

DroneZone

Prosjektoppgave IN2000

Diana Berg
Karin Margrethe Hvalby
Jonas Finborud Nyman
Marte Lunde Kvam
Hinda Yasin Muse
Felicia Ediriweera Norén

1



Team-nummer: 18
Case: Åpent case - i lufta
Emenavn: Software Engineering med prosjektarbeid
Emnekode: IN2000
Veiledere: Jesper Dahl Norgård, Gyda Elisa Sæter

¹dianaber, karinmhv, jonasfn, martelkv, hindaym, feliciae

Innhold

1	Introduksjon	4
2	Presentasjon	4
2.1	Beskrivelse av teamet og medlemmene	4
2.2	Beskrivelse av løsningen	6
2.3	Hvordan kjøre appen	11
3	Brukerdokumentasjon	12
3.1	Målgruppe	12
3.2	Aktører	12
3.3	Plattform og aksessering av applikasjonen	12
3.4	Applikasjonens funksjonalitet	13
4	Kravspesifikasjon og modellering	13
4.1	De viktigste funksjonelle kravene	13
4.2	Use case diagram	13
4.3	Tekstlige beskrivelser	14
4.4	Klassediagram og sekvensdiagram av de viktigste use-casene	15
4.5	Objektorienterte prinsipper	17
4.5.1	Design patterns - MVVM	17
4.5.2	Lav kobling og høy kohesjon	18
5	Produktdokumentasjon	19
5.1	Viktigste kvalitetsegenskapene ved appen	20
5.1.1	Funksjonalitet	20
5.1.2	Brukskvalitet	20
5.1.3	Bugs	21
5.2	API	22
5.2.1	API fra Meteorologisk Institutt	22
5.2.2	Andre APIer	23
5.3	Struktur	23
5.4	Jetpack Compose	24
5.5	Ktor	24
6	Prosessdokumentasjon	25
6.1	Verktøy	25
6.2	Valg av smidig metode	25
6.3	Oppstartsfasen	27

6.3.1	Teamet	27
6.3.2	Bli-kjent-kveld	29
6.3.3	Valg av case	29
6.4	Sprint 1	31
6.4.1	Prototyping	33
6.4.2	Funksjonalitetene i MVPen	34
6.4.3	Utfordringer i sprint 1	35
6.4.4	Retrospekt sprint 1	36
6.5	Sprint 2	37
6.5.1	Utfordringer i sprint 2	40
6.5.2	Retrospekt sprint 2	41
6.5.3	Evaluering av kravspec	42
6.6	Sprint 3	42
6.6.1	Utfordringer i sprint 3	46
6.6.2	Retrospekt sprint 3	47
6.7	Sprint 4	47
6.7.1	Utfordringer i sprint 4	49
6.7.2	Retrospekt sprint 4	49
6.8	Sprint 5	49
6.8.1	Utfordringer i sprint 5	50
6.8.2	Retrospekt sprint 5	50
6.9	GDPR	51
6.10	Universell utforming	51
6.11	Testing	51
6.11.1	Enhetstester	51
6.11.2	Brukertester	54
7	Refleksjon	55
7.1	Planlegging	55
7.2	Mirobrettet	56
7.3	Ustrukturerte og uformelle møter	57
7.4	Møtene underveis i prosjektet	57
7.5	Ekspertintervju	58
7.6	Designprosessen	58
7.7	Smidig metode	59
7.8	Rollefordelingen	61
7.9	Oppfylte vi kravene fra kravspesifikasjonen?	61
7.10	Oppsummering	61

8 Vedlegg

Vedlegg	64
8.1 Intervjuguide	64
8.2 Intervjuplan	65
8.3 Analyse fra brukerintervju	66
8.4 Samtykkeskjema	68
8.5 Klassediagram	70
8.6 Oppdatert kravspesifikasjon	71
8.7 Opprinnelig kravspesifikasjon	73
8.8 Teamavtale	76
8.9 Brukertest	78
8.10 Universell utforming	79

1 Introduksjon

Droneflyving er en hobby som har dukket opp i de nyere tidene som et resultat av innovativ teknologi, og er for en flyveentusiast deres svar på en radiostyrt bil. I motsetning til radiostyrte biler er droner mer fleksible ettersom de kan flys forbi sperringer og har en lang rekkevidde fra flyveren. Det er derfor viktig å regulere hvor dronene kan fly på grunn av sikkerhet. I tillegg er droner ofte utstyrt med kameraer, noe som skaper strenge krav til hvor de kan fly og filme. Tidligere var det lite reglement på plass for å sørge for at droneflyvingen ble gjennomført på en måte som var trygg for både flyveren og omgivelsene, men i de senere årene har Luftfartstilsynet publisert klare regler og lover som gjelder for alle som driver med hobbyflyving av drone. Politiet har i flere anledninger oppgitt at ulovlig dronekjøring kan gi bøter på opptil 12 000 kr (UAS-Norway, 2020). I den anledningen har gruppen vår ønsket å utvikle et verktøy for droneflyvere som forsøker å samle reglene og lovene i en interaktiv applikasjon som kan assistere droneflyvere med å vite om de kan fly drone på en gitt lokasjon, samt opplyse brukeren om regler og lover som de kanskje ikke visste om. Vår app er en løsning på case 4 (*Åpent case - i luften*) i emnet IN2000.

De store utviklerne av droner, som f.eks. *DJI* og *Parrot*, har vanligvis en assisterende applikasjon som brukeren kan bruke for å styre og fly dronen deres, og vi ønsker ikke å forsøke å erstatte eller implementere liknende funksjonalitet som disse appene har. Disse appene er rettet mer mot selve flyveaktiviteten, og vi ønsker derfor istedenfor å utvikle en applikasjon som hjelper en bruker med å planlegge deres droneflyving slik at en bruker kan være trygg på at de kan ta frem dronen sin og fly. Applikasjonen ønsker derfor å være et tilleggsverktøy rettet mot droneflyvere som de kan benytte seg av på forhånd av selve flyveaktiviteten.

2 Presentasjon

Dette kapittelet tar for seg hvem vi er, hva appen vår gjør, samt hvordan man kjører appen.

2.1 Beskrivelse av teamet og medlemmene

Teamet består av 6 studenter, hvor fire teammedlemmer (Jonas, Hinda, Felicia og Diana) tilhører studieprogrammet *Programmering og systemarkitektur*, mens de resterende (Karin og Marte) tilhører *Design, bruk og interaksjon*.

Figur 1 viser en presentasjon av teammedlemmene.

Jonas

Litt om meg:

- 24 år
- byttet fra design til prosa etter første året
- studert musikkproduksjon, programering og spilldesign tidligere
- liker å trenne, være med venner og game



Styrker:

koding, planlegging, research, skriving, systematisk, litt ordensfreak

Svakheter:

fremføringer, konflikt, designarbeid

Hva kan jeg bidra med:

koding, planlegging, holde tidsfrister, evaluere arbeid, researche, rapportskriving

Karin

Litt om meg:

- 21 år
- drevet mye med musikk tidligere, og litt teater og fotografi
- går på design nå
- på fritiden liker jeg å spille spill, trenne og cosplay



Styrker:

- kreativitet, planlegging

Svakheter:

- koding, konflikt, prokastinering

Hva kan jeg bidra med:

- designarbeid, rapportskriving, planlegging, evaluering

Felicia

Litt om meg:

- 20 år
- Elsker te (og iskaffe)
- Linje: Prosa
- Synes koding er veldig gøy :)
- Fra Stockholm



Styrker:

Koding, få ting til å se pent ut, kan jobbe hardt

Svakheter:

Litt perfeksjonist, hvordan legge opp smidig/sprints osv, norsk noen ganger

Hva jeg kan bidra med:

Koding, UI, rapportskriving, presentasjonen

Hinda

Litt om meg:

- 20
- går prosa



Styrker:

kommer for sent, men jobber med det, liker ikke å snakke foran en crowd

Hva jeg kan bidra med:

koding, god stemning, kreativitet

Diana

Litt om meg :

- 23 år
- Går prosa
- Bakgrunn fra fysikk og astronomi
- Driver med styrketrening og maler
- Veldig glad i katter
- Liker å spille Fortnite



Styrker

- Jobber godt under press, God til å snakke for meg og presentere, Koding, Analytisk

Svakheter

- Blir lett distraheret, Snakker for mye om andre ting, Prokrastinerer, Databaser

Hva kan jeg bidra med

- Progging, God stemning :D, Github-erfaring, Fremføring, Ærlige meninger

Marte

Litt om meg :

- 24 år
- Går design
- Har bachelor i fysikk og astronomi
- Holder på med turning og teater
- Har hund
- Leder for studentkontakter i Tekna



Styrker

- Jeg er løsningsorientert, Positiv, Analytisk

Svakheter

- Jeg trenger ofte betenkningstid, Lett distraheret, Ikke noen perfeksjonist

Hva kan jeg bidra med

- Møtevirksomhet, Koding, Design - Brukermedvirkning, Åpenhet

Figur 1: Presentasjonskort av gruppemedlemmene. Her skrev hver gruppemedlem litt om seg selv, sine styrker og svakheter, og hva de kunne bidra med i gruppen.

2.2 Beskrivelse av løsningen

Vår løsning på case 4 er appen *DroneZone*. Dette er en app som viser brukeren en oversikt over vær, regler, forbudte soner og en sjekkliste for hvorvidt brukeren kan fly drone i det valgte området. Appen består av en hovedside med kart, samt en begrunnelsesside, en værsiden og en regelside.

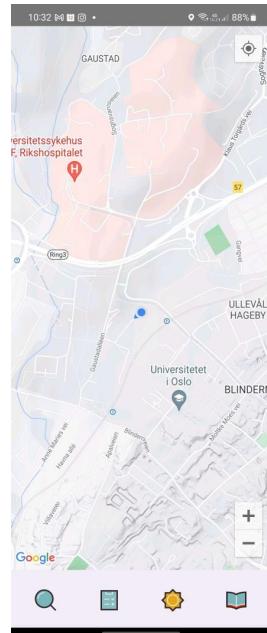
Startside

Appen starter med en *Splash Screen*, som er en oppstartsside med appens logo. Denne vises i figur 2a. Det første som vises til brukeren etter dette er hovedsiden, se figur 2b. Hovedsiden inneholder et kart som tar utgangspunkt i brukerens posisjon (vist som en blå sirkel på kartet). Brukeren kan zoome og sveipe over skjermen for å utforske andre steder på kartet.

Nederst på skjermen under kartet ligger navigasjonsbaren. Fra venstre til høyre ligger knapper for søkerbaren, begrunnelsessiden, værsiden og regelsiden.



(a) Oppstartsside. Siden viser appens logoen.

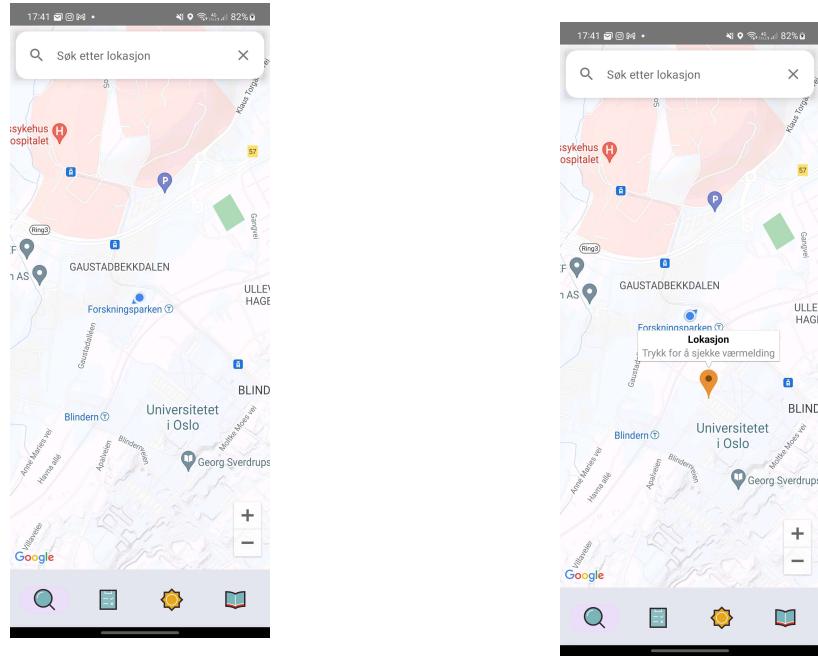


(b) Appens hovedside. Siden inneholder kart og navigasjonsbar.

Søkefunksjon og flyplassoner

Hvis brukeren vil bevege kartet til en spesifikk lokasjon, kan de benytte seg av appens søkerfunksjon ved å trykke på søkerikonet nederst til venstre i navigasjonsbaren. Da dukker søkerbaren opp øverst på skjermen, se figur 3a. Når man skriver inn en adresse i søkerbaren, kommer det opp forslag. Eksempel på dette vises i figur 4a. For å fjerne teksten man har

skrevet inn kan man trykke på kryss-tegnet i søkerbaren. For å søke trykker man på et av forslagene, og kartet vil da bevege seg til den gitte lokasjonen slik som i figur 4b.



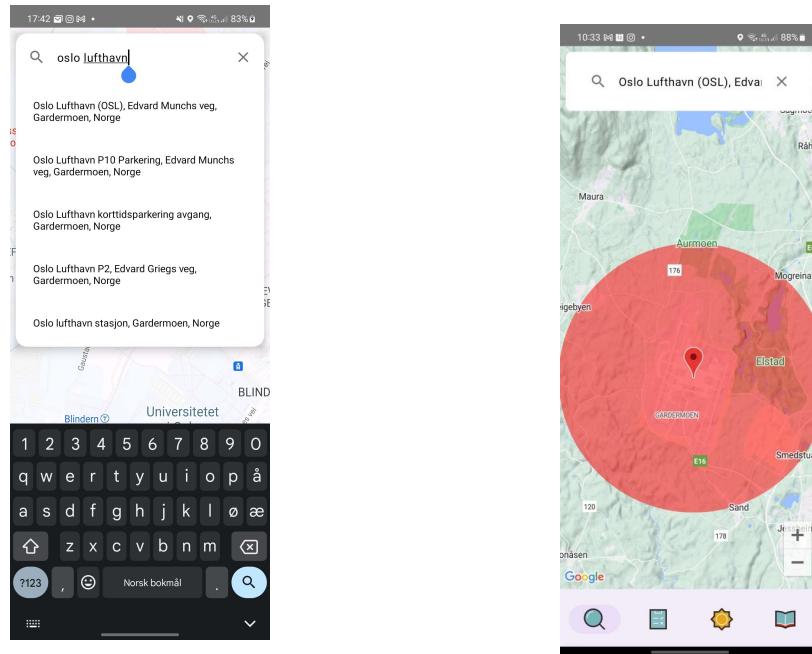
(a) Dersom man trykker på søkerikonet nedest i venstre hjørnent dukker det opp en søkerbar hvor man kan søke på ønsket lokasjon.

(b) Ved å trykke på et sted på kartet vil det dukke opp en oransje pin, som vil angi din nye lokasjon.

Figur 3

Figur 4b viser appen etter at brukeren har søkt på Oslo Lufthavn. For å fjerne søkerbaren kan man trykke på søkerikonet i navigasjonsbaren igjen. På kartet er det tegnet opp markører med røde soner rundt hver flyplass i Norge, for å vise hvor man ikke får fly drone (figur 5a). Hvis markøren til en flyplass trykkes på, vises navnet på flyplassen. Figur 5b viser hvordan dette ser ut for Oslo Lufthavn. Radiusen for enhver rød sone er 5 km, da det er påkrevd av norsk lov å holde minst denne avstanden til en flyplass hvis man skal fly drone (Luftfartstilsynet, 2022).

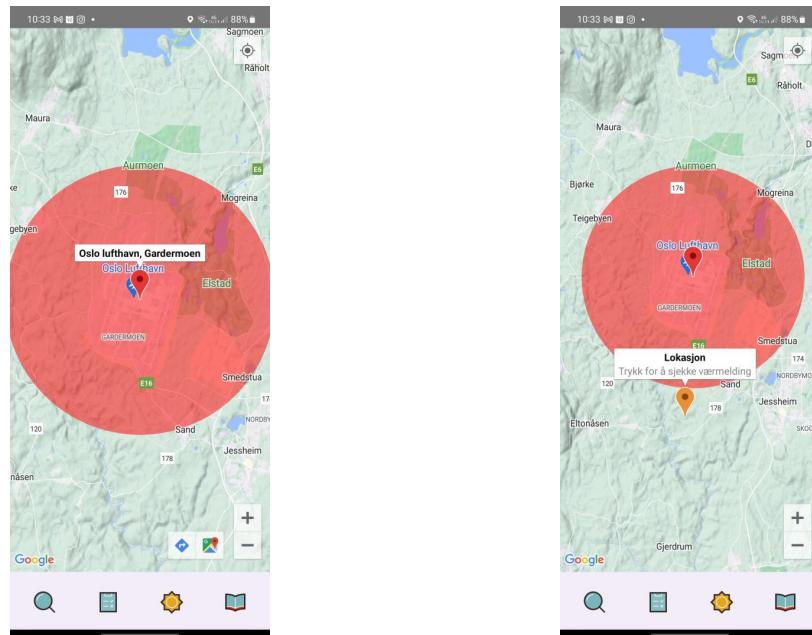
Ved å trykke på kartet kan brukeren velge en lokasjon. Da settes det en oransje markør, som vist i figur 3b. Over markøren vises et vindu med teksten *Trykk for å sjekke værmelding*, og hvis dette trykkes på vil værsiden dukke opp (figur 7a). Værsiden beskrives i kommende avsnitt.



(a) I søkefeltet kan man søke på ønsket lokasjon. Det vil også komme opp forslag til lokasjon.

(b) Kartet flytter seg til lokasjonen man søker på. I dette tilfellet Gardemoen.

Figur 4



(a) Ved å trykke på pinnen får man opp navnet på flyplassen, over pinnen.

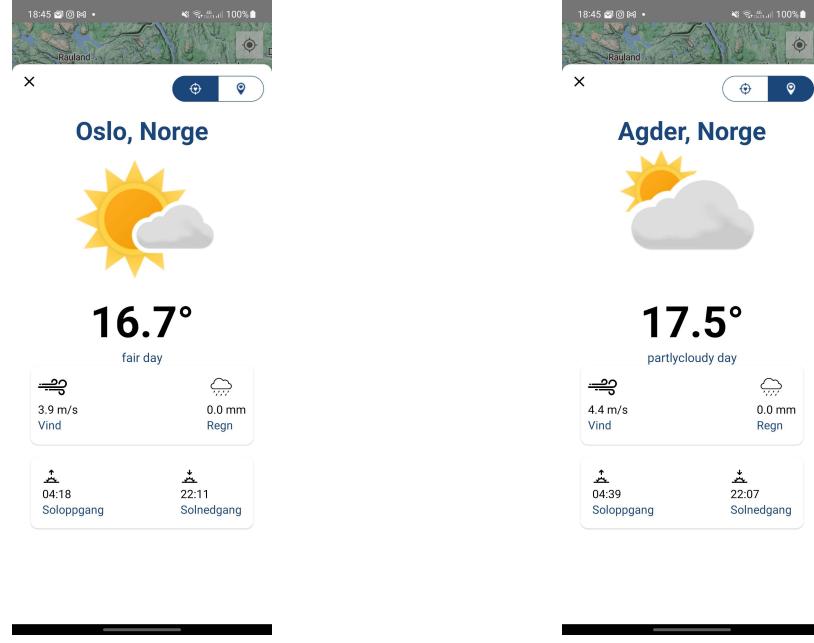
(b) Ved å trykke på kartet, kan man sette ny lokasjon. Denne vises som en oransje pin.

Figur 5

Værsiden

Værsiden (figur 6a og 6b) viser temperatur, vindhastighet og forventet mengde regn for den

commende timen. Den viser også tidspunktene for soloppgang og solnedgang. Dataen som vises er enten for markør-lokasjonen, eller brukerens lokasjon. På samme måte som med begrunnelsessiden kan brukeren enkelt veksle mellom å vise værdata for sin egne lokasjon, eller en markør-lokasjon dersom brukeren har plassert en markør.

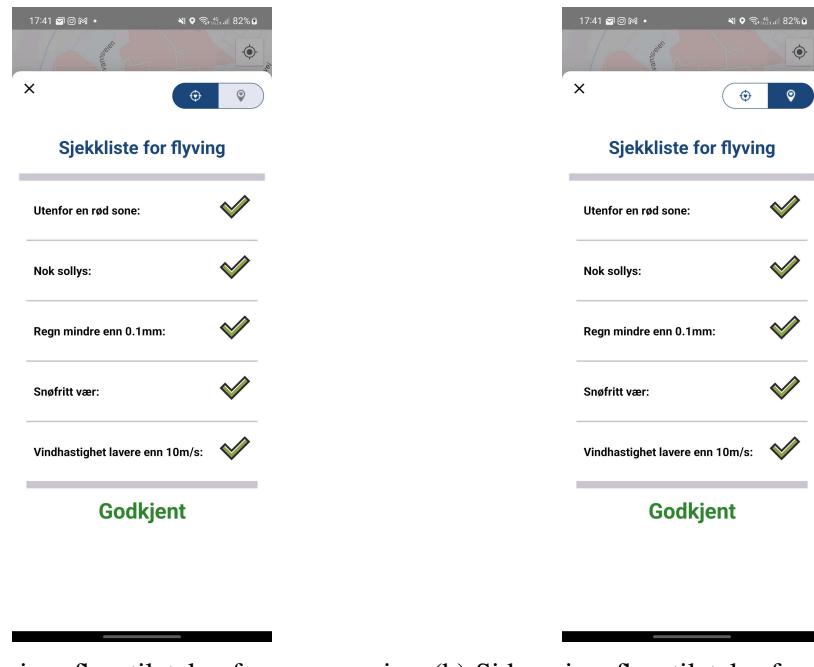


(a) Siden viser været for egen posisjon. (b) Siden viser været for pinnet lokasjon.

Figur 6

Flygetilatelse

I begrunnelsessiden (figur 7a) kan brukeren sjekke om visse krav for å fly drone er oppfylt. Dette inkluderer å være utenfor en forbudt sone, å ha nok sollys, minimalt med vind, samt fravær av regn og snø. Dersom minst ett krav ikke er oppfylt, vil brukeren få beskjed om at vedkommende ikke kan fly drone i det valgte området. Dersom alle krav er oppfylt får brukeren beskjed om at flyvingen er godkjent. Denne siden aksesseres ved å trykke på sjekkliste-ikonet i navigasjonsbaren (andre ikonet fra venstre). Hvis brukeren har markert en lokasjon, kan brukeren både sjekke for sin egen lokasjon, og for en markør-lokasjon. Dette kan enkelt veksles mellom ved å trykke på de to knappene i øverste høyre hjørne.

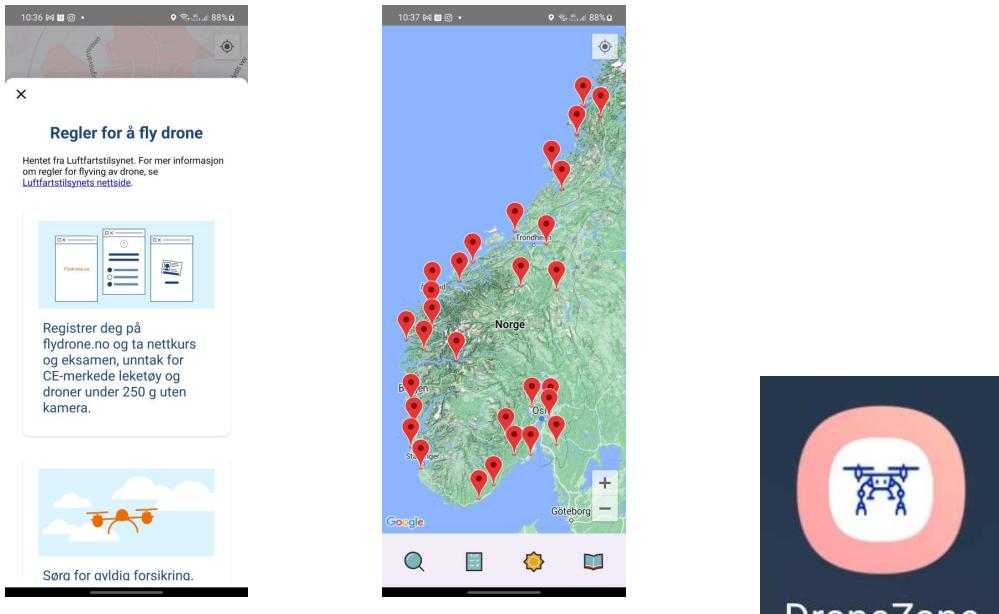


(a) Siden viser flygetilatelse for egen posisjon.
(b) Siden viser flygetilatelse for pinnet posisjon.

Figur 7

Regelsiden

Ved å trykke på ikonet lengst til høyre i navigasjonsbaren (se figur 8a), kan brukeren få opp regelsiden. Regelsiden presenterer generelle regler for å fly drone i Norge. Denne siden inneholder tekst og bilder direkte hentet fra Luftfartstilsynets nettside (Luftfartstilsynet, 2023a), som det også finnes en lenke til lengst opp i siden. Figur 8a viser hvordan regelsiden ser ut når brukeren først åpner den. Brukeren kan så bla i siden for å se alle regler.



- (a) Regelsiden. Denne inneholder reglene for dronekjøring. Reglene og illustrasjonene er hentet fra Luftfartstilsynet (Luftfartstilsynet, 2023a).
- (b) Hovedsiden når kartet er zoomet ut til å vise deler av Norge. Her kan vi se store deler av flyplassene som er plassert på kartet.
- (c) Figuren viser hvordan appikonet ser ut når den er lastet ned på mobiltelefonen.

Figur 8

2.3 Hvordan kjøre appen

For å kjøre appen kreves det en datamaskin med *Android Studio* installert. I tillegg kreves tilgang til enten en fysisk *Android*-mobil eller en virtuell mobil (emulator), og denne må være tilkoblet nettet. For å optimalt kunne kjøre appen skal API-nivået til mobilen være 33 eller høyere. Mobilen må også ha *Google Play* installert, ellers vil ikke appen fungere. Et eksempel på en god *Android*-modell for å kjøre appen er *Pixel 3a* med API-nivå 33.

Etter dette kan stegene nedenfor følges for å kjøre appen:

1. Last ned zip-filen fra *Devilry*
2. Pakk ut zip-filen (extract)
3. Åpne *Android Studio* og trykk på *Open*
4. Naviger til mappen hvor prosjektmappen som ble pakket ut ligger, og velg mappen
5. Når prosjektet er lastet opp i *Android Studio*, trykk på *Build* øverst i menyen og trykk så på *Clean Project*.
6. Når endringene har blitt lastet, trykk på *Build* igjen og så på *Rebuild Project*

- Start opp emulatoren eller tilkoblet telefon, og trykk på den grønne *play*-knappen. Appen vil da kjøre. Når appen spør om tilgang til lokasjon, må bruker gi tilgang til nøyaktig lokasjon for å kunne bruke appens funksjoner

3 Brukerdokumentasjon

Dette kapittelet tar for seg appens målgruppe, aktører, plattform og aksessering av applikasjonen, samt applikasjonens funksjonalitet.

3.1 Målgruppe

Målgruppen for appen er nordmenn som flyr drone, og ønsker å fly dronen sin på et trygt og lovlig sted i Norge. Figur 9 viser en persona (Sommerville, 2021, s.64) som er laget for å skape en bedre forståelse av målgruppen.

3.2 Aktører

Målgruppen beskrevet over vil være appens primæraktør da de initierer bruksmønstre som oppfyller appens mål. Sekundæraktører som muliggjør disse bruksmønstrene vil være MET sine APIer som har gitt tilgang til informasjon om vær og sol ved hjelp av *Locationforecast* og *Sunrise*. Luftfartstilsynet og Datatilsynet er også sekundæraktører ettersom deres nettsider er brukt for å hente ut aktuelle droneregler for appen.

3.3 Plattform og aksessering av applikasjonen

Siden applikasjonen er utviklet i *Android Studio*, vil den kun være tilgjengelig på *Android* sine mobiltelefoner. I tillegg må brukeren være tilkoblet nettet mens appen er i bruk, for å kunne laste inn kartet over brukerens lokasjon og for å kunne laste inn informasjon om vær og sol som trengs for å vurdere flyvetillatelsen. Applikasjonen forutsetter også at brukeren gir tillatelse til aksessering av den nåværende posisjonen.

For å ikke overbelaste MET sine servere, er det et krav for prosjektet at appen må bruke proxy serveren som er tilegnet UiO sine studenter. Alle kall til MET API-ene gjøres dermed



Figur 9: Persona for målgruppen.

til denne og ikke direkte til MET. Hvis proxyen eventuelt blir tatt ned, vil applikasjonen ikke lengre kunne laste inn vær- og soldata, og derfor ikke fungere som den skal.

3.4 Applikasjonens funksjonalitet

Applikasjonens hovedfunksjonalitet:

Flyvetillatelse: Sjekke om det er trygt å sende opp en drone på en gitt lokasjon, i forhold til vær, sollys og regler knyttet til flyplasser.

Regelside: Brukeren skal enkelt ha tilgang til reglene for droneflyving.

Søkefunksjon: Brukeren skal kunne søke opp en posisjon og sette plasseringen sin der.

Værmelding: Brukeren skal kunne sjekke været i en gitt lokasjon.

Kart: Brukeren skal kunne navigere seg rundt på kartet. Hen skal se hvor hen kan og ikke kan fly basert på røde soner.

Begrunnelse: Når brukeren trykker på begrunnelseskappen skal brukeren få opp en begrunnelse for hvorfor hen kan/ikke kan fly drone i den gitte lokasjonen.

4 Kravspesifikasjon og modellering

I dette kapittelet finnes ulike beskrivelser av kravspesifikasjonen vår; både i form av bruk av diagrammene *use case diagram*, *sekvensdiagram* og *klassediagram*, men også ved bruk av tekstlige beskrivelser og brukerhistorie. Det finnes også informasjon om objektorienterte prinsipper og hvordan disse er ivaretatt i appen.

4.1 De viktigste funksjonelle kravene

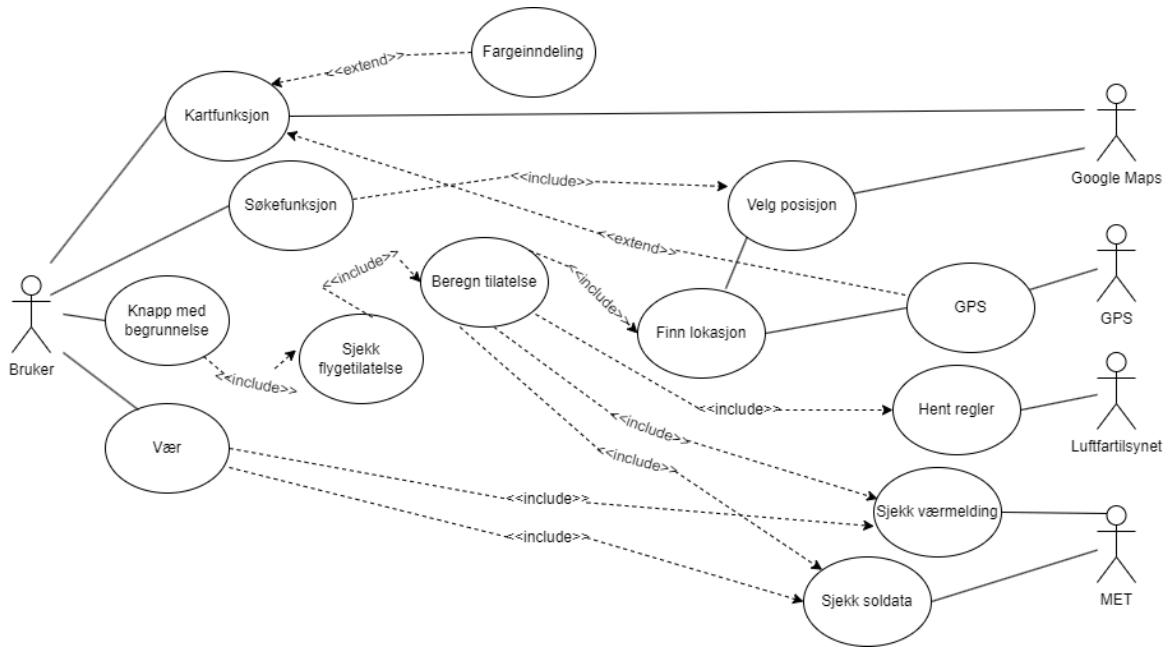
De viktigste funksjonelle kravene basert på målgruppen var å ha en *kartfunksjon*, *knapp med begrunnelse*, *GPS*, *søkefunksjon*, *værinformasjon*, og *fargeinndeling*. Disse blir nærmere forklart kravspesifikasjonen i figur 47 i Vedlegg ’Opprinnelig kravspesifikasjon’.

Disse kravene utgjør grunnlaget for en overordnet brukerhistorie:

Som bruker ønsker jeg å sjekke om min lokasjon er trygg å fly drone på, slik at jeg følger dronereglene.

4.2 Use case diagram

Figur 10 illustrerer et use case diagram som er basert på de viktigste funksjonelle kravene, samt brukerhistorien.



Figur 10: Use-case diagram basert på brukerhistorien: *Som bruker ønsker jeg å sjekke om min lokasjon er trygg å fly drone på, slik at jeg følger dronereglene.*

I forbindelse med use-case diagrammet finnes det tre ulike tekstlige beskrivelser til tre av de viktigste kravene. Dette er for å gi en lettere forståelse av interaksjonen mellom bruker og system.

4.3 Tekstlige beskrivelser

Navn: Søkefunksjon

Aktører: Bruker og API

Prebetingelse: Bruker er inne på appen med nett på mobilen.

Postbetingelse: Bruker får se området på kartet de har søkt på.

Hovedflyt:

1. Bruker trykker på søkeikon.
2. Systemet viser en søkebar øverst på skjermen.
3. Bruker skriver inn navnet på ønsket lokasjon.
4. Systemet viser forslag på steder underveis basert på søket til brukeren.
5. Bruker trykker på det riktige forslaget.
6. Systemet flytter kameraet til valgt posisjon på kartet.

Alternativ flyt:

- 4.1. Systemet gir ikke forslag da det ikke er noen steder med det.
- 4.2. Bruker må gå tilbake til steg 3.

Navn: Se værmelding for valgt posisjon

Aktører: Bruker og API

Prebettingelse: Bruker er inne på appen med nett på mobilen, samt har godtatt bruk av posisjon.

Postbettingelse: Bruker får se værmeldingen.

Hovedflyt:

1. Bruker trykker på værikonet nede i navigasjonsbaren.
2. Systemet henter ut vær- og soldata basert på brukerens lokasjon.
3. Systemet viser værmeldingen på en ny side.

Alternativ flyt:

- 1.1 Brukeren markerer en lokasjon på kartet.
- 1.2 Brukeren trykker på 'Trykk for å se værmelding'.
- 1.3 Systemet henter ut vær- og soldata basert på den markerte lokasjonen.
- 1.4 Returnerer til steg 3.

Navn: Knapp med begrunnelse

Aktører: Bruker og API

Prebettingelse: Bruker er inne på appen med nett på mobilen.

Postbettingelse: Bruker får se begrunnelse på hvorfor de ikke kan fly på et valgt område.

Hovedflyt:

1. Bruker åpner applikasjonen.
2. Systemet ber om tillatelse om å bruke brukerens lokasjon.
3. Bruker godtar bruk av sin lokasjon.
4. Systemet viser hovedskjermen.
5. Bruker trykker på knapp med begrunnelse.
6. Systemet beregner tillatelse ut fra brukerens lokasjon.
7. Systemet viser en sjekkliste, og sier om brukeren har lov til å fly drone eller ikke.

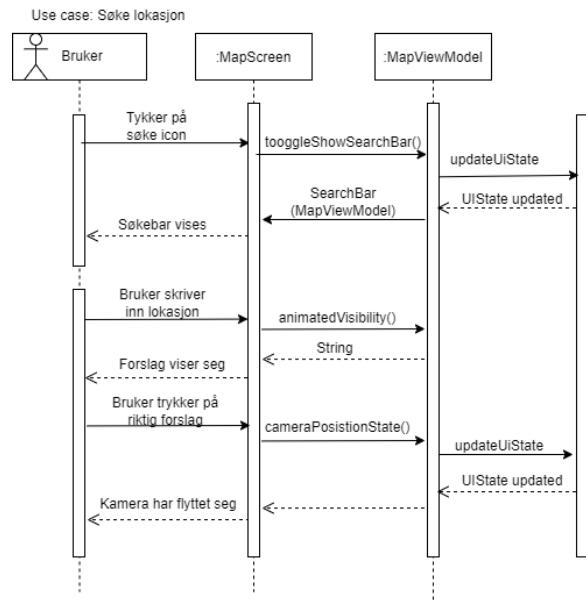
Alternativ flyt:

- 3.1 Bruker godtar ikke bruk av posisjon
- 3.2 Systemet setter posisjon på koordinat 0,0.

4.4 Klassediagram og sekvensdiagram av de viktigste use-casene

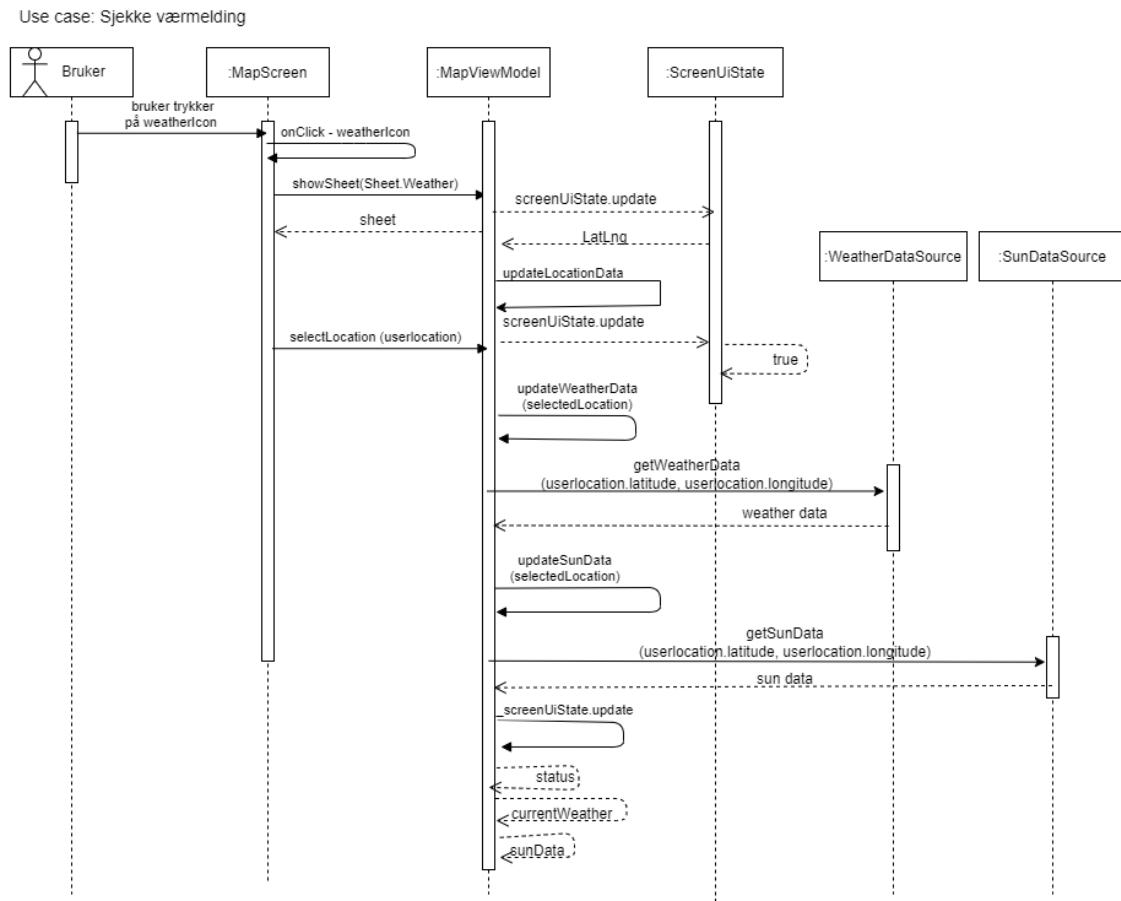
Figur 46 i vedlegg 8.5 illustrerer et klassediagram. Dette er laget for å gi en bedre og mer oversiktlig oversikt over sammenhengen mellom klassene.

Sekvensdiagrammene er laget basert på de tekstlige beskrivelsene, og er plassert i lik rekkefølge som de tekstlige beskrivelsene de illustrerer.



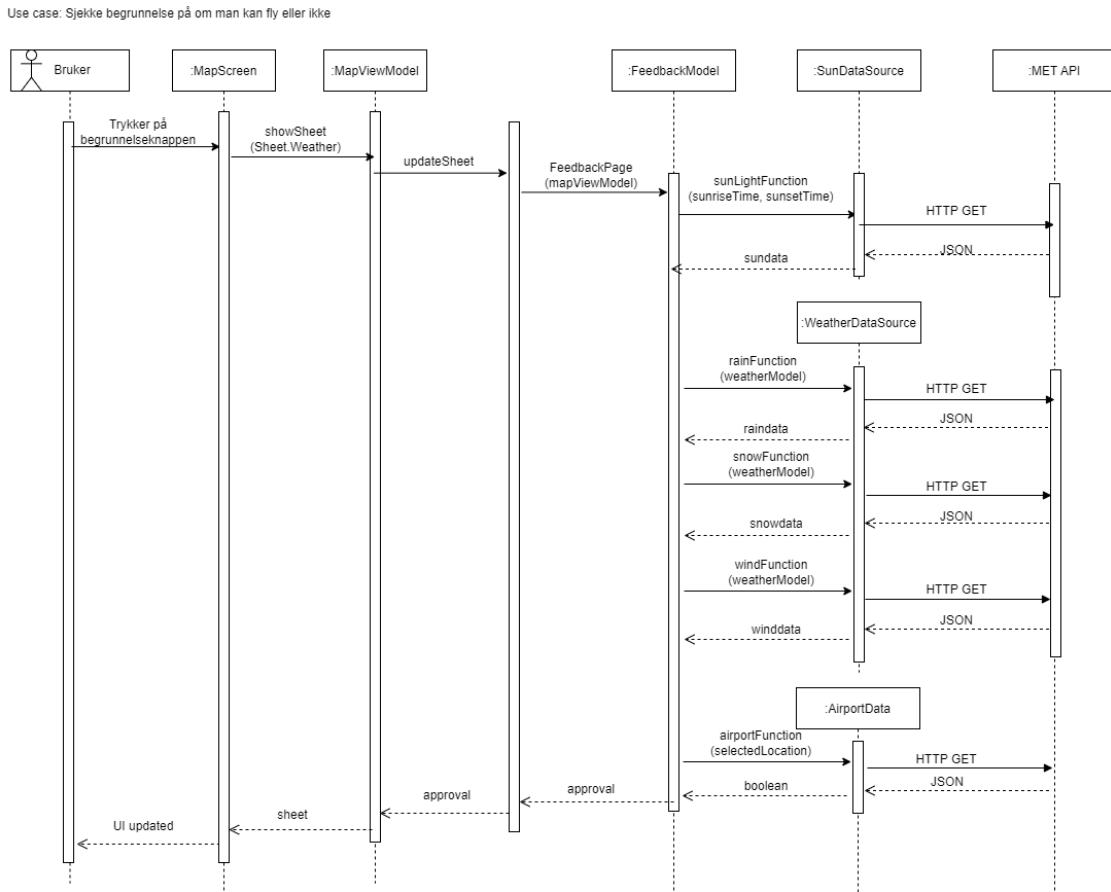
Figur 11: Sekvensdiagram av kravet 'Søke på lokasjon'

Figur 11 viser et sekvensdiagram over use caset 'Søke på lokasjon'. Dette use caset illustrerer tilfellet hvor brukeren søker etter en lokasjon og får den opp på kartet.



Figur 12: Sekvensdiagram av kravet 'Sjekke værmelding'

Figur 12 viser et sekvensdiagram over use caset 'Sjekke værmelding', og illustrerer situasjonen hvor brukeren trykker på vær-ikonet for å få opp værmeldingen.



Figur 13: Sekvensdiagram av kravet 'Sjekke begrunnelse'

Til slutt viser figur 13 et sekvensdiagram over use caset 'Sjekke begrunnelse', hvor brukeren trykker på begrunnelseskappen for å sjekke om vedkommende kan fly drone i det valgte området.

4.5 Objektorienterte prinsipper

4.5.1 Design patterns - MVVM

For å unngå teknisk gjeld og gjøre utviklingen av appen så smidig som mulig, følger appen designmønsteret **MVVM** (*Model-View-ViewModel*). Dette designmønsteret anbefales av forelesere, og anbefales også av mange andre kilder online (for eksempel Androids egen nettside for utviklere) (AndroidDevelopers, 2023).

MVVM er et designmønster som brukes for å separere det brukeren ser, UI, fra logikken bak applikasjonen. En stor fordel med **MVVM** er derfor at ulike teammedlemmer, for eksempel designere og utviklere, kan jobbe på hver sin del av koden i parallel. Dette er

mulig siden design- og forretningslogikkdelen av appen skrives uavhengig av hverandre. (MicrosoftLearn, 2022)

MVVM består av tre hovedkomponenter som har hvert sitt veldefinerte ansvarsområde:

- **Model:** Denne delen inneholder alt som skjer bak kulissene for appen. Dette inkluderer klasser som tar vare på data, og annen logikk for å innhente, bruke og endre denne dataen. (MicrosoftLearn, 2022)
- **View:** Ansvarlig for å vise frem data fra *Model*-delen til brukeren. *View* bestemmer altså selve designet til brukergrensesnittet, og skal ikke inneholde annen logikk eller beregninger som har å gjøre med dataen bak dette. *View* tar imot hendelser i form av bl.a. trykk på skjermen, og kaller på de relevante metodene i *ViewModel* hvis data skal endres eller beregnes. (MicrosoftLearn, 2022)
- **ViewModel:** Fungerer som et mellomledd mellom *Model* og *View*. *ViewModel* bestemmer hvilken data som skal vises frem i *View*, og er også ansvarlig for å reformattere eller behandle dataen fra modellen slik som den skal vises frem i *View*. *ViewModel* kan altså inneholde logikk for dette, men skal ellers inneholde så lite tunge beregninger og forretningslogikk som mulig. (MicrosoftLearn, 2022)

Diagrammet nedenfor (Figur 14) illustrerer hvordan *MVVM*-tankegangen er implementert.

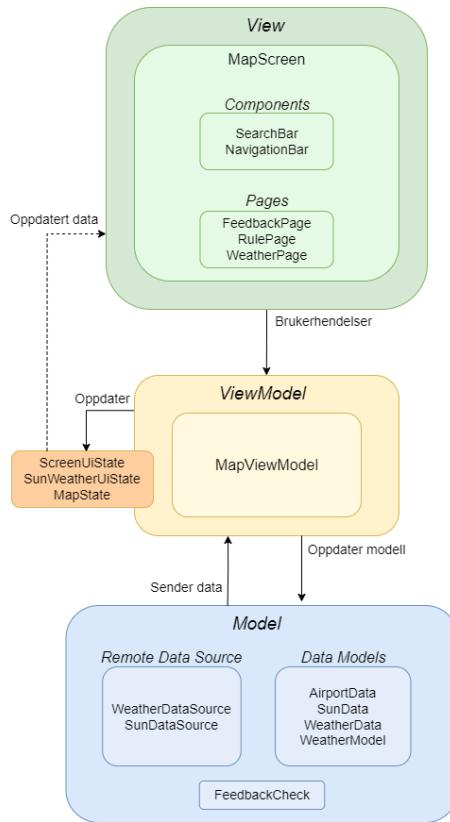
Diagrammet viser de tre hovedkomponentene, og navnet på klassene som vi laget for hver komponent. Diagrammet viser også hvordan komponentene interagerer med hverandre: Når *View* tar imot en brukerhendelse, kaller den på den relevante metoden i *ViewModel*. *ViewModel* vil da gjøre kall til modellen, som for eksempel henter data fra et API og returnerer dette til *ViewModel*.

ViewModel oppdaterer så en *UiState* med den dataen som skal vises frem. Siden *UiState* brukes av *View*, vil den oppdaterte dataen vises frem med det bestemte designet som er laget i *View*. På denne måten "vet" *View* bare om dataen den skal vise frem, og trenger ikke å være ansvarlig for logikken bak hvordan den hentes og manipuleres.

4.5.2 Lav kobling og høy kohesjon

Kobling og kohesjon er to tett knyttede begreper som beskriver hvor godt strukturert koden til programvaren er. Kobling er et mål på avhengigheten mellom de ulike objektene i programmet. Målet for et velstrukturert program er å ha lav kobling, slik at en endring i en komponent krever færre endringer i andre deler av koden. (Pagade, 2022)

Lav kobling er oppnådd i appen blant annet ved å bruke *MVVM*-mønsteret, som muliggjør å gjøre endringer i designet for appen (*UI*) uten å måtte endre i modellen.



Figur 14: Hovedkomponentene i MVVM

Kohesjon måler til hvilken grad elementene innenfor et objekt hører sammen, altså hvor veldefinert ansvarsområde objektet har og hvor stort dette ansvarsområdet er. Målet er å ha høy kohesjon. Høy kohesjon innebærer at hver klasse skal ha et lite ansvarsområde, og at dette ansvarsområdet skal være tydelig og veldefinert. (Pagade, 2022)

MVVM har bidratt til å oppnå dette for appen, ved å ha tydelig og avgrenset rollefordeling for de ulike komponentene. For eksempel har skjermen *MapScreen*, som er en del av *View*, bare ett ansvarsområde - å vise frem appens hjemmeskerm. Ansvarsområdet til en *View* er tydelig definert i *MVVM*-strukturen, og er avgrenset slik at *View* bare har ett ansvar.

5 Produktdokumentasjon

Produktdokumentasjonen tar for seg de viktigste kvalitetsegenskapene ved appen samt API, struktur og *Jetpack Compose*.

5.1 Viktigste kvalitetsegenskapene ved appen

5.1.1 Funksjonalitet

Appen tilbyr ulike funksjonaliteter som sammen tillater brukeren å ha tilgang til nyttig informasjon og veiledning for droneflyving samlet på ett sted. Appen tilbyr et stort kart der brukeren kan se egen plassering på dette kartet som oppdateres kontinuerlig. I tillegg har appen tegnet opp røde soner på kartet som illustrerer områder der droneflyving ikke er godkjent, på bakgrunn av at dette er i nærheten av en flyplass.

Appen tillater også brukeren å undersøke om andre lokasjoner tilfredsstiller flyvekrav ved å lage markører på kartet. Dette hjelper brukeren å vite om de kan reise andre steder og fly dronen deres der. For å gjøre dette lettere for brukeren er det implementert en søkefunksjon i applikasjonen som lar brukeren søke opp spesifikke steder. Brukeren vil da motta forslag som vedkommende kan velge mellom basert på input i søkerfeltet. Dette gjør at brukeren slipper å finne en lokasjon på kartet manuelt, og kan heller søke dette opp slik at kartet vil flytte seg til denne lokasjonen for brukeren.

Enten brukeren er nysgjerrig på sin egen lokasjon eller en markør-lokasjon, kan de sjekke værmeldingen og solinformasjonen på denne lokasjonen. På grunn av utfordringene med å fly drone når det regner og snør, skal denne funksjonaliteten hjelpe brukeren med å ha bedre oversikt over været. Luftfartstilsynet har også bestemt at dersom man skal fly drone ute når det er mørkt, må dronen ha et grønt, blinkende lys og være synlig for flyveren (Luftfartstilsynet, 2023b). Dette kravet fra Luftfartstilsynet, samt utfordringene med å fly drone i mørket, er grunnlaget for de implementerte soloppgang- og solnedgangstidene i appen.

Den siste funksjonaliteten, og antageligvis den mest nyttige for brukeren, er begrunnelsesiden. Begrunnelsessiden gir en samlet vurdering av om brukeren kan fly drone på en gitt lokasjon eller ikke. Denne funksjonaliteten samler informasjon slik at brukeren raskere og enklere kan sjekke omgivelsene, istedenfor å måtte sjekke alle forholdene individuelt og avgjøre på egenhånd. Dette hjelper med å oppfylle det ikke-funksjonelle kravet for appen om rask lokasjonsbegrunnelse, der brukeren skal få vite om de kan fly drone på sin lokasjon i løpet av 10 sekunder. Dette kravet kan sees i 49 i vedlegg *Oppdatert kravspesifikasjon*.

5.1.2 Brukskvalitet

For å kvalitetssikre appen har den blitt testet under utviklingen på ulike typer emulatorer, samt en fysisk *Android* telefon. Emulatorene har vært av ulike typer, som *Pixel 2* og *Pixel 5* med diverse ulike API-nivåer som 30, 31 og 33. Ettersom mobiler som *Pixel 2* har en svært kort skjerm i forhold til de litt større telefonene som *Pixel 5*, er begrunnelsessiden og

værsiden laget slik at de er *scrollable*. På denne måten kan brukeren bare bla nedover, og ingen informasjon blir mistet.

For å sikre brukskvalitet er det også utført to brukerintervjuer og en brukertest. Brukerintervjuene er grunnlaget for datainnsamlingen som er benyttet til å fastslå kravspesifikasjonen. Denne dataen er analysert og brukt til å forbedre appen. For informasjon om hvordan brukerintervjuene er utført, se kapittel 6.4. For informasjon om hvordan brukertesten ble utført og hva analysen er benyttet til, se kapittel 6.11.2.

5.1.3 Bugs

En *bug* som brukeren kan oppleve allerede fra oppstart av applikasjonen er når appen spør brukeren om de ønsker å gi tilgang til presis eller omtrentlig lokasjon. Dette dukker opp automatisk når applikasjonen åpnes for første gang. Funksjonaliteten i applikasjonen krever presis lokasjon, så dersom brukeren kun gir tilgang til omtrentlig lokasjon vil ikke appen klare å hente inn lokasjonen til brukeren, eller utnytte flere av appens funksjonaliteter. Dersom brukeren ender opp med å gi tilgang til kun omtrentlig lokasjon, er det anbefalt å slette appen og laste den ned på nytt slik at innstillingene blir nullstilt. Da vil denne forespørselet komme opp på nytt, og brukeren kan da velge presis lokasjon.

Bruker kan også oppleve at appen ikke vil fungere optimalt hvis en markør er satt på en lokasjon i en annen tidssone. Dette er fordi det er utfordrende å få hentet inn riktig tidssone for andre områder enn hvor brukeren befinner seg, og dermed justere appen basert på dette. Appen er derfor utviklet til å hovedsakelig fungere for lokasjoner i brukerens nåværende tidssone. Bruksmønstre som utfordrer dette vil i praksis påvirke tidspunktene for solnedgang og soloppgang, samt funksjonaliteten som sjekker for nok sollys i begrunnelsessiden. Dette kan for eksempel oppleves dersom man kjører appen i emulatorene i Android Studio ettersom disse kjører fra en Google-server som vanligvis befinner seg fysisk i USA.

Når man har plassert en markør, vil det noen ganger dukke opp en *bug* når man klikker på værsiden og begrunnelsessiden. Ikonet øverst til høyre viser at brukeren er på egen lokasjon, men dataen for markørens lokasjon vises. Dersom man klikker på ikonet for markørens lokasjon, deretter på ikonet for egen lokasjon igjen, vil riktig informasjon vises.

I *build.gradle* filen for appen har vi en rød *error* i én av *dependencies*'ene. Denne *dependency*'en heter 'com.google.firebaseio:firebase-crashlytics-buildtools:2.9.4'. Dette er en *import* vi ikke har lagt inn selv da den mest sannsynlig har vært tilstede siden opprettelsen av prosjektet. Vi er derfor også usikre på når nøyaktig denne *error*'en begynte å komme. Selv om dette vises som en *error*, har vi aldri opplevd at dette har vært et problem for utviklingen av appen vår, og vi har derfor ikke prioritert å løse opp i dette.

Ellers inneholder *build.gradle* filen vår mange *warnings*, men alle disse er påminnelser om

at det finnes en nyere versjon av ulike *imports* vi har lagt inn. For å unngå muligheten for at funksjonalitet i appen vår plutselig skulle slutte å fungere, har vi latt være å oppdatere *import*'ene til nyeste versjon underveis i prosjektet.

5.2 API

Applikasjonen er laget for *Android*-telefoner med API nivå 33, som tilsvarer Android-versjonen *Android 13*. Selv om alle telefoner med API nivå 23 eller høyere vil kunne kjøre appen, vil den derfor fungere best for telefoner med minimum API 33. Ved å utvikle appen med et høyere API-nivå i fokus, er det lettere å ivareta sikkerhet og ytelse for appen. Det gjør det også mulig å benytte nyere biblioteker og funksjoner. Blant annet på grunnlag av dette, anbefaler plattformen *Google Play* at utvikling av nye apper skal være rettet mot minimum API nivå 33 (Developers, 2023).

5.2.1 API fra Meteorologisk Institutt

Siden appen skal gi tilbakemelding på om både vær- og lysforholdene er gode for flyving av drone, er det benyttet to forskjellige API-er fra MET: *Locationforecast/2.0* for innhenting av værdata, og *Sunrise 3 beta* for soldata. For å hente værikonene tilhørende *Locationforecast* er i tillegg *Weathericon/2.0* fra MET benyttet.

Locationforecast

Dataen hentes inn i JSON-format, som er standarden for *Locationforecast* versjon 2.0. *Locationforecast* returnerer et værvarsel for de neste 60 timene fra kallet på API-et, og værvarselene som mottas er i 6-timers intervaller. Altså returneres ganske mye data ved en forespørsel, og en utfordring oppstod derfor når det gjaldt å forstå dataens representasjon og hvilke deler som skulle hentes ut. Bortsett fra det har API-et fungert fint for prosjektet. Dataen fra kallet som brukes i applikasjonen er temperatur, vindhastighet, forventet nedbør den kommende timen, samt en summert tekstbeskrivelse av været for den kommende timen.

Locationforecast er valgt fremfor Nowcast ettersom Nowcast kun returnerer værdata for de neste 15 minuttene. For fremtidig utvikling av appen kan *Locationforecast* være lønnsomt dersom nye funksjonaliteter, som for eksempel oversikt over fremtidig vær, skal bli implementert.

Weathericon

Weathericon returnerer en mengde værikoner, og er ment å brukes sammen med *Locationforecast*. Værikonene samsvarer med de ulike værbeskrivelsene som kan returneres i et kall på *Locationforecast*. Alle ikonene er lastet ned i PNG-format og ligger i *drawable*-mappen til *Android*-prosjektet for appen.

Sunrise

Dette API-et kan enten returnere data om solen eller månen, men for appen trengs kun soldata - derfor brukes bare denne delen. Et kall på sol API-et returnerer tidspunktene for soloppgang og -nedgang for den gitte dato, i tillegg til annen geometrisk data om solens posisjon. Kun dataen om soloppgang og solnedgang er benyttet. Versjonen som brukes er *3 beta*, som returnerer data i JSON-format.

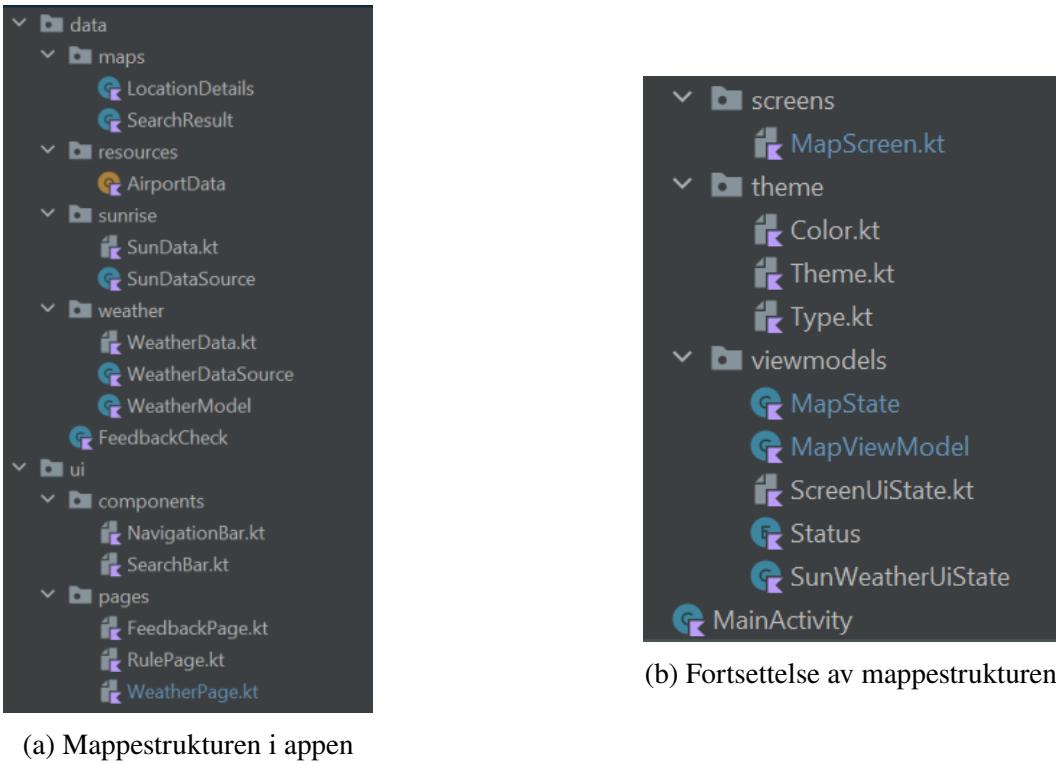
5.2.2 Andre APIer

Kartfunksjonen i appen har implementert *Google* sitt *Maps SDK-API*. Ettersom *Google* har skapt *Android*-plattformen er dette det mest kompatible API-et for kartet i prosjektet.

I kombinasjon med *Maps SDK*, utnytter også appen *Google* sitt *Places SDK* for innehenting av stedsforslag i søkefunksjonen. Dette API'et lar oss innhente lokasjonsforslag når brukeren søker opp en lokasjon, samt koordinatene knyttet til disse lokasjonene. Dette hjelper kartet å flytte kameraet til lokasjonen som brukeren velger. Det samme API'et benyttes også i værsiden for å vise lokasjonsnavnet på posisjonen til brukeren eller markøren.

5.3 Struktur

Programmet er delt inn i mange forskjellige mapper som igjen er delt inn i undermapper, som vist i figur 24a og 24b. Mappeinndelingen følger en *MVVM*-struktur der *Model*, *View* og *ViewModel* utgjør hovedmappene. I prosjektet heter disse *data*, *ui* og *viewmodels*. I hovedmappen *data* ligger alt av dataen og logikk bak appen. Den er igjen delt inn i mappene *maps*, *resources*, *sunrise* og *weather*. Her ligger filer som inneholder eller tar imot data om de spesifikke ansvarsområdene de har blitt tildelt. For eksempel ligger alt om værdata i undermappen *weather*. Hovedmappen *ui* har ansvar for visning av dataene - her finnes undermappene *components*, *pages* og *screens*. I undermappen *screens* ligger koden bak det visuelle vi ser i appen. Innholdet i undermappene *components* og *pages* er med på å få dette til å skje. *viewmodels* fungerer som en mellommann mellom modellen og visningen - her finner du en *viewmodel* og forskjellige *UI states* som brukes av *UI*.



5.4 Jetpack Compose

Jetpack Compose er et moderne deklarativt *UI*-rammeverk som har blitt brukt i utviklingen av appen. Det skiller seg fra tradisjonell *XML*-basert utvikling ved å tilby en mer intuitiv og effektiv tilnærming til *UI*-design og -implementering. (AndroidStudio, n.d.)

En viktig funksjon i *Jetpack Compose* er bruk av *State*. *States* tillater oss å definere og håndtere tilstanden til brukergrensesnittet på en enkel og strukturert måte. Ved å bruke *states* kan grensesnittet oppdateres dynamisk basert på endringer i data eller brukerinteraksjoner. Dette gjør det mulig å oppnå en sømløs oppdatering av *UI* uten behov for manuell synkronisering eller kompleks *XML*-håndtering. (AndroidStudio, 2023a)

Et annet konsept i *Jetpack Compose Composables*. *Composables* er en måte å dele opp brukergrensesnittet i mindre og gjenbrukbare komponenter. Dette gjør det enkelt å bygge og vedlikeholde appen ved å dele opp UI-elementene i logiske deler. Ved å bruke *composables* kan vi isolere og gjenbruke kode, noe som gir en mer modulær og skalerbar tilnærming til utviklingen av brukergrensesnittet. (AndroidStudio, 2023b)

5.5 Ktor

Dataen fra MET API-ene kommer i JSON format. For å gjøre om dette formatet til objekter med værdata som kan brukes i koden (deserialisering), er det vanlig å bruke et rammeverk.

Det finnes mange å velge mellom, men vi har benyttet rammeverket *Ktor*, da det var dette vi var kjent med fra før.

6 Prosessdokumentasjon

Prosessdokumentasjonen beskriver prosessen med å jobbe med appen. Her finnes informasjon om verktøy, smidig metode, rollefordeling, sprintenes gang, GDPR, universell utforming og testing.

6.1 Verktøy

Liste over verktøyene teamet benyttet seg av:

Miro ble benyttet til brainstorming, oversikt over oppgaver og fremdrift, samt oversikt over teamets styrker og svakheter.

Overleaf ble benyttet til å skrive rapporten, ettersom det ga oss muligheten til å skrive i samme dokument, og teamet fikk en fin layout på rapporten.

GitHub ble hovedsakelig benyttet til versjonskontroll.

Discord ble brukt som vår kommunikasjonskanal. Her la teamet inn ulike kanaler hvor planer, tidspunkter, ideer, lenker og annet kunne diskuteres.

Drive ble benyttet til å holde oversikt over ulike dokumenter, slik som loggføring fra møtene, samarbeidskontrakt etc.

Teams ble benyttet for kommunikasjon med veileder.

Figma ble benyttet for prototyping og skissing av visuelle idéer til design.

6.2 Valg av smidig metode

I valget av smidige metoder valgte teamet å kombinere *Scrum* og *Kanban* for å dra nytte av fordelene til begge. I *Scrum* deler man opp arbeidet i sprinter. Hver sprint startet med et sprint-planleggingsmøte, såkalt *sprintplanning* (Sommerville, 2021, s.47). Vi valgte da prioriterte oppgaver fra listen med oppgaver som skulle gjøres (*Product Backlog*) (Sommerville, 2021, s.42), som teamet mente vi kunne få til i løpet av sprinten. I resten av sprinten utførte vi *sprint execution* (Sommerville, 2021, s.47) der teamet kun fokuserte på disse oppgavene. Nye oppgaver som dukket opp underveis måtte vente til neste sprint. For å følge prosessen brukte teamet et kanban-brett. De oppgavene som ikke ble fullført ble lagt tilbake i backloggen. I starten av hvert møte hadde teamet et *stand-up* møte (Sommerville, 2021, s.50) som i utgangspunktet ikke skulle ta så lang tid. Her snakket teamet om hva alle

jobbet med og om vi hadde støtt på noen problemer som vi trengte hjelp med. Sprinten endte med to ritualer, en *sprint review* som er en demonstrasjon av de nye funksjonalitetene til produkteier (veileder) (Sommerville, 2021, s.47), og sprint-retrospeksen, også kjent som *review meeting* (Sommerville, 2021, s.40), for å diskutere positive og negative sider ved sprinten.

Underveis ville gruppen vurdere hvordan dette fungerte for oss, og hvorvidt vi skulle fortsette med *Scrumban*. Ettersom valget av smidige metoder ble tatt tidlig i prosjektløpet var det vanskelig å vite hvilke smidige metoder som fungerte for oss, og vi tenkte å prøve oss litt frem for å finne den beste løsningen. I starten ønsket vi å prøve ut sprinter med varighet på omtrent to uker, og deretter vurdere hvordan det fungerte for oss og hvorvidt vi skulle fortsette med det.

Sprintene endte opp med å ha varierende lengde med hard deadline, noe som var en kombinasjon av *Kanban* og *Scrum* (Rehkopf, n.d.). Den varierende lengden stammet fra *Kanban*, mens den harde tidsfristen kom fra *Scrum* (Rehkopf, n.d.). Vi hadde definerte roller fra start, som også hører til *Scrum* (Sommerville, 2021, s.39), og hadde samtidig en visuell fremstilling av oppgavene (som hører mer til *Kanban*). Møtetypene og reglene om å ikke endre på oppgavene underveis hører også til *Scrum* (Rehkopf, n.d.). Som vist i figur 16 ser vi at de fleste prinsippene vi brukte tilhørte *Scrum*, mens noen også tilhørte *Kanban*. Derfor kan vi konkludere med at teamet benyttet seg av *Scrumban* hele veien.

Prinsipper	Scrum	Kanban
Varierende lengde på sprints		x
Sprint-retrospekt	x	
Scrum master og spesifikke roller	x	
Daily standups	x	
Kanban-brett	x	x
Sprintplanleggingsmøte	x	
Ikke lov til å endre på, legge til eller fjerne oppgaver underveis	x	
Produktbacklog	x	
Hard deadline	x	

Figur 16: Tabellen viser om prinsippene i den smidige metodikken hører til Scrum eller Kanban.

6.3 Oppstartsfase

Teamet møttes for første gang under kickoffen. Der fikk vi muligheten til å bli litt bedre kjent med hverandre, og ble samtidig presentert for de ulike caseoppgavene. Kickoffen gjorde det enkelt for oss å komme i kontakt med hverandre, og var et fint tilbud som vi verdsatte.

6.3.1 Teamet

Under første møte bestemte gruppen seg for å lage et presentasjonskort av hvert teammedlem inne i *Miro*, slik figur 1 i seksjon 1 *Introduksjon* viser. På kortet skulle man skrive litt om

seg selv, sine styrker, svakheter og hva man selv ønsket å bidra med. Dette gjorde gruppen som en øvelse for å bli bedre kjent, men også for å få en bedre oversikt over eventuelle ansvarsområder man kunne gi til hvert teammedlem basert på deres preferanser.

Vi oppdaget også at teammedlemmene hadde forskjellige bakgrunner og kompetanse, noe som ville være fordelaktig for prosjektet. Alle følte at de kunne bidra med sine styrker, og få hjelp med sine eventuelle svakheter. Teamet ble enige om ambisjonsnivå og arbeidsmengde, og kom godt overens med hverandre. Dette var noe vi ville dra nytte av i samarbeidet.

For å sikre at alle var på samme side og hadde klare forventninger til samarbeidet, utarbeidet gruppen en teamavtale. Denne avtalen tjente som retningslinjer og veiledning for vårt samarbeid. Vi diskuterte og definerte tilstedevarsel, tidsbruk og forventninger. Vi diskuterte også hva teamet skulle gjøre dersom det oppstod uenigheter. Ved å lage denne teamavtalen sikret teamet en felles forståelse og et solid grunnlag for effektivt arbeid og samarbeid. Teamavtalen ligger i vedlegg *Teamavtale*.

Teamet fordelte også roller. Disse ble fordelt utifra behov og interesser, og alle satte seg opp der de følte de kunne bidra mest. Til slutt kom vi frem til fordelingen av følgende roller:

Teknisk ansvarlig: Jonas, Felicia

Tester: Felicia, Jonas

ScrumMaster: Hinda

Møteansvarlig: Marte

GitMaster: Diana

Design-visuell: Karin, Hinda

Design-prosess: Marte

Researchansvarlig: Jonas

Rapport: Jonas, Karin, Diana

Verktøyansvarlig(miro, overleaf): Marte

Kontaktperson: Skulle bli satt når vi fant bruker

6.3.2 Bli-kjent-kveld

I begynnelsen av prosjektet ønsket gruppen å gjennomføre et sosialt arrangement som var totalt frakoblet fra alle de faglige diskusjonene vi vanligvis hadde. Siden hver av oss kun kjente ett annet medlem fra gruppen fra tidligere, virket det hyggelig å treffes etter skoletid for å bli bedre kjent. I den anledningen diskuterte vi flere ulike aktiviteter gruppen kunne gjøre sammen, og endte til slutt opp med å møtes hjemme hos Marte for *Paint 'N' Sip*. Dette går ut på å skaffe noe godt å drikke, og sette seg ned og male sammen.



(a) Gruppebilde fra 'Bli-kjent-kveld', med maleriene.



(b) Blide fra 'Bli-kjent-kveld'.

For å lære litt mer om hverandre bestemte gruppen i løpet av kvelden at vi skulle male et dyr som man følte reflekterte seg selv som person, og presentere for de andre hvorfor man malte akkurat dette dyret. Teamet følte at dette ble en veldig hyggelig kveld der vi ble bedre kjent, og koste oss med mat og drikke, maling og godt selskap. Det var også behagelig å legge vekk alt av teknologi og skolearbeid, og kunne slippe seg fri i kreativiteten sammen med de andre. Vi avrundet kvelden med et felles gruppebilde bestående av medlemmene i gruppa og de seks fine maleriene. Se figur 17a.

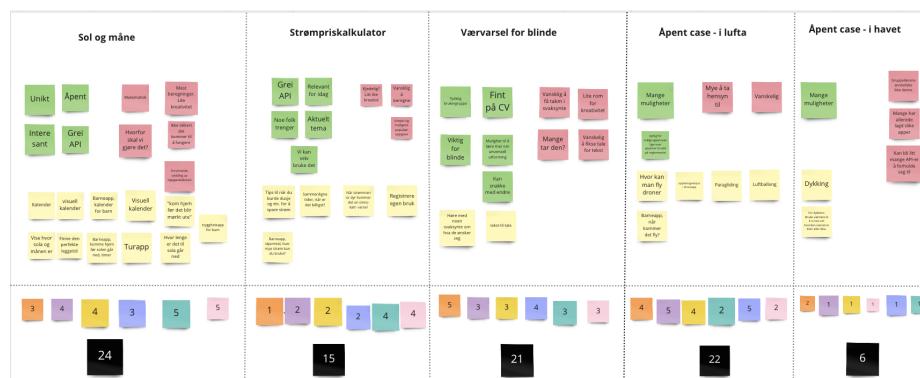
6.3.3 Valg av case

I oppstartsfasen brukte teamet litt tid på å bestemme seg for hvilket case vi skulle gå for. For å gjøre valget enklere prøvde vi oss på en brainstorming rundt de ulike casene, slik figur 18 viser. Vi gikk gjennom hvert case og skrev ned de ulike ideene vi hadde rundt caset (gul lapp), samt positive (grønn lapp) og negative (rød lapp) sider. Etter brainstormingen skrev alle hver sin lapp der de rangerte de ulike casene fra 1 til 5, der 1 representerte det de hadde minst lyst til å gjøre, mens 5 var det de hadde mest lyst til å gjøre. Vi summerte tallene sammen, og planla å velge caset med flest stemmer. Opprinnelig hadde case 1 flest

stemmer, men etter noe diskusjon konkluderte teamet med at denne caseoppgaven ble litt for utspekket. Derfor gikk vi for casen med nest flest stemmer, altså case 4 *Åpent case - i lufta*. Rangeringen av casene er illustrert i figur 19, hvor den svarte lappen viser summen av stemmene.



Figur 18: Figuren viser gruppens brainstorming av casene. Gul lapp representerer ideer vi hadde, mens grønn og rød lapp representerer henholdsvis positive og negative sider ved valg av caset.

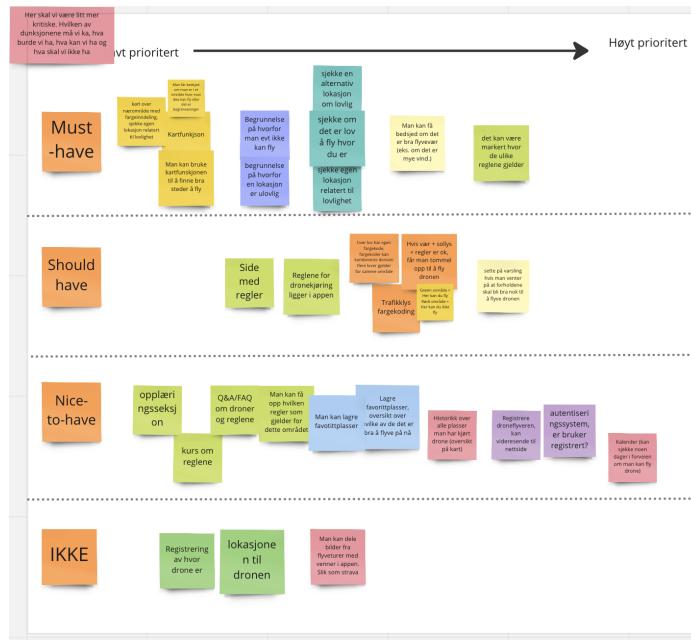


Figur 19: Etter brainstormingen stemte vi på det caset vi hadde mest lyst til å ta. Resultatet av stemmene sees på den svarte lappen.



Figur 20: Idéer til funksjonelle krav, gruppert etter kategori.

Etter å ha diskutert og stemt på alle caseoppgavene, konkluderte vi med at case 4 passet oss godt. Et åpent case ga oss friheten til å være kreative, og vi fant 'I luften' mer appellerende enn 'på havet'. Vi følte at vi kunne utforske mer kreativitet i denne oppgaven og lage noe nytt. Spesielt virket 'droner og luftrom' spennende, og vi bestemte oss for å lage en droneapp. Det hjalp at noen av teammedlemmene kjente personer som fløy droner, slik at vi hadde en lett tilgang til brukergruppen. Vi så det også som nyttig og spennende å lære mer om lovverket for droneflyving. Vi initierte derfor en brainstorming for å utforske denne ideen ytterligere. Målet var å nedtegne alle ideene vi kunne komme på for appen. Vi begynte med å ta fem minutter hver for å skrive ned alle ideene. Deretter diskuterte vi ideene og grupperte dem. Dette er illustrert i figur 20.



Figur 21: Etter å ha brainstormet rundt funksjonelle krav kategoriserte vi kravene i 'Must have', 'Should have', 'Nice to have' og 'Ikke'.

Etterpå klassifiserte vi ideene som "must have" (må ha), "should have" (bør ha), "nice to have" (kunne ha) og "ikke ha" (ikke nødvendig). I tillegg plasserte vi de langs en prioriteringsakse, hvor den går fra lavest til høyest prioritet. Med utgangspunkt i denne inndelingen utarbeidet vi en kravspesifikasjon. Disse er listet som tabell i figur 50, 51 og 52 i vedlegg *Opprinnelig kravspec*. Vi var åpne for å eventuelt gjøre endringer senere dersom brukerintervjuene åpnet opp for nye funksjonaliteter eller annerledes prioriteringer.

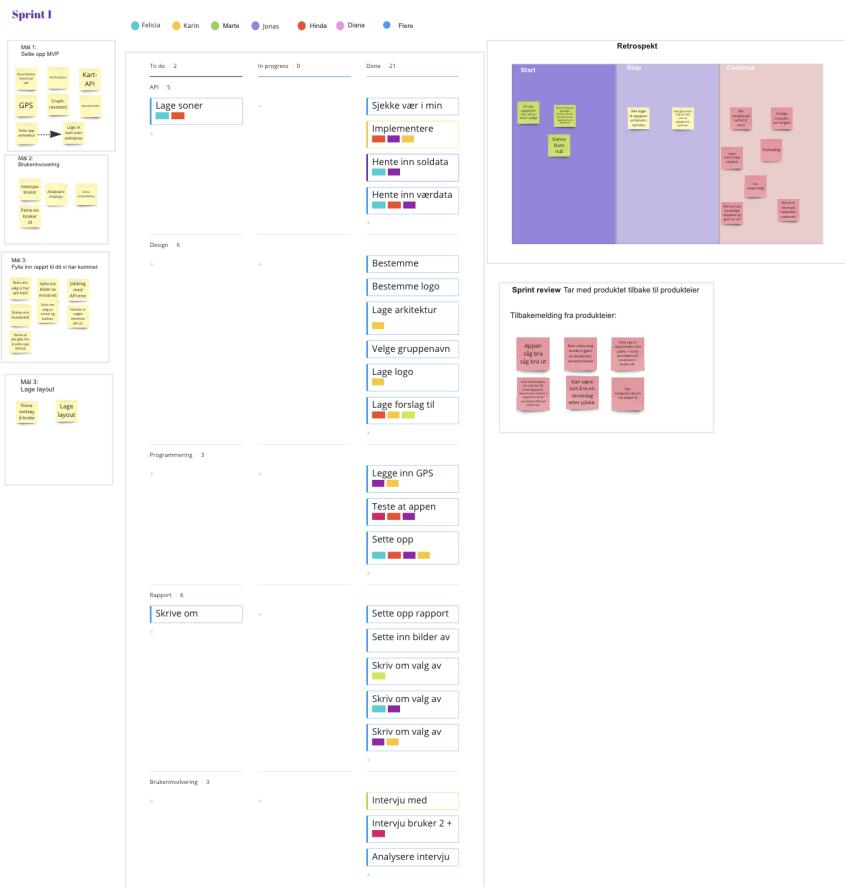
6.4 Sprint 1

I første sprint fokuserte vi på å sette opp vår *MVP* (*Minimum Viable Product*). En *MVP* skal inneholde de mest essensielle funksjonene, altså funksjonene som vi minst må ha for

å kunne levere verdi til kunden. Basert på vår kravspesifikasjon kom vi frem til at dette innebar å sette opp en grunnleggende arkitektur, laste inn APIene for soldata og værdata, implementere *Google Maps* og legge inn GPS.

Vi begynte på det grunnleggende designet, valgte API-er, hadde intervju med to brukere og skrev på rapporten. Vi bestemte oss også for logo. De to brukerne vi intervjuet var bekjente av oss i 20-årene som fløy drone, og det var derfor enkelt å komme i kontakt med dem. Intervjuene ble gjennomført ved å først utdele et samtykkeskjema, før vi deretter gjennomførte selve intervjuet ved hjelp av intervjuguiden vår (se vedlegg *Intervjuguide* og *Intervjuplan*). Analysen fra intervjuene finnes i figur 44 og 45 i vedlegg *Analyse fra brukerintervju*.

Sprinten varte i to uker og ble avsluttet med en *sprint review*, hvor vi presenterte innholdet til veileder (produkteier), og en retrospekt, hvor vi så på hva som fungerte og hva som ikke fungerte så bra.



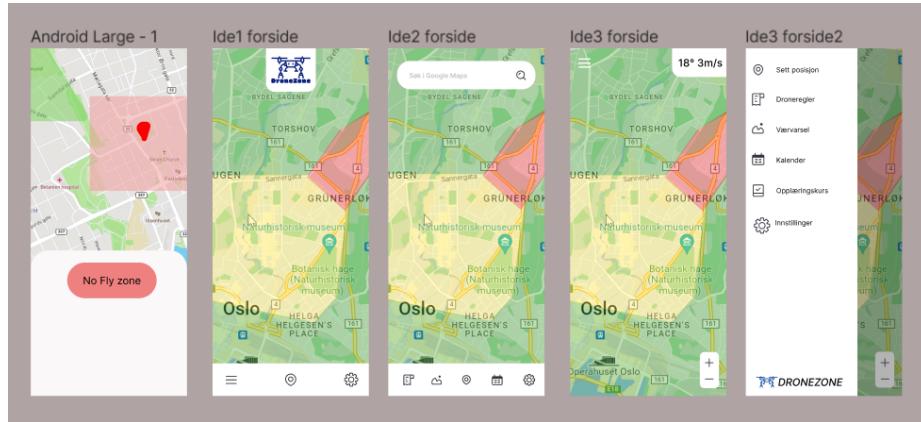
Figur 22: Figuren viser mål for sprinten, Kanban-brettet og retrospeksen for sprint 1. Det var slik vi satte opp alle sprintene våre.

Fra kanbanbrettet i figur 22 kan vi se at det var to oppgaver vi ikke fikk fullført i løpet av

denne sprinten. Disse ble lagt tilbake i backloggen, for å jobbes med i en senere sprint.

6.4.1 Prototyping

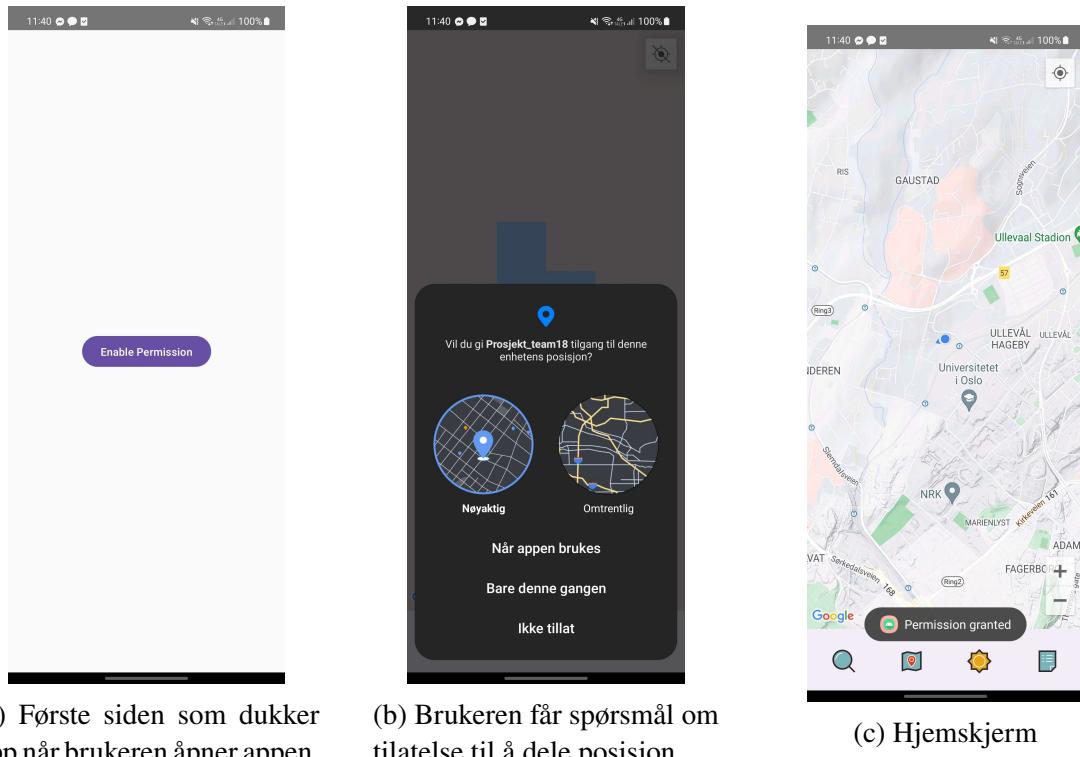
En essensiell del av sprint 1 var å prototype. Det vil si å lage en tidlig og enkel versjon av systemet for å teste ut ulike ideer (Sommerville, 2021, s.26). At vi lagde en *MVP* kan en regne som en *høyoppløselig prototype*, da den er veldig lik det endelige produktet (Joshi, 2020a, s.58). Som vi nevnte tidligere i bruken av verktøy, brukte vi også nettsiden *Figma* til å lage noen mer *lavoppløslige prototyper* vist i figur ??.



Figur 23: Lavoppløslige prototyper av ideer til design av appen

Disse er lavoppløslige da de var mye raskere å lage, og lettere å endre for å lage flere versjoner av ideer og potensielle design (Joshi, 2020a, s.35).

6.4.2 Funksjonalitetene i MVPen

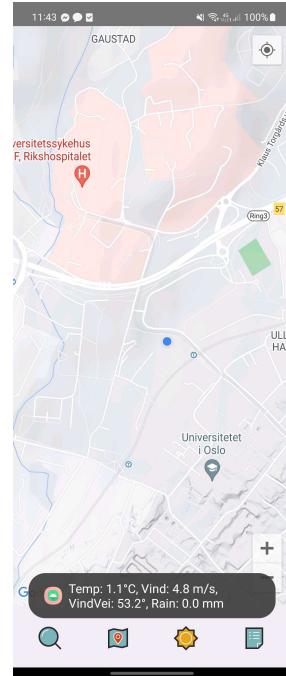


Figur 24

Figur 24a viser skjermen som vises første gangen brukeren åpnet *MVP-en*. I hendelseforløpet over trykker brukeren på knappen for å tillate posisjonen, og blir så bedt om å velge når appen skal få tillatelse til å samle data om den nåværende posisjonen, figur 24c.



(a) Hjemskjerm med søkefelt



(b) Værmelding vises som en toast-melding nederst på skjermen.

Figur 24c, 25a og 25b viser *MVP*-ens hjemmeskjerm og tre funksjonelle krav vi implementerte/begynte å implementere: Kartfunksjon, søkefunksjon og værfunksjon. Søkefeltet var ikke ennå koblet til kartet, og værmeldingen ble gitt til bruker i form av en kort melding (kalt *Toast message* i *Android*). Dette var en rask løsning for å ferdigstille *MVP*-en.

Siden kartfunksjonen er en sentral del av vår app, valgte vi å la den ta opp den største delen av hjemmeskjermen. Resten av funksjonene skal kunne benyttes gjennom å trykke på ikonene i navigasjonsbaren under kartet. Vi tenkte at dette gjorde det lettere for brukeren å navigere appen, siden kartet alltid vil ligge i bakgrunnen og det da ikke går an å 'gå seg vill' fra hjemmeskjermen. En navigasjonsbar er også noe som benyttes av de fleste velkjente appene idag, så det ville mest sannsynlig føles naturlig for bruker å også navigere vår app på denne måten.

6.4.3 Utfordringer i sprint 1

Teamet møtte på en utfordring da vi skulle kjøre appen med *Google Maps* implementert. Det fungerte hos noen, men ikke hos alle. Etter litt feilsøking ble problemet løst, men deretter oppstod det nye problemer knyttet til *Google Play Services* og at våre enheter ikke støttet dette. Dette fikk vi til slutt ordnet opp i ved å gi API-nøkkelen tilgang til alle våre maskiner, noe som tok sin tid. På noen av maskinene måtte vi også endre emulatoren fra telefonmodellen *Pixel 5* til *Pixel 2*, noe som løste problemet.

En stor utfording med utviklingen av *MVP*-en var å implementere funksjonalitet for å innhente brukerens lokasjon, samt implementere *Google Maps*. Utfordringene med dette gikk særlig på at det finnes få ressurser og at de aller fleste ressursene allerede er utdatert ettersom *Android Studio* og *Jetpack Compose* er i kontinuerlig utvikling. Her opplevde vi at veldig mange løsninger allerede var utdaterte, og det var vanskelig å bygge opp et fungerende program fra mange ulike ressurser der det stadig var utdatert kode. Dette er generelt en utfording med utvikling i *Jetpack Compose*, men særlig dersom man skal benytte seg av tilleggsAPI-er.

En annen utfording i første sprint var knyttet til GitHub, noe som var naturlig ettersom dette er et verktøy vi ikke har benyttet oss så mye av tidligere. Blant annet møtte vi på konflikter da vi prøvde å *pulle* og *pushe* prosjektet, og det oppstod problemer med lokale endringer. Disse utfordingene tok en del tid og var et hinder i prosessen.

6.4.4 Retrospekt sprint 1

I løpet av den første sprinten opplevde vi at vårt valg av smidige metoder fungerte bra. Oppgavene vi prioriterte i sprintplanleggingsmøtet var overkommelige, og varigheten på sprinten (2 uker) var akkurat passelig. Vi ble også ferdige med *MVP*-en i tide, og skrev på rapporten underveis. Teamarbeidet fungerte bra - alle tok arbeidet seriøst, og gjorde det de skulle.

Fordelen med å fokusere på å fikse ferdig *MVP*-en den første sprinten var at alle fikk et tydelig bilde av hva de viktigste kravene til appen var. Siden kun det mest essensielle trengtes, unngikk vi å henge oss opp i små detaljer og kunne istedenfor konsentrere oss på å få ut et ferdig produkt. Når vi så var ferdige med *MVP*-en var det lett for oss å se hva som måtte fikses og legges til de kommende sprintene for å få et bedre sluttprodukt. Det var også motiverende å faktisk få til en fungerende app, sammenlignet med om vi bare hadde fokusert på å perfeksjonere noen få funksjoner av appen denne sprinten.

Datainnsamlingen og brukerintervjuene ga en pekepinn på hvilke funksjonaliteter brukerne ønsket seg, og disse ønskene samsvarer med kravspesifikasjonen som allerede var utformet. Blant annet var det et ønske om en oversikt over vær og vind, en opplæringsside med regler, samt en oversikt over hvor det er mulig å fly. Dette var krav som vi allerede hadde satt som høyeste prioritet, og dermed slapp vi å endre på kravspesifikasjonen ettersom brukernes ønsker samsvarer med vår visjon.

Det vi ønsket å forbedre til neste sprint var å dele opp oppgavene i mindre deler, samt bruke mer tid på planleggingen av oppgavene slik at vi ikke ville legge til nye oppgaver mens sprinten allerede var i gang. Retrospeksen kan sees i figur 26a, mens sprintreview er illustrert i figur 26b.



(a) Retrospekt etter sprint 1

(b) Sprintreview sprint 1

6.5 Sprint 2

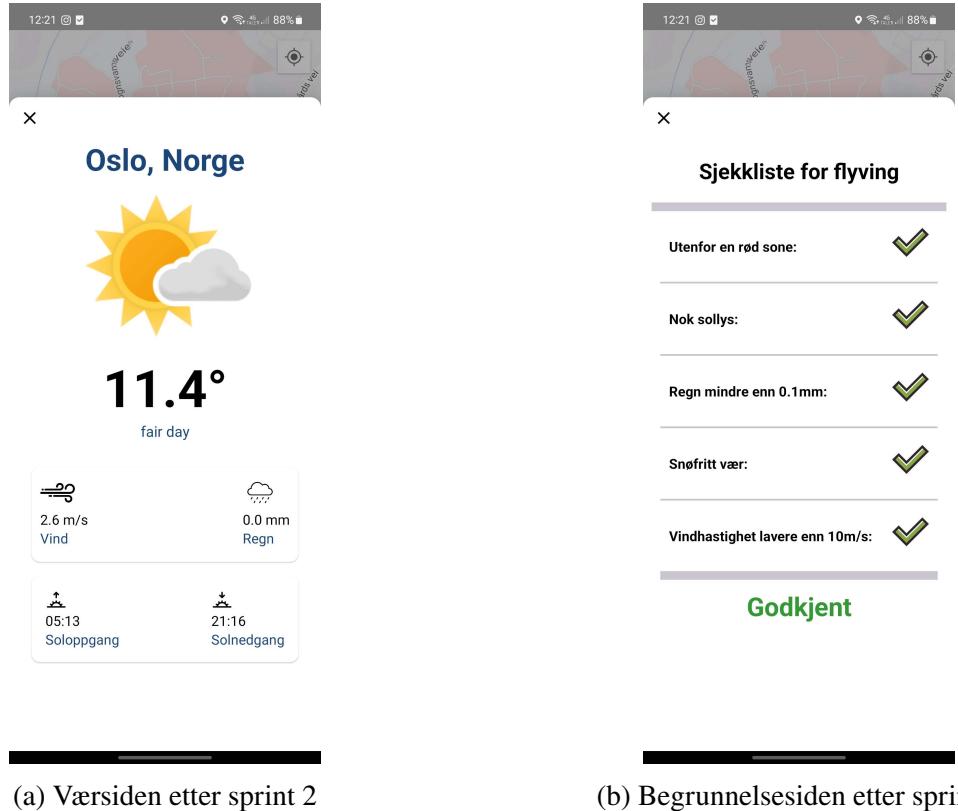
I sprint 2 ville vi hovedsakelig fokusere på å fullføre funksjonalitetene vi startet på i forrige sprint, og legge inn annen viktig funksjonalitet. Dette innebærte å gjøre ferdig søkefunksjonen slik at den kan søke opp og gå til adresser (samtidig pinne disse), legge inn flyplasser og forbudte soner rundt, legge inn soldata og tilbakemelding om været i værsiden, skrive inn ulike regler, samt designe de ulike sidene vi trengte. I tillegg ønsket vi i starten av sprinten å ha en skrivedag hvor alle sammen skrev på rapporten om det vi hadde gjort hittil. Varigheten til denne sprinten ble satt til 1,5 uke.

Ettersom det finnes mange regler knyttet til å fly drone valgte vi å fokusere på spesielt forbudte soner rundt flyplasser. Vi kom frem til at det ble for komplisert å legge inn funksjonalitet for alle reglene fordi vi ville trengt APIer med lokasjoner for alle forbudte områder å fly drone. Dette innebærer f.eks. flyplasser, fengsler, sykehus, samt mer dynamiske og midlertidige lokasjoner som festivaler. Vi valgte derfor å legge fokuset på flyplasser fordi det var et begrenset antall av disse og det var overkommelig å innhente denne informasjonen.

Vi valgte å hardkode inn flyplassene i Norge på kartet i stedet for å benytte et API for dette. Etter å ha diskutert fordeler og ulemper kom vi frem til at dette virket lettest ettersom vi kun trengte å finne koordinatene til flyplassene og legge disse i en ressurs (f.eks. en liste). Det var ikke så mange flyplasser å legge inn, så arbeidsmengden var overkommelig. En av grunnene til at vi valgte å ikke benytte et eksisterende API var fordi vi ikke ønsket å risikere å hente inn dataen fra API-et for mange ganger (og dermed bli blokkert fra API-et). Det var dessuten utfordrende å finne et passende API for flyplassene. Vi fant et API fra Geonorge ((Geonorge, n.d.)), men her måtte man velge projeksjon og format som vi ikke var kjent med. Derfor gjorde vi en vurdering på at det ville blitt mer ressurskrevende å lese oss opp på projeksjon og format samt hente data fra API-et, fremfor å bare hardkode inn koordinatene manuelt.

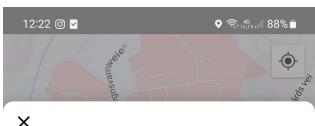
De ulike sidene for vær, regler og flyvetilatelse implementerte vi slik at de dukker opp som popup-sider over hjemmeskjermen når man trykker på tilsvarende ikon på navigasjonsbaren.

Brukeren kan enkelt stenge ned siden ved å trykke på x-en i venstre hjørne, eller bare dra/sveipe ned siden.

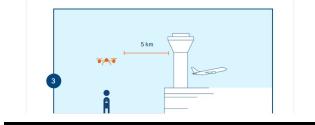
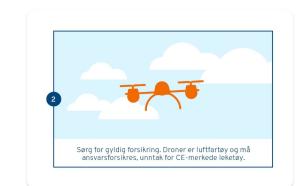


Figur 27

Vi utviklet også værsiden, 27a, som viser vær- og soldata for brukerens valgte lokasjon. Begrunnelsesiden i figur 27b ble også utviklet, men viser på dette tidspunktet kun *dummy-data* for å illustrere designet uten noe reell funksjonalitet.

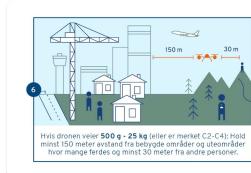
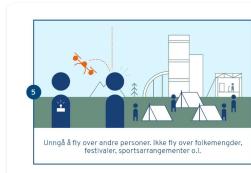


Regler for å fly drone



(a) Regelsiden etter sprint 2

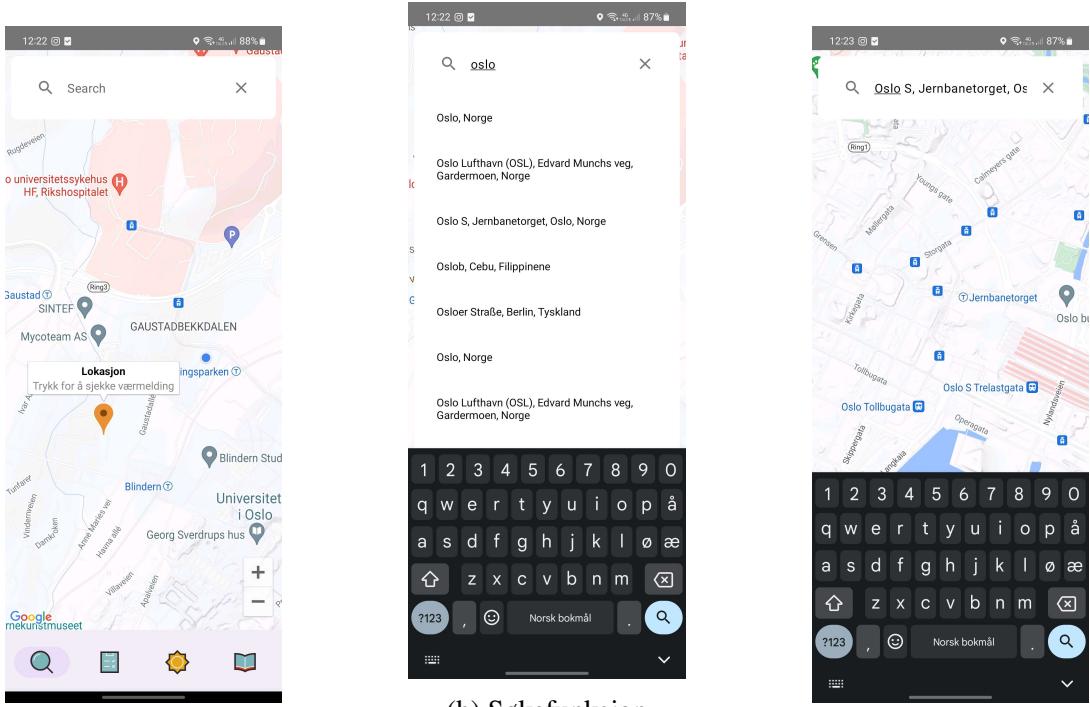
Regler for å fly drone



(b) Regelsiden etter sprint 2

Figur 28

Figurene 28a og 28b illustrerer designet av regelsiden som også ble utviklet. Brukeren kan bla opp og ned i siden for å se alle reglene. Reglene og bildene er hentet direkte fra Luftfartstilsynets nettside da vi syntes denne ressursen både illustrerte og beskrev dronereglene tydelig.



(a) Kart med markert lokasjon. Den markerte lokasjonen vises som oransje pin.

(b) Søkefunksjon
Når man skriver i søkefeltet dukker det opp forslag til adresse.

(c) Søkefunksjon. Når man har skrevet inn adresse flyttes kartet til lokasjonen.

Figur 29

Som vist i figur 29a kan brukeren markere en lokasjon ved å trykke på kartet, og får da opp en markør med beskjed fár å sjekke værsiden. Dersom brukeren trykker her dukker værsiden på figur 27a opp, og viser værdata for den markerte lokasjonen. I tillegg kan brukeren søke på et sted slik vi ser i figur 29b og kartet forflytter seg slik vi ser i figur 29c.

6.5.1 Utfordringer i sprint 2

Underveis i sprint 2 møtte teamet på en utfordring med emutatoren. Denne fungerte ikke for master-branchen, og vi brukte en hel dag på å finne ut av hva som var feilen. Det så ut til å være ulike problemer på enhetene våre, så vi lekte litt rundt med ulike emulatorer, endret litt i build.gradle og sørget for at nøkkelen til *Google Maps* sitt API var korrekt. Til slutt fikk vi emulatoren til å fungere. Denne flaskehalsen forsinket arbeidet noe, men teamet klarte raskt å hente seg inn igjen.

En annen utfordring var knyttet til å tegne opp røde sirkler rundt flyplassene på kartet. Dette viste seg å være vanskeligere og mer tidskrevende enn forventet, og vi ble derfor ikke ferdige med denne oppgaven. Mer konkret dreide utfordringen seg om at det ikke lot seg gjøre å implementere verken *CircleOptions* eller *Geofence* (som er de to alternativene vi vurderte

til å tegne opp sirkler). Dette brukte vi en del tid på, og fant ikke ut av løsningen underveis.

6.5.2 Retrospekt sprint 2

Denne sprinten opplevdes som effektiv. Teamet fikk gjort mye, og sprinten hadde en fin varighet (1,5 uke). I denne sprinten var vi flinkere til å planlegge på forhånd, og dele inn oppgavene i små nok deler. Dette gjorde det lettere å komme i gang med arbeidet. Vi fikk markert flyplassene på kartet, fikk opp markør på valgt lokasjon, laget regelsiden, laget værsiden og lagt inn data her, laget en oversikt over hvordan vi skal implementere UU (universell utforming), lagt inn forslag i søkefunksjonen samt fått søkerbaren til å gå til adressen man skriver inn.

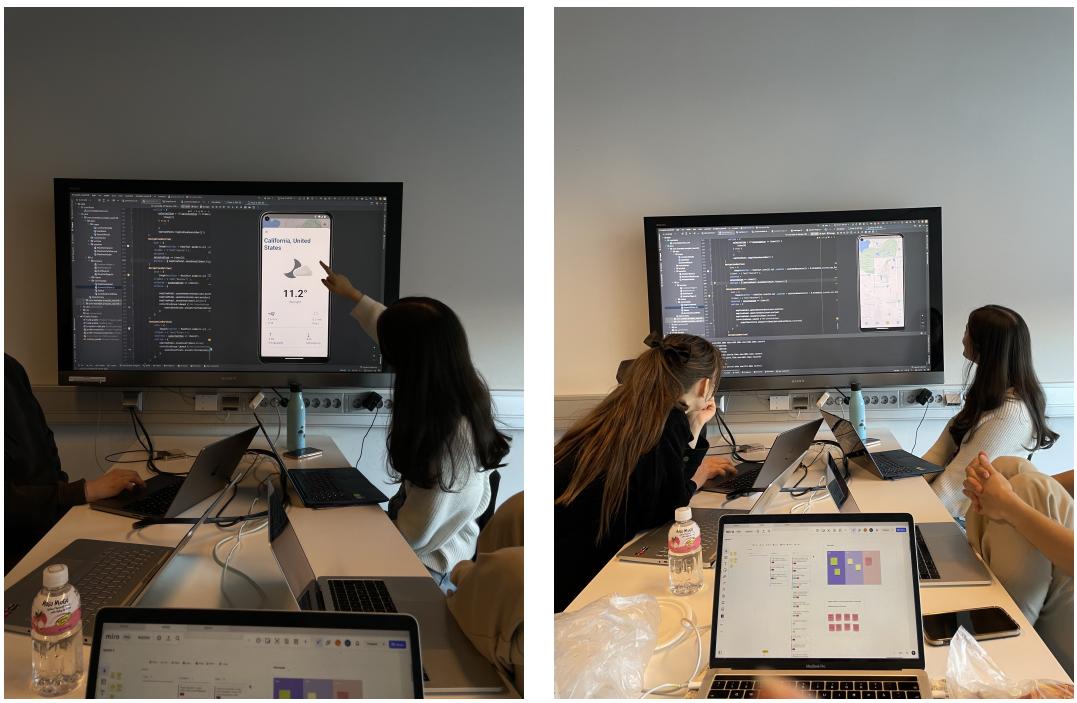
Som beskrevet tidligere endte vi opp med å legge alle flyplassene i en liste, og hardkode de inn på kartet. Dersom appen skulle blitt lansert burde vi egentlig ha brukt et eget API for å hente flyplassene, ettersom vi da ikke hadde trengt å oppdatere dataen dersom noen flyplasser blir lagt ned, eller nye flyplasser opprettes, men istedenfor kun hentet inn denne informasjonen automatisk.

Vi rakk ikke å bli ferdig med begrunnelseskappen som skal si noe om hvorfor man evt. ikke kan fly. Denne oppgaven, samt visning av de forbudte sonene, ble derfor videreført til neste sprint. Til neste sprint ønsket vi også å konkretisere oppgavene i backloggen slik at de ble mer håndterlige.

Tilbakemeldingene fra produkteier var positive.



Figur 30: Retrospekt sprint 2



Figur 31: Demonstrasjon av appen for gruppemedlemmene

6.5.3 Evaluering av kravspec

Etter råd fra veileder tok vi en evaluering av krevspecen etter sprint to. Vi valgte å spesifisere noen av kravene ytterligere, samt dele opp de ikke-funksjonelle kravene i produktkrav, organisatoriske krav og eksterne krav.

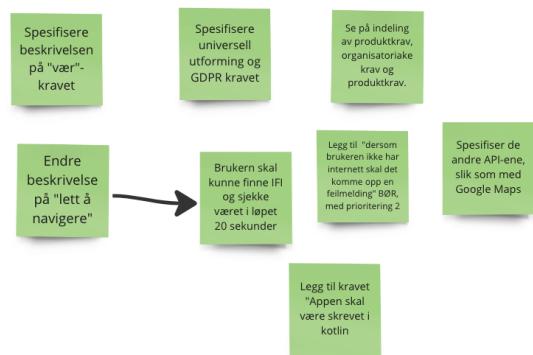
Basert på dette føyte vi kravspecen, slik at vi endte opp med kravene i figur 47, 48 og 49 i vedlegg *Oppdatert kravspec*. De fleste endringene lå hovedsakelig i de ikke-funksjonelle kravene, som nå var mer spesifikke og verifiserbare.

6.6 Sprint 3

Sprint 3 var preget av lite tid ettersom 4 av teammedlemmene ville få en hjemmeeksamen i IN2140 med oppstart 28. april. Sprint 3 varte derfor kun fra 25. - 28. april og vil beregnes som noe av de kortere sprintene man kan ha. Likevel ble kanban-brettet fylt med mange oppgaver som skulle håndteres i løpet av de tre dagene.

I sprint 3 ble det identifisert spesifikke deler av appen som skulle forbedres.

Evaluering av kravspec



Figur 32: Evaluering av kravspec

For det første gjaldt det begrunnelsessiden som angir om en bruker kan fly drone på en bestemt lokasjon. Designet for denne siden var allerede under utvikling, men funksjonaliteten manglet fortsatt implementering. Siden dette var en av hovedfunksjonene i appen, ble det ansett som viktig å fullføre denne funksjonaliteten.

Videre ønsket vi å fokusere på å tegne opp flyplassoner på kartet for å tydelig illustrere områder hvor det er forbudt å fly. Det var også nødvendig å sjekke om brukeren faktisk befant seg i en rød sone. Dette var avgjørende for funksjonaliteten til begrunnelsessiden, da siden ikke kunne gi godkjennelse til brukeren dersom vedkommende befant seg i en rød sone.

Det tredje vi ønsket å forbedre var å gjøre teksten i regelsiden mer tydelig, særlig som et bidrag for å gjøre det lettere for brukere som er mer svaksynte å lese teksten. Dette er en del av WCAG 2.0 (W3C, 2008) sitt krav 1.4.3 for universell utforming som omhandler kontrast, der det blant annet nevnes at man må ha stor nok skrift. Figur 33a og 33b viser appen før og etter denne endringen. Vi la også inn en referanse til Luftfartstilsynets nettside, slik at brukeren kan klikke seg videre for å lese mer om regelverket.

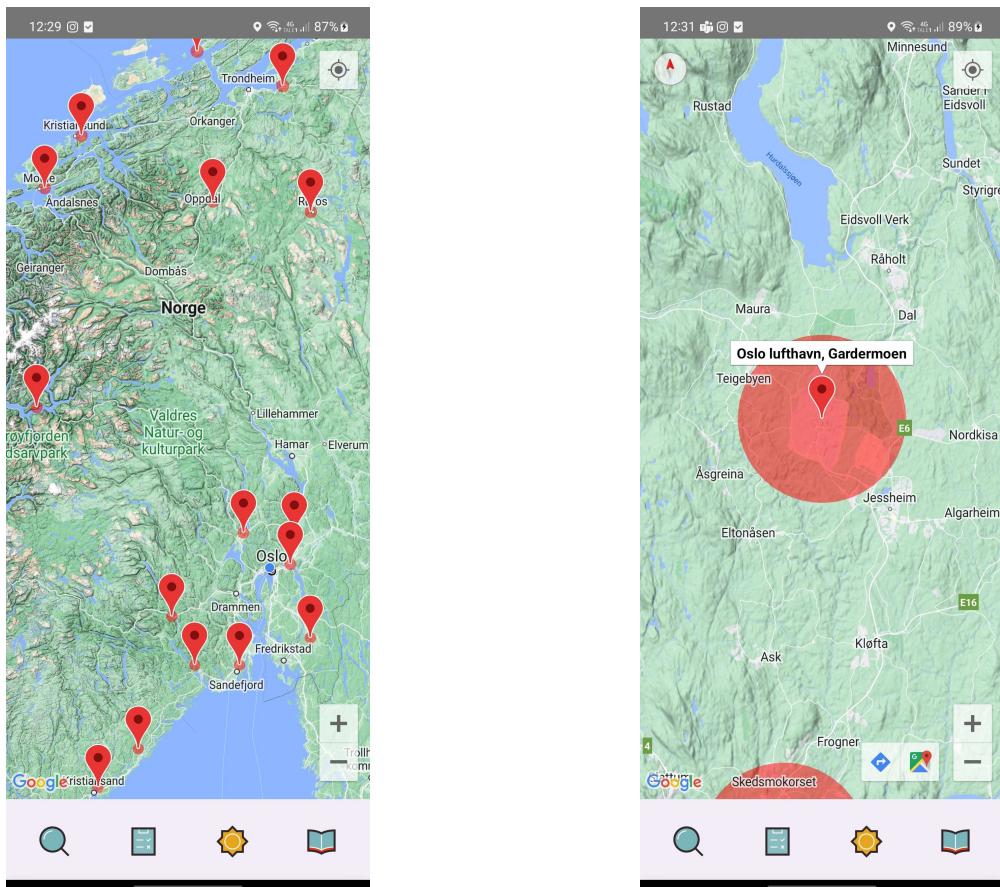


Figur 33

Arbeidet i sprinten ble gjennomført mye ved at vi satt sammen og jobbet på et grupperom. På denne måten kunne vi lett samarbeide om oppgaver, og vi opplevde dette som effektivt.

På grunn av dette løste vi blant annet utfordringen med å tegne opp flyplassoner rimelig kjapt, noe vi hadde problemer med tidligere i sprint 2. Hvordan vi skulle klare å sjekke om en bruker var innenfor en rød sone var også noe vi hadde lurt på og diskutert lenge. Vi oppdaget at vi kunne bruke en funksjon fra *Androids* bibliotek som sjekker to lokasjoner på kartet mot hverandre og kalkulerer om lokasjonene er en viss lengde unna hverandre. Dette var nøyaktig hva vi trengte siden en bruker ikke kan fly en drone under 5 km fra en flyplass, så da kunne vi sjekke brukerens lokasjon opp mot flyplasslokalasjonen, og sjekke om avstanden var mer enn 5km.

Vi valgte å tegne de røde sonene 5 km rundt pinnen til flyplassene. Resultatet av dette kan ses lenger nedenfor i figur 34a og 34b. Dette ble ikke nødvendigvis helt realistisk ettersom flyplassene kan strekke seg ut i lengde og bredde. I realiteten vil det derfor ikke nødvendigvis være sirkelformer rundt flyplassene. Å implementere forbudte soner nøyaktig 5 km fra alle kanter av alle flyplasser ble for vanskelig i praksis, så her måtte vi vise omtrentlig avstand.



Figur 34

En funksjonalitet vi implementerte i sprint 2 var å kunne sette markører på kartet. Dersom

brukeren trykket på markøren igjen ville det komme opp et info-vindu over markøren. Dette vinduet beskrev at man kunne klikke videre på info-vinduet for å sjekke værmeldingen på den lokasjonen. Vi konkluderte med at dette var litt mange steg - derfor sørget vi for at info-vinduet dukket opp automatisk når man lagde en markør. Dette gjorde det umiddelbart tydeligere for brukeren hva de kunne gjøre med markøren, og løste problemet vi opplevde med markørene i *Google Maps* med at det var utfordrende å trykke på markører.

Markørene i figur 34a og 34b er designet av *Google* og er en del av *Maps SDK-APlet*, på lik linje med kartet. Det betyr at disse to komponentene er helt kompatible med hverandre, så markørene vil blant annet holde samme størrelse på kartet uansett hvor mye man zoomer inn eller ut. I tillegg dukker det opp to ekstra ikoner nederst på kartet når man har trykket på en markør, slik man kan se i figur 33b. Den blå og hvite knappen til venstre åpner opp *Google Maps* og gir en veibeskrivelse fra brukerens posisjon til lokasjonen for markøren. Den andre knappen til høyre åpner også opp *Google Maps* med et vindu med informasjon om lokasjonen til markøren. Disse to knappene er irrelevante for appen vår, og er der kun på grunn av *Google* sine egne forhåndsbestemte innstillinger, som dessverre har få muligheter for tilpasning.

Videre i figur 35 kan man se den daværende begrunnelsessiden. Sidens design er identisk med designet etter sprint 2, men funksjonaliteten har blitt forbedret. Etter endringene sjekker den om de ulike kravene er oppnådd, og gir en samlet tilbakemelding basert på dette. Skjermdumpen er tatt etter at vi sjekket begrunnelse for en lokasjon ved Oslo Lufthavn, og vi kan se at brukeren derfor ikke oppfyller kravet om å befinner seg utenfor en rød sone.



Figur 35: Begrunnelsessiden med forbedret funksjonalitet

6.6.1 Utfordringer i sprint 3

Å tegne en rød sirkel rundt flyplassene var fortsatt en utfordring i sprint 3, men dette fikk vi løst til slutt. Det viste seg å være mye enklere enn først antatt, ettersom det eneste vi trengte var å lage et *Circle*-objekt ved hjelp av *Jetpack Compose*. Gruppen brukte derfor mye tid på å prøve å implementere *CircleOptions* og *Geofence* uten hell.

Helt på slutten av sprint 3 oppdaget vi en *bug*: dersom man markerte en pin på kartet kom begrunnelsessiden/vårsiden/regelsiden opp med én gang hvis man tidligere hadde trykket på den respektive siden. Denne buggen oppdaget og fikset vi raskt.

Dessuten fikk vi et lite problem med at begrunnelsessiden ikke oppdaterte seg med en gang man pinnet en ny lokasjon, så man måtte trykke på værmeldingen først for at det skulle fungere. Også denne feilen fikk vi løst svært raskt.

6.6.2 Retrospekt sprint 3

Denne sprinten var ganske kort, men til gjengjeld svært effektiv. Vi fikk laget ferdig de viktigste funksjonalitetene i appen, og det gjenstod bare litt småpirk. Teamet fikk gjort overraskende mye på kun noen dager. Vi opplevde at det var god kommunikasjon underveis, og vi jobbet godt sammen hele veien. Dette bidro til at sprinten ble effektiv. Det var også lett å spørre hverandre om hjelp, slik at alle kunne se på problemene som oppstod - på denne måten fikk vi løst de fleste utfordringene kjapt, og slapp å bruke lang tid på debugging.

Derimot var sprintens korte varighet også en utfordring, spesielt med tanke på sykdom og arbeidsmengde. Vi var litt for optimistiske med tanke på arbeidsmengden og det ble litt mye å gjøre. Det var flere oppgaver som stod igjen uferdige på slutten. Vi slet også litt med å komme i gang på grunn av den store arbeidsmengden. Dette opplevdes som litt demotiverende. Oppgavene som stod igjen som uferdige på slutten av sprinten omhandlet for det meste WCAG-krav som skulle implementeres, og vi ble enige om å beskrive disse ytterligere slik at det ble lettere å jobbe med disse i neste sprint. Vi ble også enige om å bli flinkere til å skrive på rapporten underveis i hver sprint, slik at vi ikke glemte hva som ble gjort når. Den fulle retrospeksen kan ses nedenfor i figur 36.



Figur 36: Retrospekt sprint 3

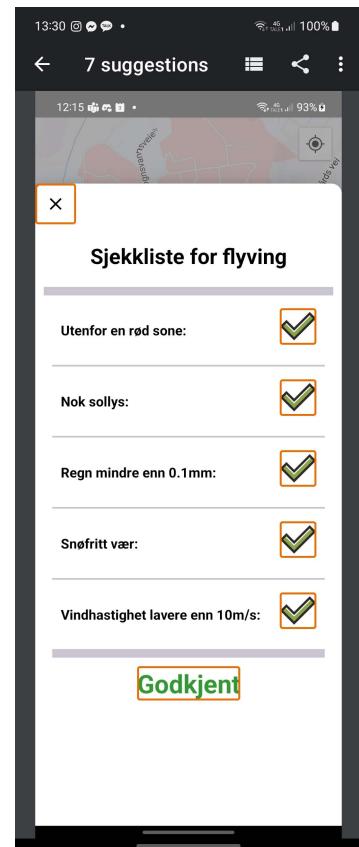
6.7 Sprint 4

I sprint 4 fokuserte vi hovedsaklig på rapportskriving, implementasjon av WCAG-krav, enhetstesting og litt debugging. Vi startet denne sprinten 2. mai, og ettersom fire av teammedlemmene hadde en annen hjemmeeksamen samtidig ønsket vi å begrense arbeidsmengden. Denne sprinten hadde en varighet på to uker, og vi jobbet hovedsaklig sammen i de oppsatte møtetidene (henholdsvis to timer på tirsdager, onsdager og fredager).

Underveis i denne sprinten merket vi at fokuset begynte å dale, og vi ble lett distraheret under møtene. Løsningen her var å prøve ut *Pomodoro* - en studieteknikk hvor vi jobbet effektivt i 20 minutter, for å deretter ha en 5 minutters pause. Da kunne vi dele inn møtene i ulike små sprinter der man jobbet fokusert i en større periode, etterfulgt av en kortere pause for å kunne snakke sammen eller gjøre noe annet. Vi opplevde dette som en effektiv teknikk, men samtidig hindret den oss litt i å samarbeide og kommunisere angående oppgaver i løpet av arbeidsintervallet. Det var derfor ikke en teknikk som passet for alle anledninger av prosjektarbeidet, men for visse situasjoner der alle på gruppen jobbet individuelt med skriving, modellering, o.l. passet den fint.

I starten av sprinten oppdaget vi appen *Accessibility Scanner* som er utviklet av *Google*, og som kan sjekke universell utformingskrav opp mot andre apper. Denne fungerer slik at man gir appen tillatelse til å kjøre over andre apper, og det vil da komme opp en blå knapp når man er inne i andre aplikasjoner. Dersom man trykker på den blå knappen vil man få muligheten til å ta et bilde av skjermen, og *Accessibility Scanner* vil da analysere applikasjonen og gi tilbakemelding på elementer som ikke oppfyller ett eller flere av UU-kravene og hva som må forbedres. I figur 37 kan man se resultatet da *Accessibility Scanner* analyserte den ene skjermen vår, og markerte elementene som burde forbedres. Dette gjorde det veldig enkelt for oss å analysere de ulike sidene av appen vår og rette opp i svakheter i designet fra et UU-perspektiv.

Da vi tidligere i prosjektet hadde uformelt analysert fargekontraster i applikasjonen vår ved hjelp av en nettside som kunne sammenlikne to farger, trodde vi at visse bilder ikke oppfylte UU-kravene om nok kontrast for lesbarhet. Da vi analyserte applikasjonen vår med *Accessibility Scanner* så vi derimot at dette ikke stemte, noe som gjorde at vi slapp å gjøre en del store endringer. Vi hadde blant annet diskutert å bytte bakgrunnsfarge på sidene i appen til en mørkeblå farge som ville skape tydeligere kontrast med bildene. Dette endte vi derimot opp med å ikke gjøre allikevel på bakgrunn av innsikten vi skaffet oss med *Accessibility Scanner*. Dette sparte oss for mye tid da vi hadde blitt nødt til å endre mange andre farger ellers i appen for å passe til den blå bakgrunnen, som skrift og ikoner.



Figur 37: Bruk av *Accessibility Scanner* for å teste UU-krav.

6.7.1 Utfordringer i sprint 4

En av de store utfordringene i sprint 4, og som vi hintet til tidligere i sprint 3, var at fire av gruppemedlemmene var preget av en stor hjemmeeksamen i IN2140. Flere av oss følte det derfor utfordrende å strukturere tiden, samt holde motivasjonen og entusiasmen for å jobbe med prosjektet oppe mens vi følte at det var andre ting vi heller ønsket å prioritere. Likevel opplevde vi at alle på gruppen respekterte arbeidsoppgavene de fikk og tok prosjektet seriøst når vi møttes for å jobbe.

6.7.2 Retrospekt sprint 4

I retrospeksen vi gjennomførte på slutten av sprinten kom det frem både positive og negative sider. Blant det positive ble det nevnt at det var fint at vi hadde klart å sette opp konkrete oppgaver for de ulike medlemmene som gjorde det lett å vite hva man måtte få gjort. Det var også positivt at vi møtte opp til møtene selv med en eksamen hengende over flere av oss. I tillegg var det fint at vi klarte å se enden av horisonten av prosjektet, og klarte å få gjennomført de fleste store oppgavene. Mye av det gjenstående arbeidet var mange mindre oppgaver som også var essensielle å få på plass før prosjektslutt.

Noen av de negative sidene ved sprinten som kom frem var at det var litt rotete arbeidsmessig ettersom vi som sagt gjensto med veldig mange små arbeidsoppgaver i mange sider av prosjektet. Dessverre var dette litt uunngåelig på slutten av prosjektet der det er mye ulikt vi måtte få ferdigstilt. I tillegg følte vi at denne sprinten hadde vært litt ambisiøs, selv om dette også var naturlig mot slutten. Vi følte at det var mindre samarbeid om oppgavene, og at vi hadde litt lav arbeidsmoral innad i gruppa. Som løsning planla vi å ha lange arbeidsdager sammen som gruppe den siste uka og jobbe mer sammen om visse oppgaver som oppleves som utfordrende og tunge. Vi bestemte oss også for å være flinkere med tydelige pauser i løpet av de siste møtene, og kanskje inkludere litt god mat og snacks for å løfte moralen.

6.8 Sprint 5

I vår aller siste sprint fokuserte vi på å ferdigstille appen og rapporten. Etter tilbakemelding fra veileder ryddet vi opp i strukturen på rapporten og gjorde ferdig alle diagrammer. En del tid gikk også til å korrekturlese rapporten, samt sjekke at vi hadde riktige referanser til bilder, vedlegg og kilder.

Vi utførte en brukertest og jobbet deretter med å fikse opp i tilbakemeldingene brukeren kom med. Se kapittel 6.11.2 for mer informasjon om hvordan vi utførte brukertesten.

Appen gjennomgikk noen siste endringer basert på tilbakemeldingene fra bruker. Disse endringene inkluderte å markere en pin med én gang brukeren søker etter en lokasjon, samt

en egen knapp for å veksle mellom egen lokasjon og pin på vær- og begrunnelsessiden. Vi la også inn appens logo og lagde en *splash-screen*. Søkebaren ble endret for å tilfredsstille UU-krav og kravet om at appen skal være på norsk. Mer spesifikt endret vi standard-teksten til 'Søk etter lokasjon' i stedet for 'Search' slik at vi oppfylte WCAG-prinsippet 'Forståelig' (Utilsynet, n.d.). Teamet la dessuten til en skygge rundt søkerbaren og navigasjonsbaren etter tilbakemelding fra bruker for å skape mer kontrast.

Teamet fikset også opp i rotasjonsproblemer, gjorde værsiden *scrollable* (slik at skjermen ikke blir beskjært ved bruk av f.eks. *Pixel 2*), og gjorde at appen fungerte for både sommertid og vintertid.

6.8.1 Utfordringer i sprint 5

Ettersom sprint 5 var svært kort (kun noen dager) var det begrenset med utfordringer i denne tidsperioden. Likevel møtte vi på noen problemer, blant annet utfordringer med kildehenvisning og bilderefanser i *Overleaf*. Dette ble løst nokså raskt, men vi skulle gjerne vært foruten.

Å fikse opp i tidssoner var en annen utfordring. Tidsforskyving pga. tidssone må sendes inn ved kall på *Sunrise API* fra MET, som brukes for å hente klokkeslett for soloppgang og -nedgang. Dette ble endret fra en hardkodet verdi til å hente ut tidssonen til brukerens mobil. Enhetstesten for henting av data fra API-et hjalp oss med å sjekke om appen fortsatt fungerte etter endringene. Alle testene passerte siden de kun var basert på norsk soldata. Derimot ble det ved kjøring av appen vist feil informasjon hvis bruker ville se data for andre tidssoner enn sin egen, og for å hente andre tidssoner krevdes bruk av et eget API. Vi valgte til slutt å ikke bruke mer tid på å fikse dette, da appen uansett er egnet til bruk av nordmenn i Norge. (*Sunrise API* fra MET).

For påliteligheten av appen forsøkte vi å rette opp i ulike situasjoner der appen kunne crashe. Vi opplevde blant annet at det å sjekke værmeldingen på visse lokasjoner var problematisk, samt å innhente stedsnavn til en lokasjon appen ikke klarte å innhente det fra. I begge tilfellene opplevde vi at appen crashet, men rettet senere opp i dette. Dette er evaluert ved å teste med markører på de samme lokasjonene som tidligere endte med en crash, som i Antarktis og nord på Grønland.

6.8.2 Retrospekt sprint 5

Sprint 5 ble avsluttet samme dag som prosjektet skulle leveres. Vi var opptatte med å ferdigstille produktet, og hadde derfor ikke et retrospekt-ritual for denne sprinten.

6.9 GDPR

Personvernforordningen, også kalt GDPR, er et regelverk for behandling av persondata som er utviklet av EU, og gjelder for alle land i EU og EØS (Lovdata, 2023). Dette er derfor noe vi som utvikler av en applikasjon som andre skal bruke må forholde oss til dersom vi samler inn personopplysninger fra brukeren. For vår del var det derimot ikke dette så relevant for utviklingen av appen, ettersom vi ikke ønsket å samle inn noe data om spesifikke brukere. Appen etterspør kun tilgang til GPS-lokasjoner mens appen er i bruk via Google sitt rammeverk for å kunne vite hvor på kartet brukeren befinner seg. Dette er derimot noe brukeren må godkjenne ved nedlasting av applikasjonen, og er informasjon som kun er til nytte for funksjonaliteten av appen mens den er i bruk. Det er likevel verdt å nevne at vi ikke vet hvorvidt Google lagrer dataen eller hva de gjør med den.

6.10 Universell utforming

For å avgjøre hvilke av reglene for universell utforming vi skulle benytte brukte vi UU-tilsynet sin oversikt over de kravene i WCAG 2.0 som er mest aktuelle for apper (Utilsynet, 2023). Vi valgte å fokusere på noen av dem.

Tabellen som viser de utvalgte kravene som ble jobbet med og deres oppfyllelse finnes i figur 55 i vedlegg *Universell utforming*.

Vi jobbet med disse kravene for å oppfylle våre ikke-funksjonelle krav *Universelt utformet*, se figur 49 i *Vedlegg*. Grunnen til at akkurat disse kravene ble valgt var fordi de var mest relevante og overkommelige å implementere.

6.11 Testing

For å kontrollere både appens funksjonalitet og bruken av appen utførte vi enhetstester og en brukertest.

6.11.1 Enhetstester

Hensikt

Enhetstesting innebærer å teste mindre, isolerte komponenter av programmet, for å se at de fungerer som forventet. Fordelene med enhetstesting er at man under prosessens gang lettere kan oppdage feil i enkelte deler av koden, som ellers kunne føre til potensielt større problemer senere. Siden isolerte komponenter testes, kan testingen skje med en gang en komponent er ferdig skrevet. Derfor valgte vi å skrive enhetstester under prosessen. (Zaptest, n.d.)

Hver enhetstest består av tre deler: *Arrange*, som har ansvar for data som trengs for testen, *Act*, hvor funksjonskall/manipulering av data for testen skjer, og *Assert*, hvor man sjekker om resultatet ble som forventet.

Gjennomføring

For å skrive enhetstestene brukte vi *JUnit 4*, som er et rammeverk for testing av *Java*-klasser men som også fungerer for *Kotlin* (Zaptest, n.d.). Vi lagde en enhetstest for klassen *FeedbackCheck*, som er ansvarlig for å beregne flyvetillatelsen som vises i begrunnelsesiden.

Klassen inneholder en funksjon for hvert flyvekrav, og hver funksjon returnerer *true* eller *false* avhengig av om kravet er oppfylt eller ikke. Derfor skrev vi to tester for hver funksjon - én hvor kravet er oppfylt, og én hvor kravet ikke er oppfylt.

Figur 38 viser et eksempel på én av testene våre.

```
@Test
@Felicia
fun testOkRainTrue() {
    // Arrange
    val feedbackCheck = FeedbackCheck()

    val weatherModel = WeatherModel(
        date = Date(),
        temperature = 6.0,
        summaryCode = "",
        summaryNextHour = "",
        rainNextHour = 0.0, // <- value for check
        windDirection = 0.0,
        windSpeed = 0.0
    )

    // Act
    val okRain = feedbackCheck.okRain(weatherModel)

    // Assert
    assert(okRain)
}
```

Figur 38: En av enhetstestene våre. Her ser man hvordan *Arrange*, *Act* og *Assert* er benyttet.

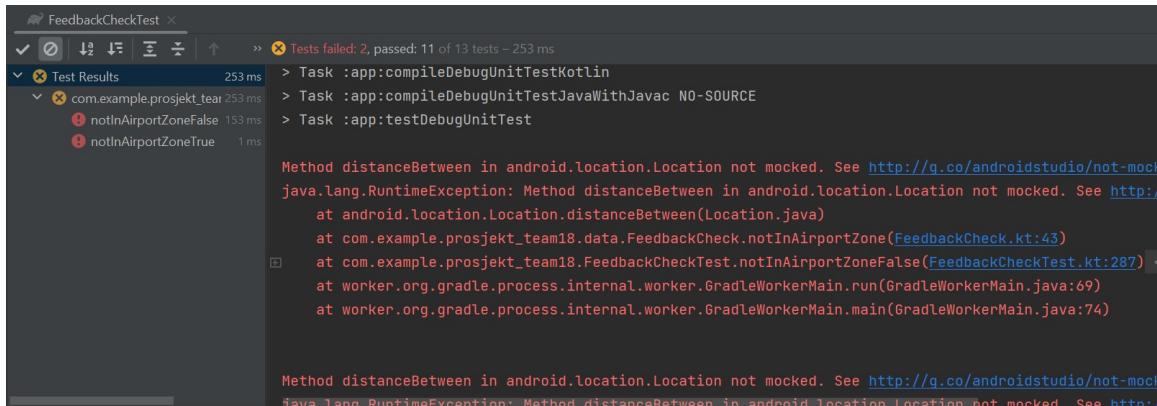
testOkRainTrue() i figur 38 skal teste at funksjonen *okRain* returnerer *true* dersom det ikke er for mye regn ute til å fly drone. I *arrange*-delen lager vi testdata til å sende inn i funksjonen, som i dette tilfellet er en dataklasse for værdata hvor mengden regn er satt til 0,0. I *act*-delen kaller vi på funksjonen *okRain* med dataklassen som argument, og i *assert* sjekker vi at *okRain*-funksjonen returnerte *true* som forventet.

Til slutt inneholder klassen *FeedbackCheck* en funksjon som tar inn resultatet til disse funksjonene som argument, og beregner flyvetillatelsen. Derfor skrev vi også to tester for denne,

en hvor flyving skal være tillatt (alle argumenter er *true*), og en hvor flyving ikke skal være tillatt (minst ett av argumentene er *false*).

Kjøring av testene

Vi skrev alle testene i en klasse *FeedbackCheckTest*, som vi puttet i mappen *test* som er tildegnat enhetstester. Ved kjøring passerte alle tester unntatt de to som testet om en lokasjon var innenfor en flyplasssone eller ikke. Disse to testene ga feilmeldingen som vises på figur 39 og 40:



```
FeedbackCheckTest
  Tests failed: 2, passed: 11 of 13 tests – 253 ms
    Test Results      253 ms
      com.example.prosjekt_team18 253 ms
        > Task :app:compileDebugUnitTestKotlin
        > Task :app:compileDebugUnitTestJavaWithJavac NO-SOURCE
        > Task :app:testDebugUnitTest

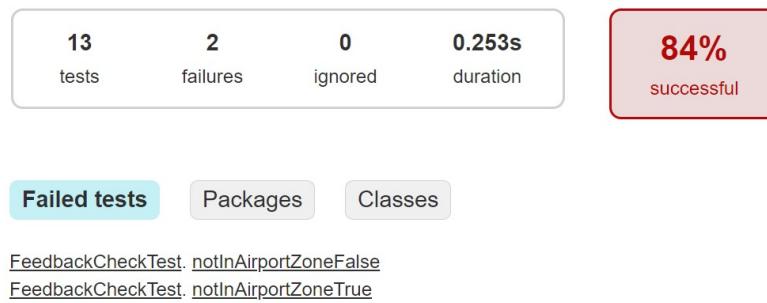
        Method distanceBetween in android.location.Location not mocked. See http://g.co/androidstudio/not-mock
        java.lang.RuntimeException: Method distanceBetween in android.location.Location not mocked. See http://g.co/androidstudio/not-mock
          at android.location.Location.distanceBetween(Location.java)
          at com.example.prosjekt_team18.data.FeedbackCheck.notInAirportZone(FeedbackCheck.kt:43)
        at com.example.prosjekt_team18.FeedbackCheckTest.notInAirportZoneFalse(FeedbackCheckTest.kt:287) <
          at worker.org.gradle.process.internal.worker.GradleWorkerMain.run(GradleWorkerMain.java:69)
          at worker.org.gradle.process.internal.worker.GradleWorkerMain.main(GradleWorkerMain.java:74)

        Method distanceBetween in android.location.Location not mocked. See http://g.co/androidstudio/not-mock
        java.lang.RuntimeException: Method distanceBetween in android.location.Location not mocked. See http://g.co/androidstudio/not-mock
```

Figur 39: Feilmelding etter å ha testet om en lokasjon var innenfor en rød sone.

Kjøring av testene

Test Summary



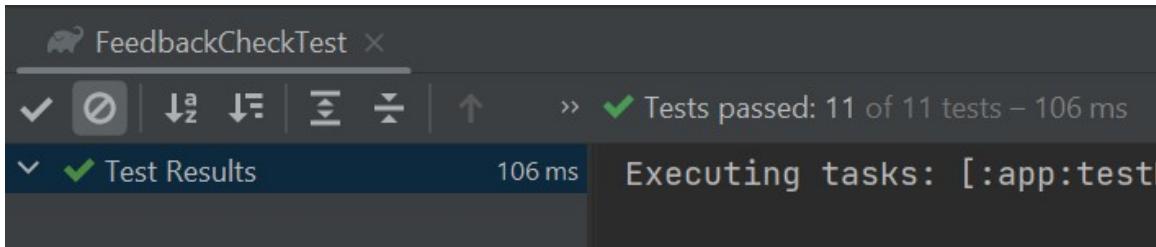
Figur 40: Generert testrapport fra Gradle i Android Studio.

Det var en utfordring å løse dette problemet, da vi ikke fant noe på nettet som kunne hjelpe oss å forstå feilmeldingen. Til slutt klarte vi likevel å løse dette problemet, og løsningen var å flytte de to funksjonene til en egen testklasse, *AirportTest*, som vi puttet i mappen *androidTest* istedenfor *test*. Løsningen vises i figur 41.

Det viste seg at feilmeldingen kom av at funksjonen kalte på *distanceBetween*-funksjonen, som er en funksjon innebygd i *Android* sitt bibliotek. Alle tester som ligger i *test*-mappen kjøres som vanlige *Java*- eller *Kotlin*-programmer, uten å starte *Android* emulatoren, og derfor kunne ikke denne funksjonen aksesseres. Etter å ha flyttet de to funksjonene til *androidTest*-mappen kjørte de uten feilmelding (se figur 42).

Tests	Duration	Pixel_5_API_29
Test Results	106 ms	1/1
AirportTest	106 ms	1/1
notInAirportZoneTrue	106 ms	✓

Figur 41: Løsningen var å flytte funksjonene til en egen testklasse.



Figur 42: Screenshot av feedbacktesten gjort i Android Studio

6.11.2 Brukertester

I tillegg til enhetstestene utførte vi også en brukertest, hvor målet var å teste hva *softwaren* gjør og hvordan brukeren interagerer med systemet, fremfor å teste om det finnes feil i koden (Somerville, 2021, s.270). Vi gjennomførte brukbarhetstest og brukergrensesnitttest.

I brukbarhetstesten (Somerville, 2021, s.271) ga vi brukeren en rekke oppgaver som skulle utføres. Se figur 53 i vedlegg *Brukertest* for oversikt over oppgavene.

Vi kjørte en *User Interface test* for å sjekke om brukeren likte utseendet og systemets brukergrensesnitt (Somerville, 2021, s.271). Etter gjennomført brukbarhetstest hadde vi et kort, ustrukturert intervju hvor vi spurte om hva brukeres syntes om bruken av appen.

Basert på brukertesten og tilbakemeldinger fikk appen positive vurderinger. Brukeren fant den funksjonell og tiltalende, og vedkommende likte spesielt det enkle og oversiktlige designet som ikke overveldet dem med unødvendige elementer. For å forbedre brukervennligheten ble det foreslått å plassere markøren umiddelbart når man søker etter en lokasjon, liknende tilnærmingen i *Google Maps*. Videre ønsket brukeren en tydeligere 'Finn-min-plassering'-knapp for å enkelt kunne gå tilbake til sin egen posisjon. Enkelte forslag ble også fremsatt angående ikonbruken, som å erstatte det eksisterende ikonet med et hakemerke og endre ikonfarge basert på brukerens evne til å fly. Til slutt ble det påpekt et behov for økt kontrast og tydelighet i søkefeltet og søker-ikonet. Samlet sett var brukeren fornøyd med at appen oppfylte sitt formål uten unødvendig støy eller kompleksitet. Fullstendig tilbakemelding ligger i figur 54.

Vi tok med oss disse tilbakemeldingene videre og fikset noen av problemene, som nevnt i kapittel 6.8 som omhandlet Sprint 5.

Brukeren var en venn, så observasjonen var preget av noe bias (Joshi, 2020b, s.43). Dette er noe vi kunne unngått ved å utføre testen på noen vi ikke kjente fra før av. Likevel følte vi at brukeren vår var ærlig, og nølte ikke med å komme med forslag til forbedringer.

Grunnet kort tid rakk vi kun å gjennomføre brukertesten på én bruker. Det hadde likevel vært interessant å få snakket med flere for å avdekke andre potensielle forbedringsområder. Dersom vi skulle jobbet videre med appen hadde det også vært interessant å kjøre en beta-test, hvor man gir en begrenset versjon av appen til en gruppe eksterne brukere for å få tilbakemeldinger (Sommerville, 2021, s.271). På den måten kunne vi samlet inn verdifull data om brukbarheten til appen fra flere brukere.

7 Refleksjon

Ettersom dette var et omfattende prosjekt var det en del temaer å reflektere over. Vi valgte å hovedsakelig reflektere over planlegging, verktøy, møtetyper, ekspertintervju, designprosessen, smidig metode, rollefordeling, samt hvorvidt vi oppfylte kravene fra kravspesifikasjonen.

7.1 Planlegging

I teamet vårt var vi svært ivrige etter å komme i gang med programmeringen, og dette førte til at planleggingsfasen ble nedprioritert. Vi hoppet raskt til programmeringsfasen allerede i sprint 1. Selv om dette ga oss muligheten til å starte utviklingsarbeidet tidlig og bidro til en rask fremdrift i prosjektet.

Ved å hoppe rett inn i programmeringen fikk vi raskt muligheten til å teste ut funksjonaliteten

og motta tidlige tilbakemeldinger om hva som fungerte og hva som ikke gjorde det. Dette var nyttig for å identifisere eventuelle problemer eller mangler på et tidlig stadium av prosessen.

Likevel var det noen ulemper ved å ha begrenset planlegging og design. Risikoen for omfattende endringer senere i prosjektet økte, noe som kunne medføre ekstra arbeid og tidstap. Mangelen på en helhetlig visjon kunne også føre til inkonsistens i designet og brukeropplevelsen.

En annen ulempe var mangelen på fokus på universell utforming i begynnelsen av prosjektet. Dette økte risikoen for betydelige endringer senere for å sikre at appen var tilgjengelig og brukervennlig for alle brukere.

I ettertid innser vi at vi burde ha brukt mer tid på å planlegge funksjonaliteter og design før vi startet programmeringen. Ved å ta hensyn til dette underveis, opplevde vi at designet ble tilpasset programmeringen i stedet for motsatt. Det ble også vanskeligere å tydelig avgrense når vi var ferdige med programmeringen.

For fremtidige prosjekter ser vi verdien av å investere mer tid i planlegging av funksjonalitet og design. Ved å ha en solid plan på plass på forhånd kan vi unngå unødvendige endringer senere i prosjektet og sikre en mer helhetlig og brukervennlig app.

7.2 Mirobrettet

Vårt bruk av et Miro-brett har vært svært verdifullt i prosessen med å utvikle droneappen vår. Vi har valgt å presentere informasjonen på en visuell måte, noe som har gjort det enkelt for produkteieren å få innsikt i prosessen og forstå utviklingen på et overordnet nivå.

Miro-brettet har gitt oss muligheten til å organisere og visualisere ideer, krav og funksjoner på en strukturert og intuitiv måte. Vi har kunnet opprette digitale kanban-brett, legge til notater, bilder, visualisere sprinter og andre visuelle elementer for å representere ulike aspekter av appen vår.

Videre har Miro-brettet også tilrettelagt for bedre samarbeid og kommunikasjon i teamet. Vi har kunnet jobbe sammen i sanntid, dele ideer og kommentarer, og opprettholde et oppdatert og felles syn på prosjektet. Den visuelle fremstillingen har også gjort det enklere å formidle konsepter og sørge for at alle er på samme side.

Totalt sett har bruken av Miro-brettet vært svært positivt for prosessen vår. Det har bidratt til et mer engasjerende og effektivt samarbeidsmiljø. Den visuelle presentasjonen har gitt klarhet og oversikt, og vi har kunnet visualisere vår fremgang på en måte som alle kan forstå.

7.3 Ustrukturerte og uformelle møter

I prosjektet vårt har vi erfart at vi har hatt en del ustrukturerte og uformelle møter. Til tross for denne uformelle tilnærmingen, har vi likevel klart å holde oss til den smidige metoden, slik som å ha *daily standups*. Vi har også ført møtereferat til hvert møte. Dette kan tyde på at vårt team har hatt en naturlig arbeidsflyt og har kunnet tilpasses seg i tråd med de smidige prinsippene. Den uformelle tilnærmingen kan ha bidratt til å skape et miljø der alle teammedlemmer føler seg komfortable med å dele ideer og diskutere utfordringer.

Imidlertid fikk vi et tips fra veilederen vår om å introdusere mer strukturerte møter i sprint 3. Dette forslaget ble presentert for å gi oss muligheten til å utforske hvordan en mer strukturert tilnærming kunne påvirke arbeidsprosessen og teamets samarbeid. Ved å innføre mer strukturerte møter kunne vi oppnå flere fordeler. For det første hjalp det oss med å være mer fokuserte og effektive i diskusjoner og beslutningstaking. Videre bidro det til bedre dokumentasjon og oppfølging av avtaler og oppgaver.

Konklusjonen er at vårt team hadde suksess med en uformell tilnærming til møter i prosjektet. Likevel var vi åpne for å utforske og erfare en mer strukturert metode for å se om det kunne gi oss ytterligere fordeler. Ved å eksperimentere med ulike tilnærmingar kunne vi videreutvikle vårt teamarbeid og optimalisere vår produktivitet i samsvar med våre prosjektmål.

7.4 Møtene underveis i prosjektet

Underveis i prosjektet har teamet hatt en rekke ulike møter. Dette inkluderer sprintplanleggingsmøte, *daily standups*, sprintslutt, *sprint review*, samjobbing og skrivemøter. Vi opplevde at det å ha mange møter var nyttig for å jobbe jevnt med prosjektet. Som tidligere nevnt var noen av møtene ustrukturerte og litt uformelle, men vi fikk likevel gjort det vi skulle og regnet derfor ikke dette som en hindring. Spesielt sprintplanleggingsmøtene og *sprint reviews* fungerte veldig godt ettersom alle hadde fokus og bidro til møtene. *Daily standups* var vi gode på i starten, men begynte etterhvert å glemme det litt av. Likevel hadde teamet god kommunikasjon underveis, slik at det var tydelig hva alle jobbet med og hvorvidt noen trengte hjelp. Dermed var det ikke katastrofalt at vi glemte *daily standups* innimellan.

Størsteparten av møtene våre gikk til samjobbing. Dette fungerte svært godt for oss, ettersom vi fikk mulighet til å løse utfordringer raskt med hjelp av de andre i teamet. Ved å ha mange møter sørget vi for at vi stadig var koordinerte og visste hvordan alle lå an.

Teamet valgte å også ha egne skrivemøter når det trengtes. Dette var for å sørge for at vi lå godt an på rapporten, og fikk med oss den viktigste informasjonen fra prosessen underveis.

Dette fungerte veldig godt ettersom vi kunne skrive på rapporten sammen, og dermed diskutere og stille spørsmål underveis slik at vi ble enige om innholdet. Vi opplevde mange skrivemøter som mer effektive enn om vi skulle skrevet på rapporten individuelt. Riktignok ble noen av skrivemøtene mot slutten av prosessen preget av hjemmeeksamen i IN2140, og teammedlemmene følte seg til tider slitne. Til tross for dette fikk vi likevel gjort det vi skulle, og skrivemøtene fungerte generelt godt.

Teamet har vært flinke til å møte opp, og eventuelt bli med digitalt over *Discord* dersom noen ikke hadde mulighet til å stille opp fysisk.

7.5 Ekspertintervju

Valget om å ikke inkludere ekspertintervjuer i utviklingen av vår droneapp er et resultat av vår prioritering av brukerintervjuer som primær kilde for informasjon. Vi ønsket å legge vekt på brukernes perspektiv og behov for å sikre at appen ble brukervennlig og funksjonell. Ved å samle innsikt fra brukerne direkte har vi kunnet ta hensyn til deres opplevelser og utfordringer, noe som har vært avgjørende for å utforme en app som tilfredsstiller deres krav.

Selv om eksperter kunne ha bidratt med spesialisert kunnskap og innsikt på områder som brukerne kanskje ikke er klar over, har vi valgt å støtte oss på informasjonen fra Luftfartstilsynet som en troverdig kilde. Luftfartstilsynet er ansvarlig for droneregler og -forskrifter, og deres informasjon er autoritativ og oppdatert. Ved å benytte oss av denne pålitelige veiledningen, har vi sikret at appen vår er i samsvar med gjeldende lover og regler for dronekjøring.

En annen viktig faktor i vår beslutning er ressursbegrensninger. Engasjering av eksterne eksperter kan være tidkrevende og kostbart. Ettersom vi allerede hadde tilgang til pålitelig informasjon fra Luftfartstilsynet, kunne vi unngå ekspertintervjuer og fokusere våre ressurser på å utvikle en brukervennlig app basert på den eksisterende kunnskapen vi hadde.

7.6 Designprosessen

I designprosessen for utviklingen av droneappen valgte vi å fokusere på å forstå brukerbehovene gjennom brukerintervjuer. Dette tillot oss å identifisere potensielle brukere, deres krav og preferanser. Ved å lytte til deres perspektiver og tilbakemeldinger kunne vi danne et solid grunnlag for å utvikle appen med brukerens behov i tankene. Dette var et viktig første skritt for å sikre at appen ville være relevant og nyttig for målgruppen.

Basert på brukerbehovene utarbeidet vi en kravspesifikasjon som definerte de viktigste funksjonalitetene og egenskapene til droneappen. Dette bidro til å skape klarhet rundt hva som

skulle implementeres, inkludert funksjoner som kartvisning, informasjon om restriksjoner for droneflyging, tillatelser og muligheten for brukere å rapportere ulovlige droneaktiviteter. Kravspesifikasjonen fungerte som et referansepunkt for å sikre at appen oppfylte brukernes forventninger og behov.

Et aspekt som vi kunne ha forbedret i designprosessen var bruk av skisser eller *wireframes* for å visualisere layouten og strukturen til appen. Dette ville ha gitt oss muligheten til å eksperimentere med ulike designalternativer og motta tilbakemelding fra både teammedlemmer og potensielle brukere. Ved å involvere brukere tidligere i designprosessen, ville vi ha hatt en bedre mulighet til å identifisere eventuelle forbedringsområder og tilpasse designet for å møte deres behov og preferanser.

Et annen aspekt som kunne vært bedre håndtert var utviklingen av det visuelle designet for appen. Vi kunne tatt hensyn til appens formål for å utvikle en visuell identitet som støtter brukeropplevelsen og funksjonaliteten. Dette inkluderer valg av fargepalett, typografi og grafiske elementer som ville skape en helhetlig og engasjerende opplevelse for brukerne.

Vi erkjenner også at vi ikke gjennomførte brukertestning og iterasjon i tilstrekkelig grad i designprosessen. Med kun én runde intervjuer og en avsluttende test gikk vi glipp av muligheten til å oppdage flere potensielle problemer eller utfordringer som brukerne kunne støte på. Ved å jobbe mer iterativt og involvere brukere gjennom ulike faser av designprosessen, ville vi ha hatt muligheten til å kontinuerlig forbedre appens brukervennlighet og funksjonalitet basert på deres tilbakemeldinger.

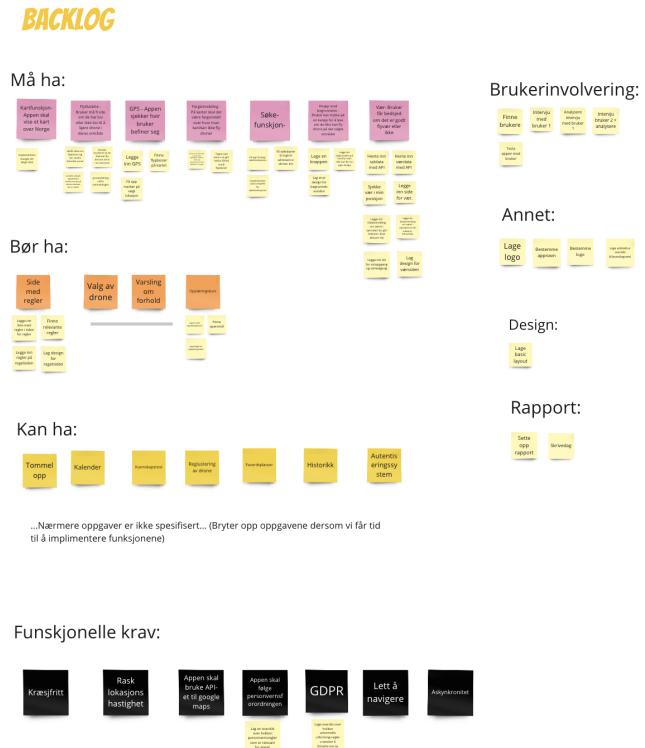
Samlet sett har designprosessen for droneappen vært en lærerik opplevelse. Vi har lykkes med å forstå brukerbehovene gjennom brukerintervjuer og definert viktige funksjonaliteter gjennom en kravspesifikasjon. Likevel er det områder vi kunne forbedret, men vi tar med oss lærdommen videre.

7.7 Smidig metode

Dersom det oppsto oppgaver underveis som vi innså måtte gjøres, var vi flinke til å motstå fristelsen for å legge dem til i pågående sprint. Vi forstod viktigheten av å opprettholde sprintens fokus og ikke legge til ekstra oppgaver underveis som kunne påvirke vår evne til å fullføre sprintmålene.

Bruken av kravspesifikasjonen som utgangspunkt for backlogen ga oss en klar oversikt over funksjonaliteter og krav som skulle implementeres. Dette gjorde det mulig for oss å prioritere oppgaver og arbeide med dem i henhold til sprintene. Imidlertid oppdaget vi at det var utfordrende å plukke oppgaver som var tilstrekkelig oppdelt. Noen oppgaver var for store og komplekse til å bli fullført innenfor en sprint. For å håndtere dette utfordrende aspektet

eksperimenterte vi med å lage en mer detaljert backlog. Målet var å dele funksjonaliteten inn i mindre oppgaver på forhånd, slik at det skulle bli enklere å plukke oppgaver ved sprintstart. Disse oppgavene var basert på kravene i kravspesifikasjonen. Dessverre ble den mer detaljerte backlogen ikke effektivt brukt da den ble introdusert sent i prosessen. For ytterligere illustrasjon, se figur 43.



Figur 43: Backlog hentet fra Miro-brettet

Det var en utfordring å finne riktig mengde oppgaver å jobbe med. På grunn av den begrensete tiden og det omfattende arbeidet vi hadde foran oss, var vi bekymret for å legge til for få oppgaver. Som et resultat endte vi ofte opp med å ta på oss for mange oppgaver, noe som kunne føles demotiverende når vi så på de gjenværende oppgavene på kanban-brettet. Ettersom vi hadde muligheten til å legge tilbake oppgavene i backlogen hvis de ikke ble fullført i løpet av sprinten, føltes det tryggere å velge for mange oppgaver enn for få. Vi var redde for å ikke bli ferdig i tide og ønsket derfor å være på den sikre siden.

Bruken av smidige metoder har vært svært lærerikt for oss som team. Det har ikke bare bidratt til å forbedre samarbeidet og kommunikasjonen mellom teammedlemmene, men også skapt en plattform for jevnlige møter der vi kunne diskutere fremdriften, utfordringene og planleggingen i prosjektet. Ved å bruke smidige metoder har vi oppdaget nye tilnærminger, verktøy og teknikker som har beriket vår forståelse av prosjektstyring og utvikling. Vi har også lært verdien av regelmessig refleksjon, evaluering og tilpasning for å sikre kontinuerlig vekst for å nå målene.

7.8 Rollefordelingen

I oppstartsfasen ble teamet enige om en rollefordeling som kunne fungere for oss. I praksis har noen av rollene holdt seg gjennom prosjektet, mens andre har blitt mindre tydelige. For eksempel har Rapportansvarlig underveis blitt en rolle som alle har inntatt, og Kontaktperson ble aldri satt. Scrum master har heller ikke vært en tydelig rolle vi har holdt oss til. Dette hindret oss likevel ikke i å fullføre prosjektet. I oppstartsfasen nevnte vi at det ville være naturlig at enkelte roller har overlapp, og at vi stadig vil ha tett samarbeid slik at alle hjelper til når det trengs. Dette er et prinsipp vi har holdt oss til i veldig stor grad, og muligens noe av grunnen til at enkelte roller ble mindre viktige for oss. Tett samarbeid kan ha ført til at flere teammedlemmer inntok ulike roller uten å egentlig være klar over det.

7.9 Oppfylte vi kravene fra kravspesifikasjonen?

De viktigste funksjonelle kravene, altså MÅ-kravene våre (figur 47 i vedlegg *Oppdatert kravspesifikasjon*), ble oppfylt. Dette inkluderer kartfunksjon, knapp med begrunnelse, GPS, søkefunksjon, værinformasjon og fargeinndeling. I tillegg fikk vi oppfylt ett BØR-krav (figur 48 i vedlegg *Oppdatert kravspesifikasjon*), nemlig regelsiden. Dersom vi hadde hatt mer tid til å jobbe med prosjektet er det mulig at vi hadde fullført flere krav, men samtidig er den viktigste funksjonaliteten på plass, og vi har oppfylt vår visjon.

Vi oppfylte dessuten alle de ikke-funksjonelle kravene (figur 49 i vedlegg *Oppdatert kravspesifikasjon*). Dette gjør at vi totalt sett er fornøyde med resultatet.

7.10 Oppsummering

Å jobbe med dette prosjektet har vært både spennende og utfordrende. Vi synes det var både gøy og nyttig å jobbe sammen i et team, og følge smidig metodikk. Dette er noe vi vil ta med oss videre ut i arbeidslivet, da dette prosjektet er svært relevant for fremtidig arbeid.

Det har vært lærerikt å lære mer om *Kotlin* og bruke ulike API-er, samt kjenne på viktigheten av god kommunikasjon innad i teamet. Vi har opplevd teamarbeidet som effektivt, og har vært flinke til å hjelpe hverandre underveis. Selve arbeidet har også til syvende og sist gått bra, til tross for ulike utfordringer vi hadde på veien. Disse var spesielt knyttet til tekniske utfordringer og hjemmeeksamen i et annet emne som minimerte arbeidsmengden i dette prosjektet. Likevel har vi fått gjort det viktigste vi ønsket å gjøre, og er stolte av produktet vårt.

Referanser

- AndroidDevelopers. (2023). *Recommendations for Android architecture*. <https://developer.android.com/topic/architecture/recommendations> (accessed: 10.05.2023)
- AndroidStudio. (2023a). *State and Jetpack Compose*. <https://developer.android.com/jetpack/compose/state> (accessed: 26.05.2023)
- AndroidStudio. (2023b). *Thinking in Compose*. <https://developer.android.com/jetpack/compose/mental-model> (accessed: 26.05.2023)
- AndroidStudio. (n.d.). *Build better apps faster with Jetpack Compose*. https://developer.android.com/jetpack/compose?gclid=CjwKCAjwscGjBhAXEiwAswQqNJy_oNjPAwPG1UPmkxzbBwE&gclsrc=aw.ds (accessed: 26.05.2023)
- Developers, A. (2023). *Meet Google Play's target API level requirement*. <https://developer.android.com/google/play/requirements/target-sdk> (accessed: 23.05.2023)
- Geonorge. (n.d.). *Lufthavner Flyplasser*. <https://kartkatalog.geonorge.no/metadata/lufthavner-flyplasser/8de1886c-2c54-4f1a-a527-566987ae7fdb> (accessed: 12.04.2023)
- Joshi, S. G. (2020a). *Design, prototyping og konstruksjon*. https://www.uio.no/studier/emner/matnat/ifi/IN1050/h20/forelesning_200922.pdf (accessed: 26.05.2023)
- Joshi, S. G. (2020b). *Kvalitativ analyse*. https://www.uio.no/studier/emner/matnat/ifi/IN1050/h20/forelesning_200908.pdf (accessed: 26.05.2023)
- Lovdata. (2023). *Lov om behandling av personopplysninger (personopplysningsloven)*. https://lovdata.no/dokument/NL/lov/2018-06-15-38/KAPITTEL_1#%C2%A71 (accessed: 26.05.2023)
- Luftfartstilsynet. (2022). *Fly drone trygt*. <https://luftfartstilsynet.no/droner/fly-drone-trygt/> (accessed: 12.04.2023)
- Luftfartstilsynet. (2023a). *Fly drone trygt*. <https://luftfartstilsynet.no/droner/fly-drone-trygt/> (accessed: 12.04.2023)
- Luftfartstilsynet. (2023b). *Kan jeg fly når det er mørkt?* <https://luftfartstilsynet.no/droner/faq---droner---samlete-sporsmal-sporsmal-1/kan-jeg-fly-nar-det-er-morkt/> (accessed: 12.04.2023)
- MicrosoftLearn. (2022). *Model-View-ViewModel (MVVM)*. <https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm> (accessed: 16.05.2023)
- Pagade, G. (2022). *Difference Between Cohesion and Coupling*. <https://www.baeldung.com/cs/cohesion-vs-coupling> (accessed: 16.05.2023)
- Rehkopf, M. (n.d.). *Kanban vs. scrum: which agile are you?* <https://www.atlassian.com/agile/kanban/kanban-vs-scrum> (accessed: 10.05.2023)
- Sommerville, I. (2021). *Engineering software products : an introduction to modern software engineering*. Pearson Education Limited.

- UAS-Norway. (2020). *Politiets slår ned på ulovlig droneflyving*. <https://www.uasnorway.no/politiets-slar-ned-pa-ulovlig-droneflyvning-gir-hoyere-boter-for-ulovlig-droneflyvning/> (accessed: 25.05.2023)
- Utilsynet. (2023). *Universell utforming av apper*. <https://www.uutilsynet.no/regelverk/universell-utforming-av-apper/230> (accessed: 25.05.2023)
- Utilsynet. (n.d.). *WCAG sortert etter prinsipp*. <https://www.uutilsynet.no/wcag-standarden/3-forstaelig/717> (accessed: 10.05.2023)
- W3C. (2008). *Web Content Accessibility Guidelines (WCAG) 2.0*. <https://www.w3.org/TR/WCAG20/> (accessed: 24.05.2023)
- Zaptest. (n.d.). *Hva er enhetstesting? Dyp ned i prosessen, fordeler, utfordringer, verktøy og mer!* <https://www.zaptest.com/no/hva-er-enhetstesting-dyp-ned-i-prosessen-fordeler-utfordringer-verktøy-og-mer> (accessed: 03.05.2023)

8 Vedlegg

8.1 Intervjuguide

Innledning

1. Ønsker velkommen
2. Sier at vi starter opptak og understreker at informant har gjort seg kjent med samtykkeskjema og har skrevet under på forhånd.
3. Forklarer formålet:

Oppvarming

1. Hvor lenge har du flydd drone? 1år
2. Hvor ofte flyr du drone?
3. Hva fikk deg til å starte å fly drone?
4. Er det noen årstider eller tidspunkter du foretrekker å fly på?

Hoveddel

1. Har du støtt på noen utfordringer ved dronekjøring?
2. Hvordan vet du hvor du kan fly? Er det oversiktlig å finne?
3. Vet du om noen regler knyttet til å fly drone, og synes du disse er oversiktlige/lette å finne frem til?
4. Benytter du noen hjelpeemidler i forbindelse med flyvingen?
5. Er det noen hjelpeemidler du savner?
6. Har du vært borti en droneapp tidligere?
7. Hvis ja, var det noe spesielt du likte ved appen?
8. Var det noe spesielt du mislikte eller som kunne vært annerledes?
9. Hvilen funksjoner kunne du tenke deg å ha i en slik app?
10. Hva ønsker du å oppnå ved å bruke en droneapp?

Avrunding

1. Er det noe mer du ønsker å legge til?

8.2 Intervjuplan

Informant	Rollestittel	Ansvarsområde
Informant 1	Dronekjører	Kjører drone.
Tidspunkt	Deltakere	Hensikt
22. feb 2022	Informant 1 Skribent 1 (S) Skribent 2 (S) Intervjuer 1 (I) Intervjuer 2 (I)	Undersøke hvilke funksjonstester brukeren ønsker seg.

Tabell 1: I = Intervjuer. S = Skribent og ansvarlig for lydopptak. O = Observator. Vi planlegger å gjennomføre to intervjuer, hovedsaklig over Zoom.

Praktiske bestemmelser:

Lengde: 60 min

Utstyr: Mobiltelefon til opptak, intervjuguide, notater på laptop.

Sted: Temas

Deltaker: Informant 1

Intervjuer:

Skribent:

Tidspunkt:

Intervjuform: Semistrukturert. Spørsmålstyper: Lukkede spørsmål til å åpne et emne, og åpne spørsmål for utdypning.

Utdeling av samtykkeskjema: Sendes på mail i forkant av intervjuet.

Annet: Skribent noterer timestamps underveis for å gjøre det enklere å finne interessante deler av lydopptaket.

8.3 Analyse fra brukerintervju

ANALYSE INTERVJU 1

Punkter vi kan ta med oss videre fra intervjet

Flyr ca. 1 gang i måneden

Personen tar det på følelser for å avgjøre om det er innanfor å fly området hen befinner seg i.

Dersom det ikke er lov til å kjøre drone i det gitte området, vil ikke krontrollappen sende opp dronen

Brukeren har flydd drone av og på i 1 år

Kunne vært interessant med et sosiale medier interface, men at man før kan miste appens hensikt litt når det blir for mye av slikt

Personen kommer ikke på noen hjelpemiddler som hen savner

Foretrekker å fly på sommeren mtp. batteritid og bedre lys.

Personen tenker at det er oversiktig å finne reglene. Det er kun ett googlesøk unna

Hen ønsker å ha en app som "Forteller hvor lovlig det du holder på meg er"

Med en droneapp ønsker han å oppnå:
1. Syre og kontrollere drone
2. Kunne finne relevant informasjon uten å være innom 20 forskjellig apper på mobilen."

Hjelpemiddler: kontrollerappen, yr UAV forcast

"Trur ikke det interfacet sier noe om vær og vind, så det kunne vært nyttig"
Det er snakk om kontrollerappen

Figur 44: Analyse fra brukerintervju 1

ANALYSE INTERVJU 2

Punkter vi kan ta med oss videre fra intervjuet

Flydd
drone
i 9-10 år

Kjapt å fly
på høsten,
fordi da er
det grått

Bruker
app på
tlf

Flyr ca. 1 gang
i mnd eller
annenhver
mnd

Vind er
konstant en
utfordring, og
regn og snø

Savner
egentlig
ikke noe
spesielt

Flyr drone for å få
bilder og videoer
fra luften, og få et
unikt perspektiv
på landskapet

Sjekker regler
på
Luftfartstilsyne
t. Ikke alltid
oversiktlig

Vil gjerne ha
opplæringsdel i
appen, samt
oversikt over vær
og vind, og hvor
det er mulig å fly

Figur 45: Analyse fra brukerintervju 2

8.4 Samtykkeskjema

Oslo / 20.03.23

Vil du delta i brukerundersøkelsen «Droneflyving»?

Vi er en gruppe i emnet *IN2000 – Software Engineering med Prosjektarbeid* ved Institutt for informatikk ved Universitetet i Oslo. Med dette skrivet ønsker vi å informere hva prosjektet våres har som formål, og spørre deg om du vil delta i prosjektet, samt berette hva deltagelse vil innebære for deg.

Formål

Formålet med prosjektet er å undersøke droneflyving. Vi ønsker også å forstå dine behov og ditt syn på temaet, slik at vi kan lære mer om hva en trenger som en droneflyver, samt lage en app som tilfredsstiller dine behov.

Deltakelse

Du blir spurta om å delta fordi du faller innenfor våres målgruppe, definert som «nordmenn som flyr drone, og ønsker å fly dronen sin på et trygt og lovlig sted i Norge». Dersom du velger å delta ønsker vi å intervju deg som datainnsamlingsmetode. Intervjuet vil vare i cirka 30 min, og jeg kommer til å notere underveis.

Frivillig deltagelse

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

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

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

Personlige opplysninger vil bli anonymisert i transkriberingen og rapporteringen senest 26.05.23; ingen andre enn jeg og våre gruppelærere vil ha tilgang til dataen, og det som oppbevares av anonymisert rapportering fra intervjuet vil følge Universitetet i Oslo sine rutiner for sikker oppbevaring.

Navn og kontaktinformasjon vil kun vi på gruppen ha tilgang på. Dataen kan ettersendes deg ved ønske.

Rettigheter

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

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

Hvis du har spørsmål til undersøkelsen, eller ønsker å benytte deg av dine rettigheter, ta kontakt med Marte Lunde Kvam eller min universitetsansatte fagkontakt Jesper Dahl Norgård på e-post jesperdn@ui.no.

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

Med vennlig hilsen

Marte Lunde Kvam, på vegne av Gruppe 18

Tlf: [REDACTED]

Samtykkeerklæring

Jeg har mottatt og forstått informasjon om brukerundersøkelsen «Droneflyving», og har fått anledning til å stille spørsmål.

Jeg samtykker til å ha et intervju, og at mine opplysninger behandles frem til semesteret er avsluttet.

[REDACTED]

Sted og dato

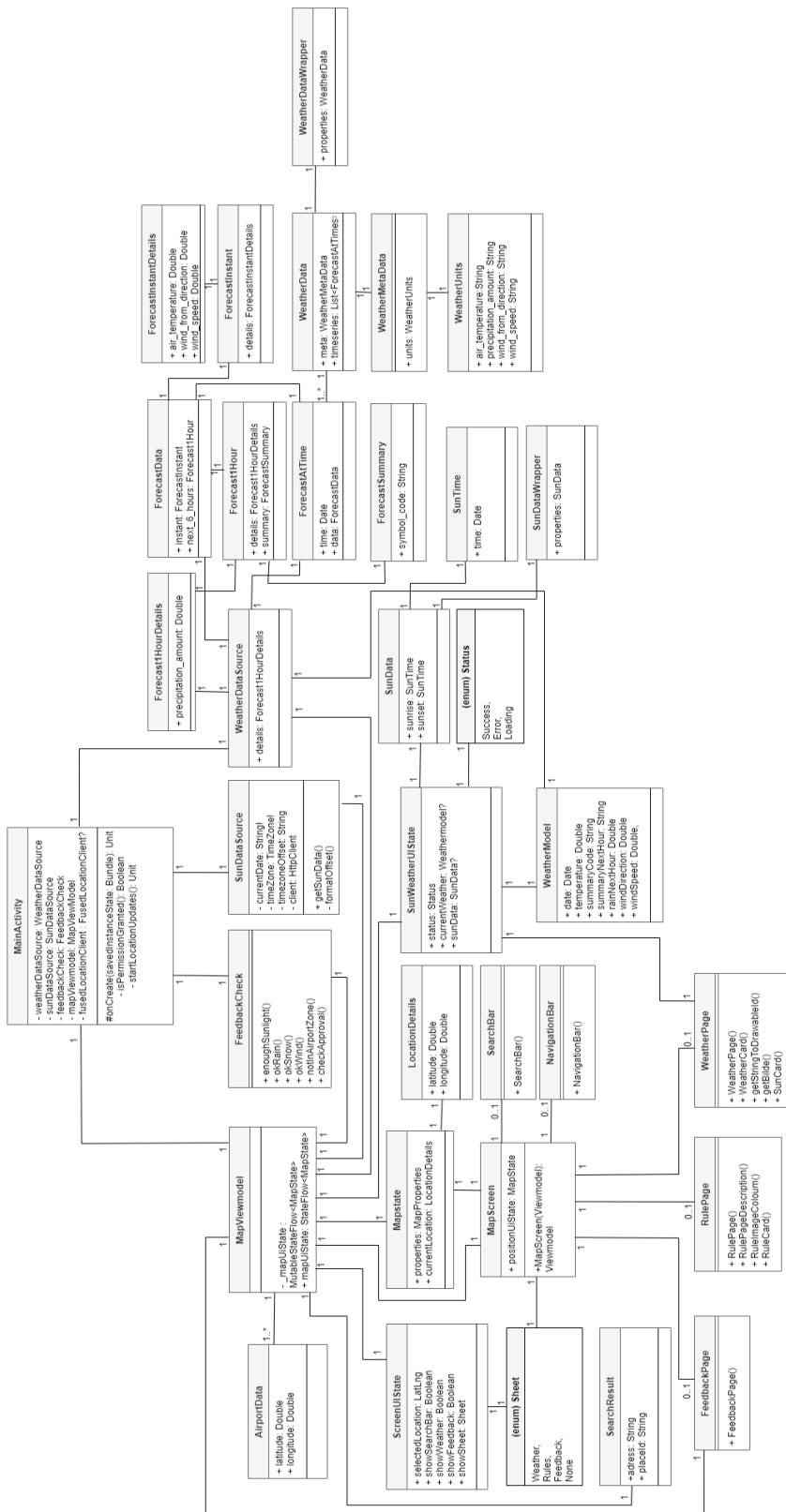
[REDACTED]

Fullt navn

[REDACTED]

Signatur

8.5 Klassediagram



Figur 46: Klassediagram som gir oversikt over applikasjonens klasser og hvordan de henger sammen.

8.6 Oppdatert kravspesifikasjon

MÅ

Krav	Beskrivelse	Prioritering	Type krav
Kartfunksjon	Appen sin hovedskjerm skal vise et interaktivt kart som viser geografisk informasjon over Norge	1	Funksjonelt krav
Knapp med begrunnelse	Brukeren skal kunne trykke på en knapp for å få tilgang til en begrunnelse eller forklaring på hvorfor droneflyging er tillatt eller ikke tillatt i det valgte området.	1	Funksjonelt krav
GPS	Appen skal vise brukerens nåværende geografiske posisjon.	1	Funksjonelt krav
Søkefunksjon	Brukeren skal kunne søke etter en spesifikk adresse eller område for å få opp informasjon om de gjeldende dronereglene der.	1	Funksjonelt krav
Værinformasjon	Appen skal gi brukeren beskjed om flyforholdene er egnet for droneflyging basert på gjeldende værforhold.	1	Funksjonelt krav
Fargeinndeling	Kartet skal ha en tydelig fargeinndeling som indikerer områder hvor droneflyging er tillatt eller ikke tillatt.	2	Funksjonelt krav

Figur 47: Oppdaterte må-krav

BØR

Krav	Beskrivelse	Prioritering	Type krav
Side med regler	Appen skal ha en egen side der alle drone reglene ligger.	2	Funksjonelt krav
Valg av drone	Bruker må velge hva slags drone de skal fly. Basert på valget får man tilpassede regler.	2	Funksjonelt krav
Internett feilmelding	Dersom brukeren ikke har internett skal det komme opp en feilmelding.	2	Funksjonelt krav
Varsling om værforhold	Bruker skal kunne velge om de vil få varsel når forholdene er gode nok til å fly drone.	3	Funksjonelt krav
Opplæringskurs	Brukere får et opplæringskurs i droneflygning.	3	Funksjonelt krav

Figur 48: Oppdaterte bør-krav

Ikke-funksjonelle krav

Krav	Beskrivelse	Prioritering	Kravtype
GDPR	Appen skal følge regler i personverforordningen.	2	Eksternt krav
Kræsjfritt	Appen skal ha minimalt med bugs, og skal ikke kræsje	1	Produktkrav
Rask lokasjonshastighet	Brukeren bør finne ut om lokasjonen sin er godkjent eller ikke på under 10 sekunder	1	Produktkrav
Universell utforming	Appen skal følge følgende krav til universell utforming:	2	Eksternt krav
	<ul style="list-style-type: none"> • 1.3.1 Informasjon og relasjoner • 1.3.2 Meningsfylt rekkefølge • 1.4.3 Kontrast (minimum). • 2.4.3 Fokusrekkefølge (Nivå A) • 2.4.4 Formål med lenke (i kontekst, Nivå A) • 3.2.3 Konsekvent navigering (Nivå AA) • 3.2.4 Konsekvent identifikasjon (Nivå AA) • 3.3.2 Ledetekster eller instruksjoner (Nivå A) 		
Lett å navigere	Brukern skal kunne finne lFI og sjekke været i løpet 20 sekunder	2	Produktkrav
Språk	Appen være på norsk	1	Produktkrav
Asynkront	Appen skal hente data fra API-ene asynkront slik at den ikke henger seg.	1	Organisatorisk krav
Kodespråk	Appen skal skrives i Kotlin	1	Organisatorisk krav
Plattform	Appen skal utvikles i Android studio	1	Organisatorisk krav
API-level	Appen skal kjøres på Android med API level 23	1	Organisatorisk krav
Google Maps	For å hente inn lokasjon og lage Google Maps skal appen bruke API-et Maps SDK (Google)	1	Organisatorisk krav
Lokasjon	For å hente inn forslag til lokasjoner basert på søk skal appen bruke API-et Location SDK (Google).	1	Organisatorisk krav
Soldata	For å hente tidspunkter for soloppgang og solnedgang skal appen bruke API-et Sunrise 3 beta (MCT API)	1	Organisatorisk krav
Værdata	For å hente inn værets på en gitt lokasjon skal appen bruke API-et Locationforecast/2.0 (MET API for værdata)	1	Organisatorisk krav
Tidsfrist for MVP'en	MVP'en må være ferdig til 31.mars	1	Organisatorisk krav
Tidsfrist for levering av appen	Appen må være ferdig og klar til levering innen 26.mai	1	Organisatorisk krav
Loggføring	Hvert møte skal loggføres	1	Organisatorisk krav
Dokumentasjon	Arbeidet og veivalg skal dokumenteres jevnlig i Mirobrettet	1	Organisatorisk krav
Teamet skal møtes til faste møtetidspunkter.	Faste møtetidspunkter: tirsdag: kl.12-14 onsdag: kl.12-14 fredag: kl.10-12 Romansvalig har ansvar for å booke rom til møtene.	1	Organisatorisk krav
Budsjett	Utviklingen av appen skal holde seg innenfor budsjettet på 0 kr.	1	Organisatorisk krav
Rapport	Rapporten skal skrives i Overleaf.	1	Organisatorisk krav
Språk	Rapporten og kommentarer i koden skal skrives på norsk.	1	Organisatorisk krav

Figur 49: Oppdaterte ikke-funksjonelle krav

8.7 Opprinnelig kravspesifikasjon

KRAV	BESKRIVELSE	PRIORITERING
MÅ		
Kartfunksjon	Appen sin hovedskjerm skal vise et kart over Norge.	1
Flygetillatelse	Bruker må få vite om de har lov eller ikke til å kjøre drone i deres område.	1
GPS	Appen sjekker hvor bruker befinner seg.	1
Fargeinndeling	På kartet skal det være farge inndelt over hvor man kan/kan ikke fly droner.	2
Søkefunksjon	Bruker kan søke på adresse/område for å se reglene som gjelder der	2
Knapp med begrunnelse	Bruker kan trykke på en knapp for å lese om hvorfor de ikke kan fly drone på det valgte området.	2
Vær	Bruker får beskjed om det er godt flyvær eller ikke.	1
BØR		
Side med regler	Appen skal ha en egen side der alle drone reglene ligger.	2
Valg av drone	Bruker må velge hva slags drone de skal fly, basert på valget får man tilpassede regler.	2
Varsling om forhold	Bruker skal kunne velge om de vil få varsel når forholdene er gode nok til å kunne få drone.	3
Opplæringskurs	Bruker får en opplæringskurs i droneflyging.	3

Figur 50: Bilde av 'Må ha' og 'Bør ha' krav.

KAN		
Tommel opp	Når alle forhold er gode for å fly får bruker en tommel opp på skjermen.	2
Kalender	Brukere kan sjekke dager i forveien for å se om det er greit å kjøre drone.	2
Kunnskapstest	Brukere må/kan velge å ta en test i drone reglene.	3
Favorittplasser	Brukere kan lagre sine favorittplasser på appen.	3
Historikk	Appen har en historikk på alle steder bruker har flydd drone.	3
Registrering av drone	Brukeren kan koble opp sin egen drone til appen.	3
Autentiseringssystem	Brukere kan registrere seg på appen og ha en egen profil.	3
Venner	Brukere kan legge til venner på appen.	3
Bildefunksjon	Brukere kan ta bilder og dele dem innen i appen med vennene sine.	3

Figur 51: Bilde av kravene vi eventuelt kunne ha implementert om det ble tid til overs.

KRAV	BESKRIVELSE	PRIORITERING
Krav	Beskrivelse	
Flygetillatelse	Sjekke om det er trygt å sende opp drone på en gitt lokasjon, i forhold til vær, sollys og regler.	Funksjonelt krav
Kartfunksjon	Viser hvor brukeren er i kartet	Funksjonelt krav
Kart-API	Appen skal benytte seg av Google Maps sitt kart-API.	Ikke-funksjonelt krav
GPS	Appen skal kunne hente inn lokasjonen til brukeren.	Funksjonelt krav
Crash-resistant	Appen skal ikke kunne crashe iløpet av kjøretiden.	Ikke-funksjonelt krav
Asynkronitet	Appen skal hente data fra API-ene asyntront slik at den ikke henger seg	Ikke-funksjonelt krav <input checked="" type="checkbox"/>

Figur 52: Kravene til MVP'en (Minimal Viable Product)

8.8 Teamavtale

Teamavtale

1. Tilstedeværelse

Det forventes at alle møter opp til de fleste møtene og ellers bidrar likt i gruppa på oppgavene deres.

2. Tidsbruk

Gruppa ser for seg at hvert individ skal investere omtrent 30-35 timer i uka på prosjektarbeidet. Dette inkluderer også tiden vi deltar på møter og har felles opplegg.

3. Forventninger til den enkeltes bidrag

Kommunikasjon: Vi forventer at alle i gruppen kommuniserer jevnlig og tydelig med hverandre. Dette inkluderer å svare på meldinger og e-poster innen rimelig tid, og å gi klar og ærlig tilbakemelding til hverandre.

Ansvarlighet: Vi forventer at alle tar ansvar for sine oppgaver og holder seg til tidsfrister. Dette inkluderer også å gi beskjed i god tid dersom man ikke kan levere innen fristen, slik at resten av gruppen kan justere sin arbeidsfordeling.

Samarbeid: Vi forventer at alle jobber godt sammen og bidrar til et positivt arbeidsmiljø. Dette inkluderer å være åpen for andres synspunkter og ideer, og å jobbe for å finne felles løsninger på eventuelle uenigheter.

Respekt: Vi forventer at alle behandler hverandre med respekt og verdighet. Dette inkluderer å unngå nedlatende kommentarer, personlige angrep eller diskriminerende språk.

Bidrag: Vi forventer at alle bidrar til prosjektet på en positiv måte, og at alle har en viktig rolle å spille. Dette inkluderer å ta initiativ, komme med konstruktive forslag og jobbe aktivt for å oppnå felles mål.

Kvalitet: Vi forventer at alle strekker seg etter å levere arbeid av høy kvalitet. Dette inkluderer å sørge for at alle oppgaver er grundig gjennomarbeidet, og at resultatene er relevante og korrekte.

Plikter: Vi forventer at alle oppfyller sine plikter som medlem av gruppen, inkludert å delta på møter, levere oppgaver i tide og bidra til diskusjoner og beslutninger. Vi sier også ifra i god tid dersom vi ikke kan delta på møtene.

Hva som skjer ved avvik eller uenigheter

Dersom det oppstår avvik eller uenigheter er det viktig at vi kommuniserer med hverandre

og prøver å komme frem til en løsning. Her er det viktig at alle lytter til hverandre og er åpne for andre forslag. Dersom det blir nødvendig vil vi booke time med veileder slik at vi kan få hjelp til å løse potensielle konflikter dersom vi ikke klarer dette på egenhånd.

8.9 Brukertest

Oppgaver	Tid	Antall feil
Sjekk været i din lokasjon	1 sek	0
Sjekk om du kan fly i din lokasjon	36 sek	2 (Trykket først på regelsiden)
Fin ifi og sjekk været	12 sek	1 (Satt ikke en pin og fikk derfor ikke oppdatert været)
Finn ut om du kan fly ved gardemoen	47 sek	1 (Hen trudde at pinnen var satt, ettersom det allerede var en pinn ved gardemoen. Prøvde også å trykkepå den for å få oppdatert plasseringen, men det fungerte ikke.)
Finn reglene for å fly drone	4 sek	1 (Trykket først på begrunnelsessiden.)

Figur 53: Resultatene vi fikk av brukertesten.



Figur 54: Tilbakemeldinger vi fikk av brukeren.

8.10 Universell utforming

Krav	Kort fortalt	Hva har vi gjort	Oppfylt
1.3.1 Informasjon og relasjoner	Ting skal være kodet slik det ser ut	<ul style="list-style-type: none"> Vi oppfyller dette da appen er godt formulerert. Vi har ikke mye overflødig informasjon som forstyrrer brukeren. De ulike sidene har en rolle. Enkelt å finne alle siden Alle bildene har lesselig bildetekst 	Delvis oppfylt
1.3.2 Meningsfull rekkefølge	Presenter innhold i en meningsfull rekkefølge	<ul style="list-style-type: none"> Går ikke ann gå se vill i appen, med tanke på at alt er på en siden. Vi har ikke alternative navigeringsmetoder slik som søkefunksjon Brukeren kan bruke både kart og søkefunksjon for å navigere i kartet 	Delvis oppfylt
1.4.3 Kontrast (minimum)	Kontrastforholdet mellom teksten og bakgrunnen er minst 4,5:1	<ul style="list-style-type: none"> Brukt appen "Accessibility Scanner" for å sjekke om kontrastene er bra nok 	Oppfylt
2.4.4 Formål med lenke (i kontekst, Nivå A)	Alle lenkers mål og funksjon fremgår tydelig av lenketeksten	<ul style="list-style-type: none"> Brukt appen "Accessibility Scanner" for å sjekke og godkjenne at alt av tekst og bilder har hatt stor nok kontrast. 	Oppfylt
3.2.3 Konsekvent navigering (Nivå AA)	Navigasjonslinker som gjentas på flere sider skal ha en konsekvent rekkefølge	<ul style="list-style-type: none"> Brukt nettstedet "icons8.com" for å finne passende ikoner, der vi har prøvd å velge symboler andre apper også bruker. 	Oppfylt
.2.4 Konsekvent identifikasjon (Nivå AA):	Elementer som har samme funksjonalitet på tvers av flere sider er utformet likt	<ul style="list-style-type: none"> Knappene nederst på siden er konsekvente. De ulike sidene har lik design-utforming 	Oppfylt
3.3.2 Ledetekster eller instruksjoner (Nivå A)	Det vises ledetekster eller instruksjoner når du har skjemaelementer som må fylles ut	<ul style="list-style-type: none"> I søkerfeltet hvor man kan skrive inn adresse står det forklart hva man skal skrive inn 	Oppfylt

Figur 55: Kravene og beskrivelsen er hentet fra (Utilsynet, 2023).