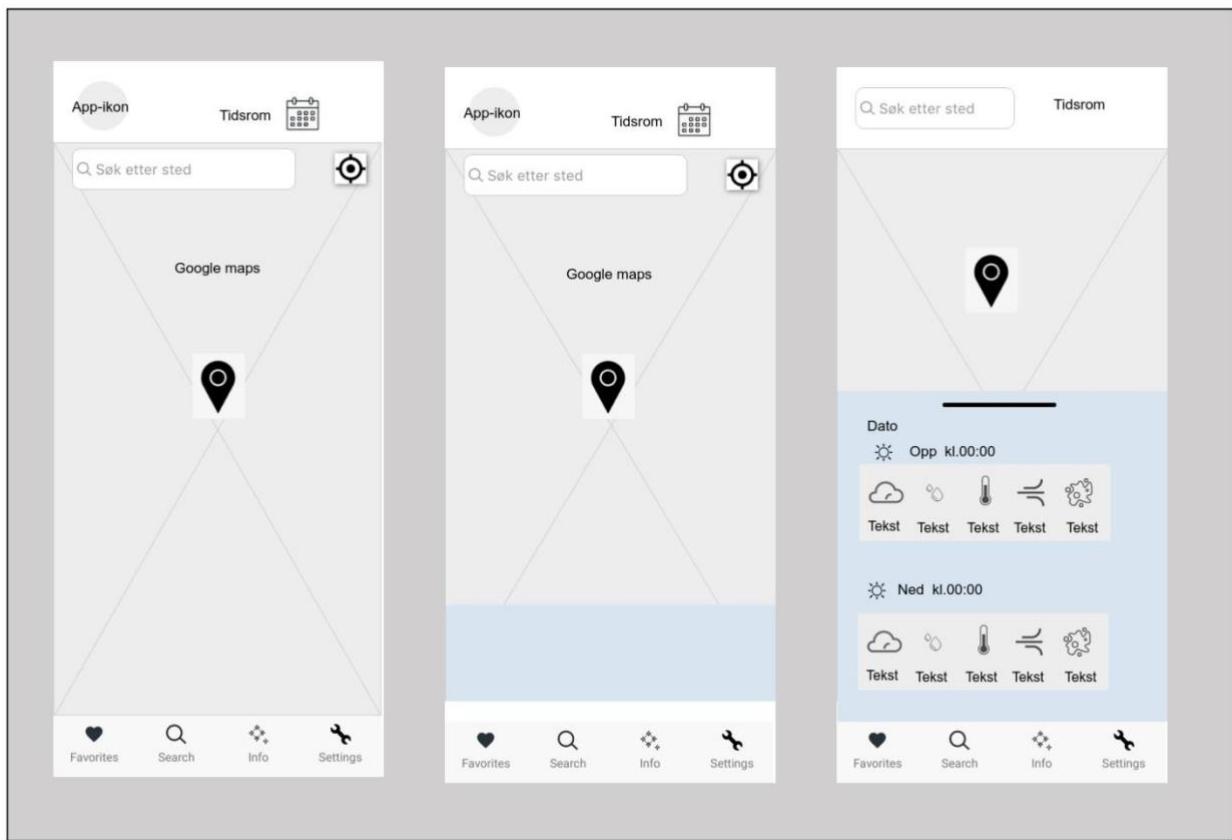
Første utkast til UI (*Figur 47*) var kun ment for intern bruk.



Figur 47: Grove skisser med plassholdere.



Andre utkast (*Figur 48*) tok bare for seg en enkelt funksjon.



Figur 48: Resultatet kan vises ved å trekke opp et vindu.



Tredje utkast (Figur 49) utforsket hvordan *Favoritter* kunne implementeres.

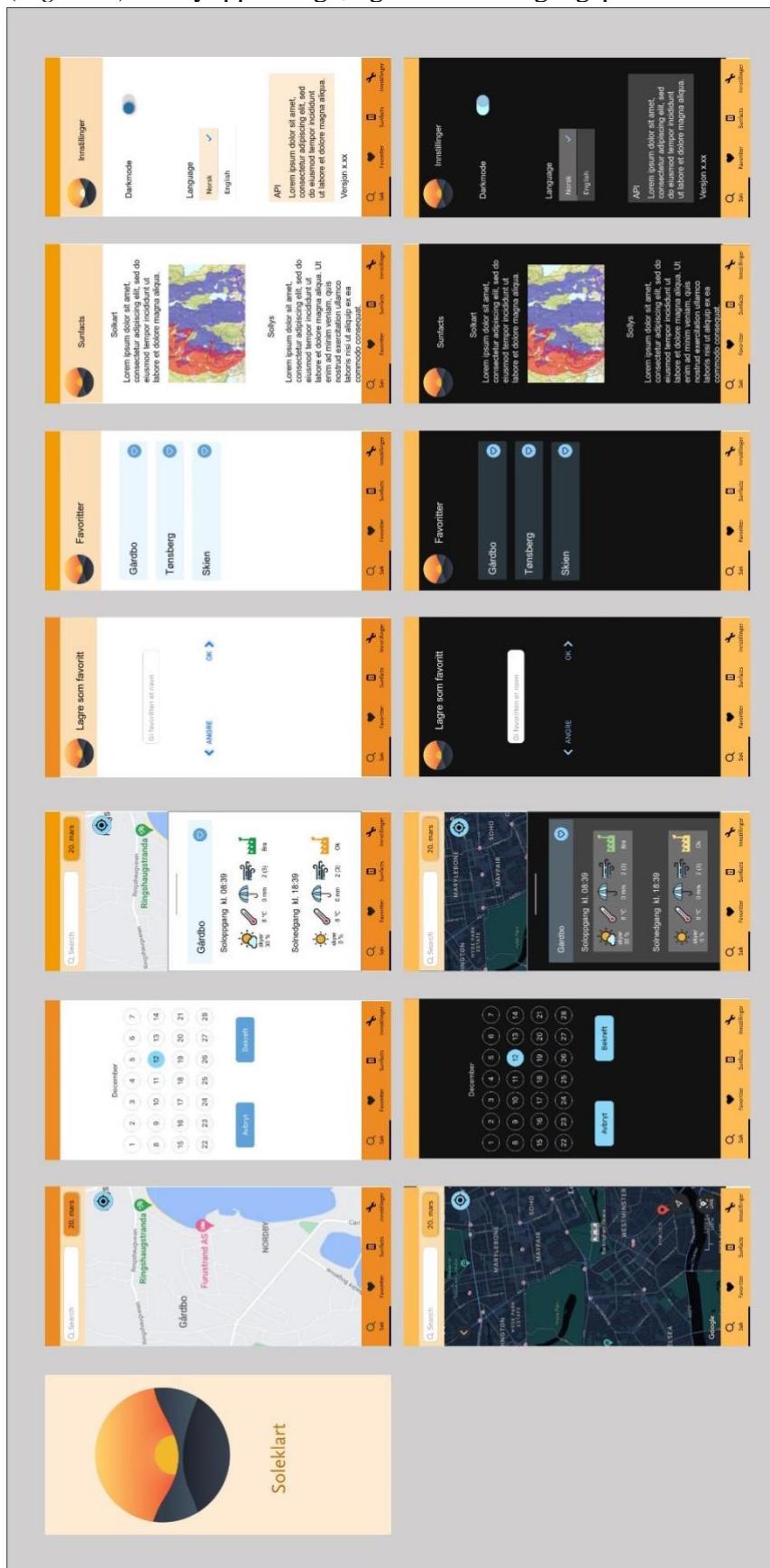
The figure displays six screenshots of a mobile application interface, illustrating various ways to present favorite locations:

- Row 1:** Shows a grid of three locations: Gårdbo, Tønsberg, and Skien. Each location has a red circular icon with a question mark.
- Row 2:** Shows the same three locations, but each has a detailed weather forecast below it. For example, Gårdbo's forecast includes "Soloppgang kl. 08:39" and "Solnedgang kl. 18:39".
- Row 3:** Shows a map view with a search bar at the top. Below the map, a detailed weather forecast is shown for Tønsberg, including "Soloppgang kl. 08:39" and "Solnedgang kl. 18:39".

Figur 49: Favoritter kan presenteres på ulike måter.



Fjerde utkast (*Figur 50*) er høyoppløselige, og ment som utgangspunkt for kodingen.

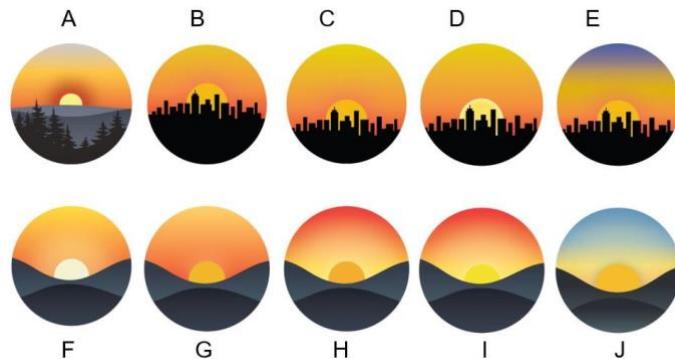


Figur 50: detaljerte skisser med farger og innhold.



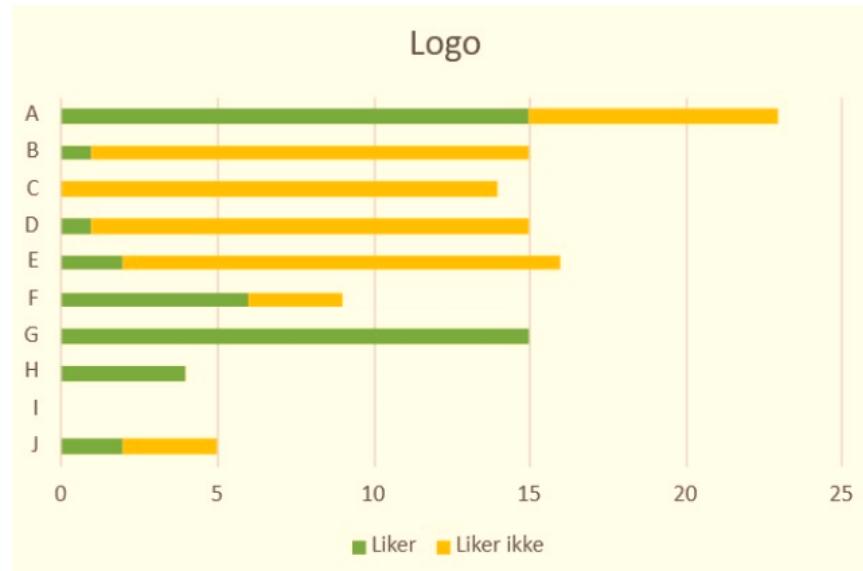
9.8 Spørreundersøkelse knyttet til visuelle ikoner

Gruppen ønsket å finne ut hvilke preferanser brukerne har til visuell utforming. Dermed ble det laget en rekke utkast til app-ikon (Figur 51), og samlet flere ulike stiler på ikoner for værvarsle. Disse ble så skrevet ut på A4-ark, som ble vist til studenter ved IFI. For å unngå bias, fikk de ikke vite hva andre respondenter hadde svart, med unntak av dem som satt sammen i grupper og hørte hva sidemannen sa. Respondentene ble bedt om å velge en eller flere favoritter, og en eller flere de absolutt ikke likte.



Figur 51: utkast til logo.

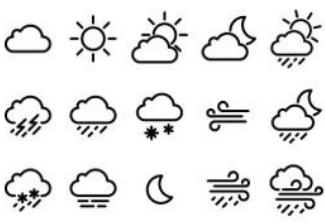
Mange utdype og også valgene sine. Det var generell enighet om at en urban silhuett ble for mørk og dominert, og ikke ville fungere så godt når ikonet er lite (Figur 52). Likevel var det en stor andel som likte A best, til tross for at den også er mer detaljert enn F-J. Mens enkelte likte at himmelen var blå, og mer realistisk, var det andre som mente dette så feil ut. Det var også noen som mente at J så ut som en soloppgang, og F en solnedgang, mens andre mente det stikk motsatte. A og G fikk like mange positive stemmer, men av disse var det kun G som ikke fikk negative stemmer.



Figur 52: Stolpediagram som viser respondentenes logo-preferanser.

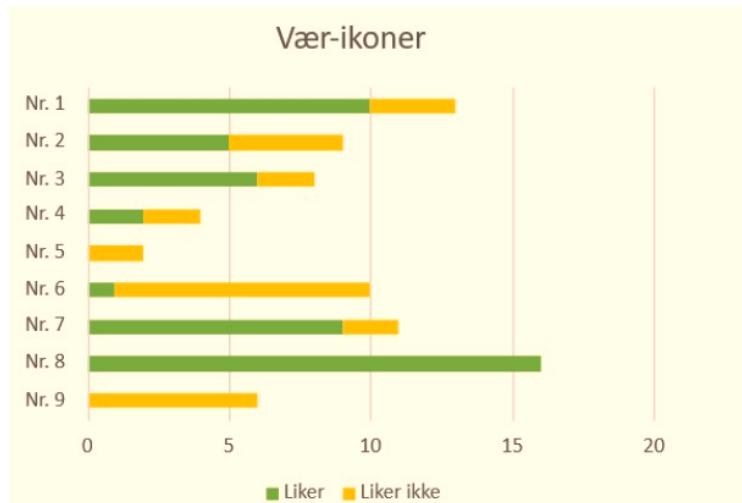


Vær-ikonene var i ulike stiler, med to eksempler fra hver stil, i tillegg til nr. 9 (*Figur 53*). Dette gjorde det mulig å undersøke om folk foretrakk flerfargede, ensfargede eller linjesymboler, samt stil innad i disse gruppene.

1	2	3
		
4	5	6
		
7	8	9
		

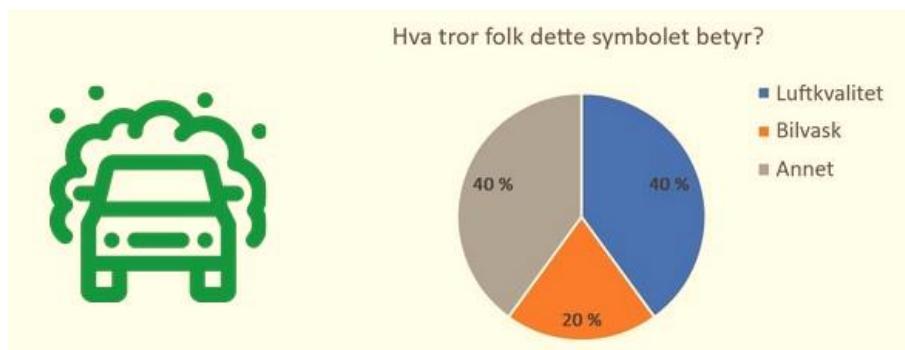
Figur 53: Et variert utvalg av ikoner. Ikonene fra www.vecteezy.com.

Resultatene viser en klar favoritt (*Figur 54*). Flere begrunnet valget med at kombinasjonen av farger og markerte linjer gjorde symbolene lettere å “lese” og skille fra hverandre.



Figur 54: Stolpediagram som viser respondentenes ikon-preferanser.

På spørsmål om hva dette symbolet betyr, svarte 40% “luftkvalitet/ svevestøv/ forurensning”, mens majoriteten mente det hadde en annen betydning (Figur 55).



Figur 55: Ikon for svevestøv, og et kakediagram som viser hva respondentene tenkte ikonet betyr.

Dersom man lager en ordsky av svarene, viser det at “Luftkvalitet/ svevestøv/ forurensning” (her forenklet til “luftkvalitet”) helt klart var budskapet det var størst enighet om (Figur 56).



Figur 56: Ordsky av hva respondentene mente ikonet betyr.



Flere av respondentene fikk et tilleggsspørsmål om hvordan man kan symbolisere "luftkvalitet", men det var bare noen få som klarte å komme med et forslag: "eksosrør med eksos", "fabrikkpipe" og "brun sky". Respondentene sa også at selv disse symbolene kunne føre til misforståelser, og at de ikke anså disse som optimale alternativer. Til tross for at mange tolket ikonet som "luftkvalitet" uttrykte respondentene mye usikkerhet. Derfor valgte gruppen å bytte ut ikonet med en fabrikk.

I denne undersøkelsen fikk respondentene se symbolet ute av kontekst, uten tekst, og i fargen grønn. Det var flere som fortalte at de ble forvirret av fargen. Grønn brukes ofte for å symbolisere at noe er positivt eller miljøvennlig, mens ikonet viser en bil og noe de gjettet på var eksos eller forurensning, som oppleves negativt. Symbol og farge formidler ulike følelser, noe som resulterte i en generell forvirring. Det ville være interessant å gjøre en undersøkelse til, der symbolet ble vist i sin rette kontekst, og med undertekst, for å se om flere respondenter tolket symbolet slik gruppen ønsket. Den samme disharmonien mellom farge og ikon vil også oppstå med fabrikk-ikonet, men fabrikk-ikonet vil neppe feiltolkes i like stor grad som bil-ikonet.