



Pakkeliste-applikasjonen: *PackMate*

IN2000 – Software Engineering med prosjektarbeid

Teammedlemmer: Aron (aronbj), Sander (sandehal), Trym (taeriks), Haakon (haakonpk), Ahmed (ahmeaah) og Alexander (palovold).

Team-nummer : 14

Veiledere : Gyda Sæther og Jesper Nordahl

Dato : 26.05.2023

Innholdsfortegnelse

1.0 Presentasjon.....	3
1.1 PackMate – en hjelpende hånd fra PackBuddy.....	3
1.2 Introduksjon.....	4
1.3 Motivasjon.....	5
1.4 Visjon.....	6
1.5 Teammedlemmene.....	6
2.0 Brukerdokumentasjon.....	9
2.1 Målgruppe og hensikt.....	9
2.2 Oversikt – Funksjonalitet.....	9
2.3 Hovedskjermer og bruksområde.....	10
2.4 Pre-betingelser.....	12
2.5 Installasjon.....	13
3.0 Kravspesifikasjon og modellering.....	13
3.1 Kravspesifikasjon: Funksjonelle krav.....	14
3.2 Kravspesifikasjon: Ikke-funksjonelle krav.....	15
3.3 User stories.....	17
3.4 Use case – Modellering av krav.....	18
3.5 Klassediagram.....	19
3.6 Sekvensdiagram.....	20
3.7 Objektorienterte prinsipper og design patterns.....	21
4.0 Produktdokumentasjon.....	23
4.1 Oversikt – Løsningen.....	23
4.2 Konkretisering.....	24
4.3 Kvalitetsegenskaper og evaluering.....	29
4.4 API fra Meteorologisk Institutt.....	33
4.5 Utfordringer – API fra Meteorologisk Institutt.....	34
5.0 Prosessdokumentasjon.....	35
5.1 Formelle rammer for prosjektet.....	35
5.2 Prosjektløp.....	42
5.3 Iterasjonene.....	45
5.4 Oppsummering.....	59

6.0 Bibliografi.....	60
7.0 Vedlegg.....	62
7.1 Sekvensdiagram.....	62
7.2 Brukertestguide MVP.....	64
7.3 Klassediagram.....	66
7.4 Testing av WCAG krav.....	67
7.5 Prototype iterasjon 1.....	68
7.6 Prototype iterasjon 2.....	68
7.7 Prototype iterasjon 3.....	69
7.8 Siste retrospektiv – Iterasjon 1.....	70
7.9 Mislykkede avatarer.....	71
7.10 Teamavtale.....	72

1.0 Presentasjon

1.1 PackMate – en hjelpende hånd fra *PackBuddy*



Heisann. Jeg heter *PackBuddy* og jeg er din personlige tur-assistent!

Du trenger ikke å fortvile over sur vind eller uforutsigbare regnskurer når du har meg ved din side! Enten du er en erfaren turgåer eller en rykende fersk nybegynner, så skal jeg gi deg den viktigste informasjonen som du trenger for å ha en knall tur-opplevelse! Alt du trenger er å laste ned appen min, *PackMate*.

"Hvordan er dette mulig?!"

"Dette må jo være for godt til å være sant!?"
- Aron, Sander, Alexander, Haakon, Trym og Ahmed.

Jeg vet, applikasjonen *PackMate* kan virke som en teknisk bragd uten sidestykke og du lurer nok på hvordan jeg klarte det. Ved hjelp av presise værdata fra Meteorologisk institutt og komplekse algoritmer, kan jeg estimere den perfekte tur-pakkelisten for å beskytte deg mot naturkraftene. Ønsker du å lære mer? Da er det bare å lese videre!

1.2 Introduksjon

Denne rapporten er en overordnet beskrivelse av arbeidet for å fremstille pakkeliste-applikasjonen *PackMate*, som ble gjort i emnet IN2000 – Software Engineering med prosjektarbeid. Applikasjonen ble utviklet i henhold til kravene for Case 4: *Åpent case – i lufta*.

Dette caset omhandler å fremstille en applikasjon relatert til atmosfæren og bruke obligatoriske datakilder fra Meteorologisk institutt. Hensikten med prosjektet er å gjennomføre et omfattende systemutviklingsprosjekt i team og få trening i å bruke moderne metoder, teknikker og verktøy innen software engineering (UiO, 2023).

Case 4: *Åpent case – i lufta* har følgende formelle krav:

- Forslaget for applikasjonen krever godkjenning fra veileder/kursledelsen.
- Applikasjonen må bruke minst to værprodukter, inkludert MetAlerts dersom det er relevant for situasjonen.
(Meteorologisk institutt, 2023)

1.3 Motivasjon

Vi valgte Case 4: *Åpent case - i lufta*. Vi valgte dette caset spesifikt fordi:

- For det første, var det den suverene favoritten da gruppen ble bedt om å rangere casene etter interesse. Gruppen var mer motivert til å lage en applikasjon relatert til vær eller luftfart, relativt sett mot de andre alternativene.
- For det andre, gav caset større kreativ frihet for anvendelsen av API. Ettersom caset er åpent tillater det for større handlingsrom og kreativ frihet.

Applikasjonen *PackMate* har et tydeligere fokus mot vær og vindforhold, fremfor luftfart. Dette var en avveining som ble gjort for å tilspisse applikasjonen og sikre at man benyttet API fra Meteorologisk institutt på en hensiktsmessig måte.

1.4 Visjon

I dette prosjektet utviklet man pakkeliste-applikasjon *PackMate*. For å konkretisere produkt-visjonen, benyttes rammeverket fra *Crossing the Chasm* av Geoffrey Moore (Sommerville, 2021, s.18):

Løsningen *PackMate* er tiltenkt turgåere i Norge, erfarte eller ikke, som trenger anbefalinger om hva man skal ha med på tur. *PackMate* er en pakkeliste-app som skaper helhetlige pakkelister, med et særskilt fokus på bekledning. Ulikt andre tjenester, er denne løsningen gratis og særlig tiltenkt nybegynnere.

Visjonen er fremstilt som et resultat av aktiv diskusjon. Kravene til caset utgjør rammene for visjonen, men i utgangspunktet er selve beskrivelsen av løsningen *PackMate* skapt av utviklerne, fremfor en eksplisitt ekstern kunde.

1.5 Teammedlemmene

Teamet vårt er en tverrfaglig gruppe som ønsker å lage den desidert "kuleste pakkeliste-applikasjonen" noensinne! Fire av teammedlemmene går Programmering og systemarkitektur og de to resterende medlemmene er fra Digital økonomi og ledelse. Turinteressen blant teammedlemmene varierer, men alle har verdifull innsikt å dele. Gruppen skal skape en pakkeliste-applikasjon som kommer alle til gode. Både de erfarte turgåerne, men særlig de uerfarne. Det er altså en styrke at vi stiller ulikt med tanke på relevant domenekunnskap.



Sander

Studieprogram: DIGØK

Favorittemne: IN1010

Forventninger: Jeg vil lære mer om prosjektarbeid som prosess, de ukentlige møtene og det tekniske arbeidet!

Turpreferanse: Jeg er mest vant til kortere turer i naturen, ikke langt unna der jeg bor. Liker å gå tur, men først og fremst når det er sol og fint vær.



Ahmed

Studieprogram: PROSA

Favorittemne: IN2090

Forventninger: Det blir artig å ha møter med teamet, og å lage en fet app!

Turpreferanse: Pleier ikke å gå turer, men er gira på å kunne bruke en slik app i fremtiden.



Trym

Studieprogram: PROSA

Favorittemne: IN2030

Forventninger: Jeg ser frem til å jobbe i team med en flott gjeng karer. Det blir også veldig spennende å arbeide med apputvikling for å få en smakebit på arbeidslivet.

Turpreferanse: Jeg har gått tur hele livet og er aktivt ute nå også 😊

Studieprogram: DIGØK

Favorittemne: EXPHIL <3

Forventninger: Det vil alltid gå bra til sist

Turpreferanse: Ja liker å gå tur, men jeg er ikke en villmarkens mann akkurat.



Aron

Studieprogram: PROSA

Favorittemne: IN1010

Forventninger: Jeg ser frem til å lære mer om både utvikling og programmering av en app, og det å jobbe i et team og prøve meg på andre deler av prosjektet som design, testing og arkitektur.

Turpreferanse: Jeg liker å gå tur, men jeg vil ikke si at jeg er en ekspert på turgåing.

Alexander



Studieprogram: PROSA

Favorittemne: IN2010

Forventninger: Jeg ser frem til å jobbe om et større prosjekt og ha et sluttprodukt jeg er stolt av.

Turpreferanse: Elsker å gå tur! Hver sommer drar jeg og familien min på en lengre fjelltur i fjellet.



Haakon

2.0 Brukerdokumentasjon

2.1 Målgruppe og hensikt

Denne brukerdokumentasjonen er utformet i henhold til anbefalinger fra NDLA og er tiltenkt de som ønsker en innføring i bruk av pakkeliste-applikasjonen *PackMate* (NDLA, 2023). Det forventes ikke betydelige tekniske forkunnskaper for å kunne benytte seg av brukerdokumentasjonen. Enhver bruker anbefales å ta en kjapp gjennomgang for å sikre en positiv bruksopplevelse. For å benytte *PackMate* må man ha en Android-enhet med minimum API-nivå 26.

Målgruppen for applikasjonen er alle med interesse for å gå på tur, men særlig uerfarne turgåere som trenger lettfattelige anbefalinger for bekledning.

2.2 Oversikt – Funksjonalitet

Applikasjonen *PackMate* tilbyr følgende funksjonalitet:

- Brukeren kan velge lokasjon på et kart over Norge.
- Brukeren kan se værmeldingen for lokasjonen man har valgt.
- Brukeren mottar farevarsler dersom det eksisterer for lokasjonen man har valgt.
- Brukeren mottar en pakkeliste som består av ytter- og innerplagg, tilpasset vær og vindforhold ved lokasjonen.
- Pakkelisten kan lagres lokalt på enheten til brukeren.

2.3 Hovedskjermer og bruksområde



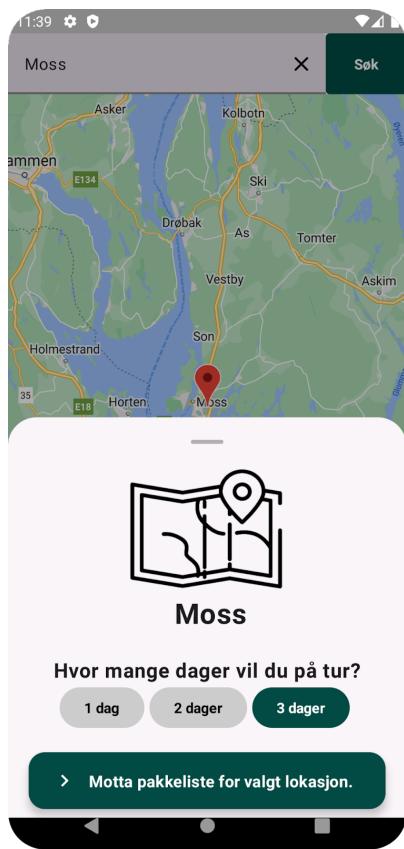
2.3.1 Skjerm: Lagrede pakkelister

Ved oppstart av applikasjonen *PackMate* vil man bli møtt med "Lagrede pakkelister-skjermen". Denne skjermen inneholder en oversikt over lagrede pakkelister. Man kan også navigere seg til denne skjermen ved å trykke på stjerne-ikonet, nederst til høyre, navngitt "Lagret". Om man har lagrede pakkelister, kan disse inspireres nærmere ved å trykke på dem. Alternativt kan alle pakkelistene slettes ved å trykke på "Slett alle-knappen" øverst. Dersom man ønsker å opprette en ny pakkeliste kan man navigere seg til "kart-skjermen" ved å trykke på forstørrelsesglass-ikonet, nederst til venstre, navngitt "Kart".

2.3.2 Skjerm: Kart

Når man har trykket på forstørrelsesglass-ikonet nederst til venstre, vil brukeren navigere seg til "kart-skjermen". Denne skjermen inneholder et interaktivt kart over Norge. Brukeren kan enten søke opp en lokasjon eller fysisk trykke på stedet der man vil på tur. Det er mulig å trykke på andre lokasjoner, selv utenfor Norge, men da kan man ikke garantere at applikasjonen vil fungere slik den er tilskiktet.





2.3.3 Skjerm: Kart

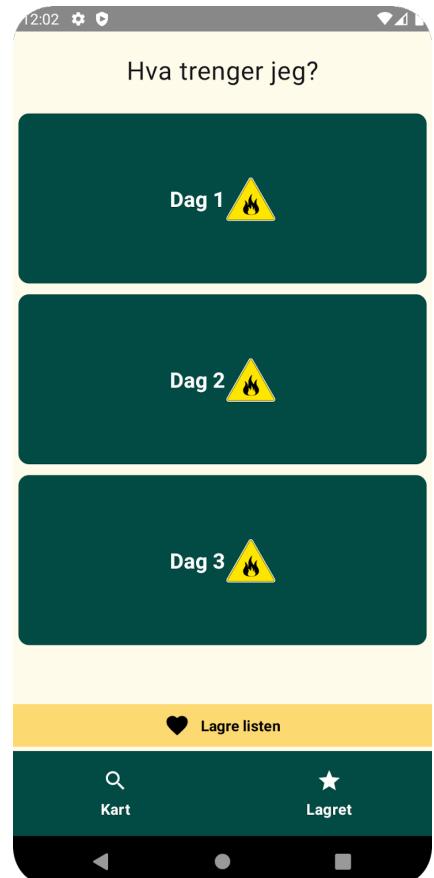
Etter valgt lokasjon, vil det komme fram et bottom-sheet. Her bekreftes lokasjonen som er valgt og man velger én, to eller tre dagers tur, som det skal lages en pakkeliste for. Ved å trykke på knappen "Motta pakkeliste for valgt lokasjon" nederst på skjermen, vil man bli brakt til "tur-oversikt-skjermen". Alternativt kan man dra ned bottom-sheet og trykke på et nytt sted på kartet, for å velge en ny lokasjon.

2.3.4 Skjerm: Tur- oversikt

På "tur-oversikt-skjermen" presenteres en oversikt over den valgte lokasjon. Dersom turen er lengre enn en dagstur vil det være flere knapper å interagere med, på denne skjermen.

Dersom man ønsker å lagre pakkelistene for hver dag ved den valgte lokasjonen, kan brukeren trykke på knappen med hjerte-ikonet og teksten "Lagre listen", nederst på skjermen.

Dersom man ønsker å se på den anbefalte pakkelisten for en av dagene ved lokasjonen, kan brukeren trykke på den dagen man ønsker informasjon om. Deretter vil man havne på "pakkelisten-skjermen" for den utvalgte dagen.





2.3.5 Skjerm: Pakkeliste

Dette er ”pakkeliste-skjermen”. Her kan man sideveis gjennomgå anbefalte klesplagg, både for ytterlag og innerlag. Nederst på skjermen kan man ekspandere en oversikt for å vise været. Der får man informasjon om vind, temperatur og regn som kan forventes på dagen. Ved å trykke på ”Farevarsel”, vil appen vise farevarselet for den utvalgte lokasjonen. Denne knappen eksisterer kun om det er et farevarsel i området. For å motta en forklaring av poengsystemet for klærne som fremvises, kan brukeren trykke på ”i” knappen øverst i høyre-hjørne. Dersom man ønsker å gå tilbake til listeoversikten, finnes det en tilbakeknapp, øverst i venstre hjørne.

2.4 Pre-betingelser

Applikasjonen *PackMate* er beregnet for mobil-enheter på operativsystemet Android, med minimum API-nivå 26. Det er mulig å bruke *PackMate* gjennom emulator, slik som i Android Studio, men sjekk at operativsystemet ditt tillater for emulering. Sannsynligvis vil emulering fungere uten vesentlige endringer på systeminnstillinger, men for enkelte PC-er må man inn i BIOS for å tillate emulering. BIOS- innstillinger kan aksesseres ved å slå av PC-en, og under oppstart, vil det være en notifikasjon som forteller deg hvilke knapper du må trykke på for å gå inn i BIOS.

2.5 Installasjon

Slik det er nevnt ovenfor, anbefales emulering gjennom Android Studio dersom man ønsker å benytte emulator. Ved tidligere emulering av *PackMate* har det vært benyttet Android Studio versjon Flamingo | 2022.2.1. Flamingo. Fysiske Android-enheter, som har tilstrekkelig API-nivå og Google-play tjenester aktivert, skal også fungere for å drive *PackMate*. Ettersom applikasjonen ikke er tilgjengelig offentlig, kan den aksesseres gjennom filene som inngår i mappen "Team-14-TurPakkeListe" på Devilry. For installasjon kan man anbefale følgende handlinger:

1. Last ned zip-filen fra Devilry og pakk den ut.
2. Åpne prosjektmappen i Android Studio og trykk på knappen "Sync Project With Gradle Files" øverst i høyre hjørne.
3. Du vil få opp en melding som sier "Sync Android SDKs" og denne må du godkjenne.
4. Deretter kan du kjøre appen på en emulator som har Google services og API level 26 eller høyere.

3.0 Kravspesifikasjon og modellering

For applikasjonen er det utformet en kravspesifikasjon som har inndeling etter prioritet.

Prioritetene er inndelt fra: 1 - Høy prioritet, 2 - Middels prioritet og 3 - Lav prioritet.

Kravene som ikke er oppfylt i den nåværende versjonen av *PackMate* er merket med tegnet: *, etter navnet på det gjeldende kravet. I tillegg er oppfylte krav fargelagt med grønn og ikke-oppfylte med rød farge. Kravspesifikasjonen har vært kontinuerlig revidert, med utgangspunkt i de ulike iterasjonene av applikasjonen. Dette har resultert i kravspesifikasjonen nedenfor:

3.1 Kravspesifikasjon: Funksjonelle krav

Funksjonelle krav	Beskrivelse	Prioritet
Anbefaling om klær.	Applikasjonen skal gi brukeren anbefaling om passende klær.	1
Valg av område.	Brukeren skal kunne velge et område for turen som de skal på i Norge.	1
Motta værmelding.	Brukeren skal kunne motta en værmelding for et utvalgt område.	1
Motta farevarsel.	Brukeren skal kunne motta et farevarsel for et utvalgt område.	1
Lagre pakkeliste.	Brukeren skal kunne lagre pakkelisten for et utvalgt område.	1
Støtte tre WCAG 2.1 -krav.	Systemet skal ta høyde for minst tre ulike WCAG 2.1 - krav. Disse spesifiseres nedenfor.	1
Anbefaling om solkrem. *	Applikasjonen skal gi brukeren en hensiktsmessig anbefaling om solkremfaktor.	2
Anbefaling om annet turutstyr. *	Applikasjonen skal gi brukeren en anbefaling om annet turutstyr – slik som brodder eller telt.	2
Tilpasse pakkeliste etter vektbegrensning. *	Applikasjonen skal gi brukeren en pakkeliste tilpasset vektbegrensninger på en tur-sekk.	2
Valg av antall dager for tur.	Brukeren skal kunne velge antall dager de er på tur.	2
“Fjell-meny”. *	Brukeren skal kunne motta en oversikt over værforhold på et utvalg av “toppturer” i Norge.	3
Dark-mode. *	Applikasjonen skal støtte Dark-mode.	3
Adventure mode. *	Brukeren skal kunne trykke på en knapp for å endre stil-settingen til en “eventyr-modus”.	3

3.2 Kravspesifikasjon: Ikke-funksjonelle krav

Ikke-funksjonelle krav	Beskrivelse	Prioritet
Android-støtte.	Systemet skal fungere på Android.	1
Utviklet i Kotlin.	Systemet skal være utviklet ved bruk av Kotlin.	1
Utviklet med Scrumban.	Systemet skal utvikles gjennom den smidige arbeidsmetodikken Scrumban.	1
Tekniske avbrudd.	Systemet skal ikke krasje.	1
Utvikling med GitHub.	Systemet skal utvikles ved hjelp av tjenesten GitHub.	1
Jetpack Compose.	Systemet skal anvende biblioteket Jetpack Compose for å utforme UI-elementer.	1
Visualisering av arbeidsoppgaver med Trello.	Back-log og prosjektfremgang skal visualiseres med tjenesten Trello.	1
Prototyping av applikasjonen med Figma.	Prototyper av <i>PackMate</i> skal visuelt fremstilles ved hjelp av tjenesten Figma.	1
MetAlerts.	Systemet skal benytte MetAlerts for å fremstille farevarsler for brukeren.	1
Google Maps.	Systemet skal benytte Google Maps for å fremstille kartet for brukeren.	1
Locationforecast.	Systemet skal benytte Locationforecast for å fremstille vær og vind data.	1
Material 3.	Systemet skal anvende kode fra Google sitt open-source-design system, <i>Material 3</i> .	2
Tidseffekt.	Systemet skal bruke under to sekunder fra start til man er inne på hovedskjermen.	2

3.2.1 Eksterne funksjonelle krav: Universell utforming

Applikasjonen *PackMate* ivaretar følgende krav for universell utforming med utgangspunkt i *Web Content Accessibility Guidelines (WCAG) 2.1*:

- **WCAG 2.1 (Perceivable)** – Systemet skal være utformet etter WCAG 2.1 krav (1.4 Distinguishable): "Make it easier for users to see and hear content including separating foreground from background" (W3C, 2023).
- **WCAG 2.1 (Operable)** – Systemet skal tilstrebe å være utformet etter WCAG 2.1 krav (2.2 Enough Time): "Provide users enough time to read and use content" (W3C, 2023).
- **WCAG 2.1 (Understandable)** – Systemet skal tilstrebe å være utformet etter WCAG 2.1 krav (3.1 Readable): "Make text content readable and understandable" (W3C, 2023).

3.3 User stories

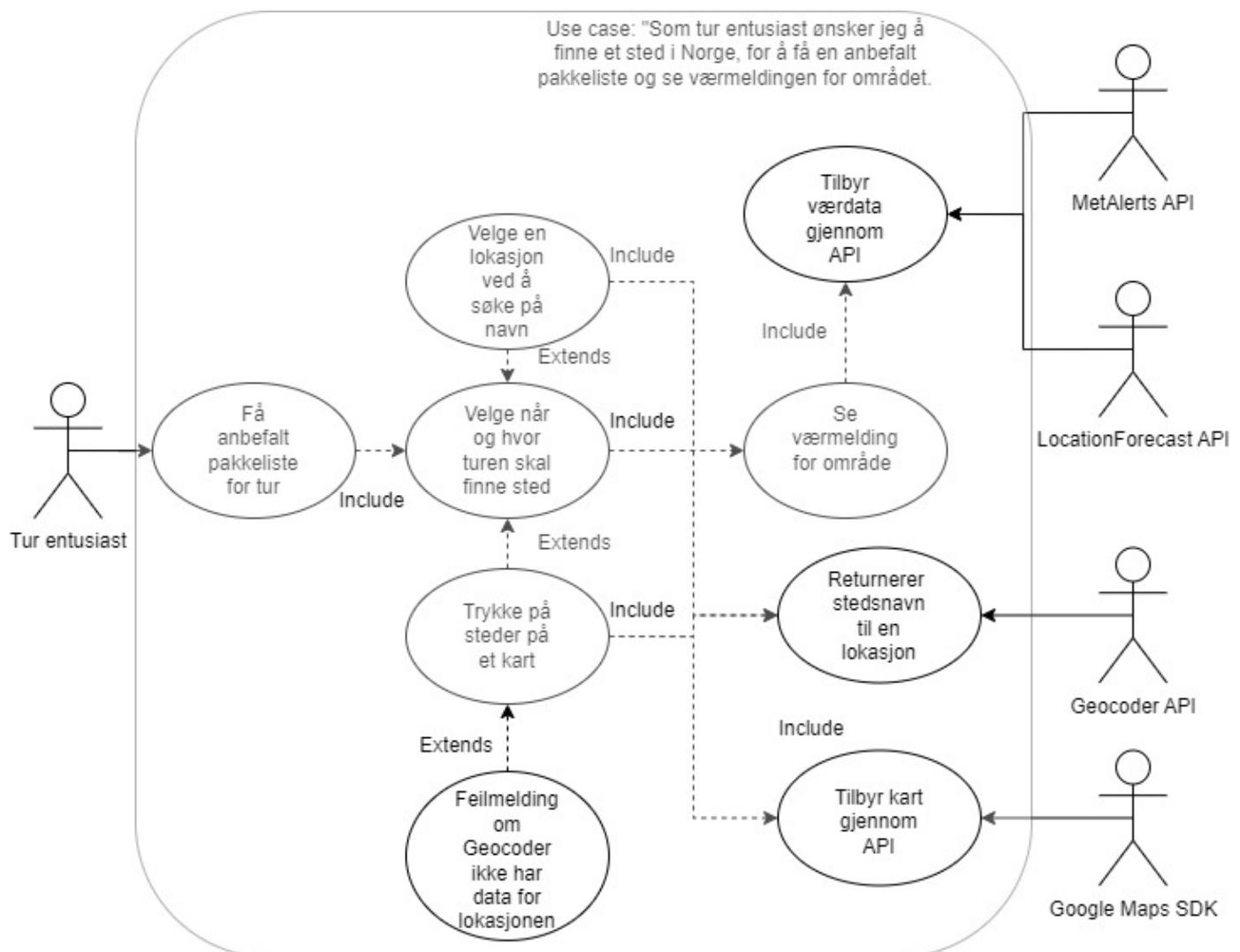
For å utforme kravspesifikasjoner og backlogger er user stories et viktig verktøy. Dette er beskrivelser av scenarioer hvor en bruker forsøker å gjøre noe med et software-system (Sommerville, 2021, s.76). Disse beskriver en sekvens av interaksjon med systemet, uten å gi en over-detaljert beskrivelse. Fokuset er på den tiltenkte brukerens ønsker, som skaper et utgangspunkt for å utforme et bestemt trekk ved applikasjonen.

Her er et utvalg av user stories som utgjør utgangspunktet for kravspesifikasjonen til *PackMate* og den tilhørende backloggen:

- Som en erfaren turgåer, ønsker jeg raskt å velge hensiktsmessige klær, for å kunne kjappere gå på tur.
- Som en erfaren turgåer, ønsker jeg å se egenskapene til klærne jeg skal bruke på tur, for å kunne forbedre min egen forståelse for turklær.
- Som en uerfaren turgåer, ønsker jeg lettfattelige råd for bekledning, slik at jeg enklere kan nyte turopplevelsen.
- Som en turgåer, ønsker jeg en lettfattelig oversikt over værforholdene for en gitt lokasjon, slik at jeg kan avgjøre om jeg vil på tur.
- Som en turgåer, ønsker jeg å motta farevarsler for en gitt lokasjon, fordi jeg vil være trygg på tur.
- Som en turgåer, ønsker jeg å kunne lagre pakkeinnholdet mitt digitalt, slik at jeg ikke må huske hva jeg har med meg eller trenger.

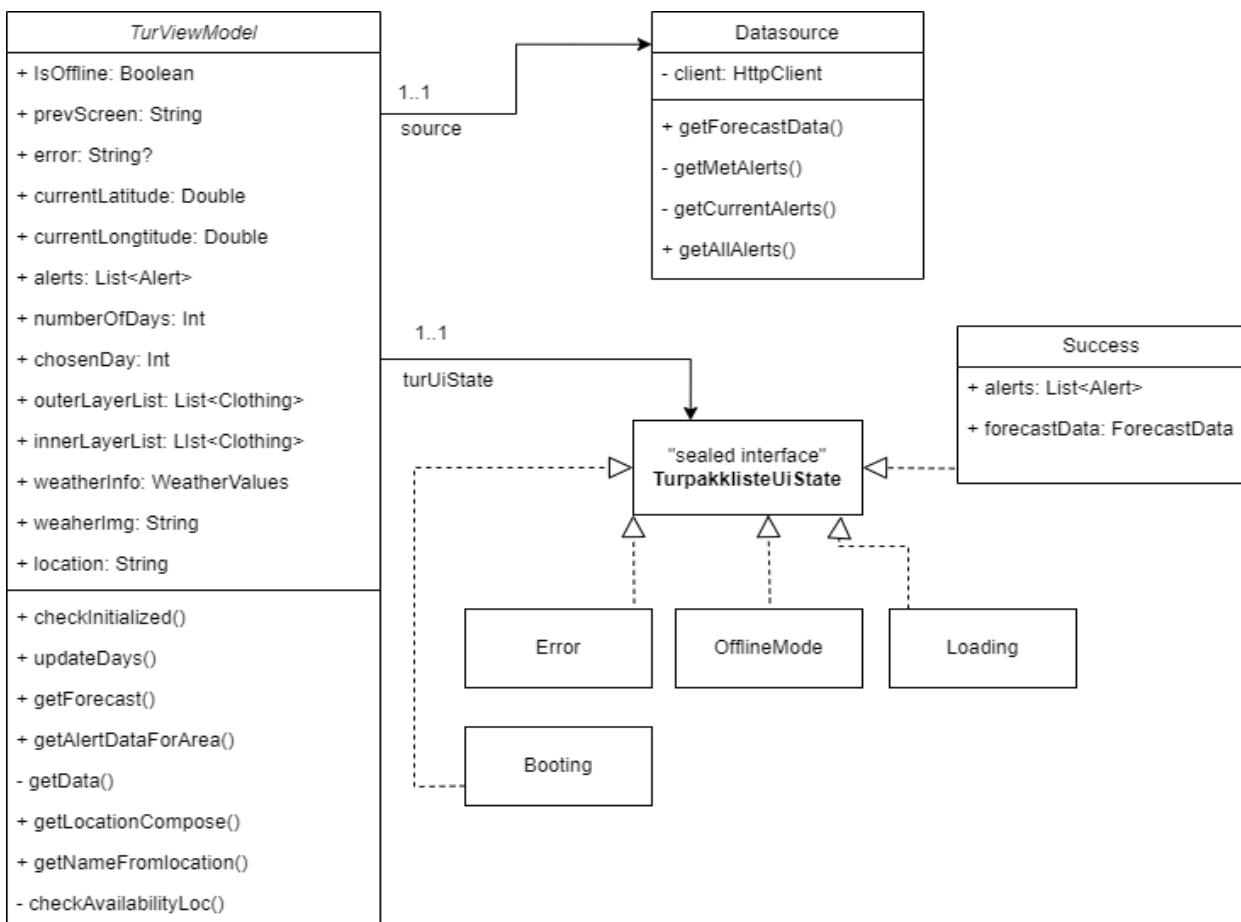
3.4 Use case - Modellering av krav

For å visuelt fremstille og spesifisere oppførselen til et system kan man benytte use case diagram. Et use case diagram er gjerne enkelt utformet, det spesifiserer konteksten til et system, fanger kravene til systemet og tar utgangspunkt i sluttbrukerens perspektiv. I tillegg er det et kommunikasjonsverktøy som kan effektivt fremstille forholdet mellom aktører som muliggjør systemet (Visual Paradigm, 2022). Nedenfor er det et enkelt use case diagram som fremstiller den tiltenkte oppførselen for deler av systemet *PackMate*:



3.5 Klassediagram

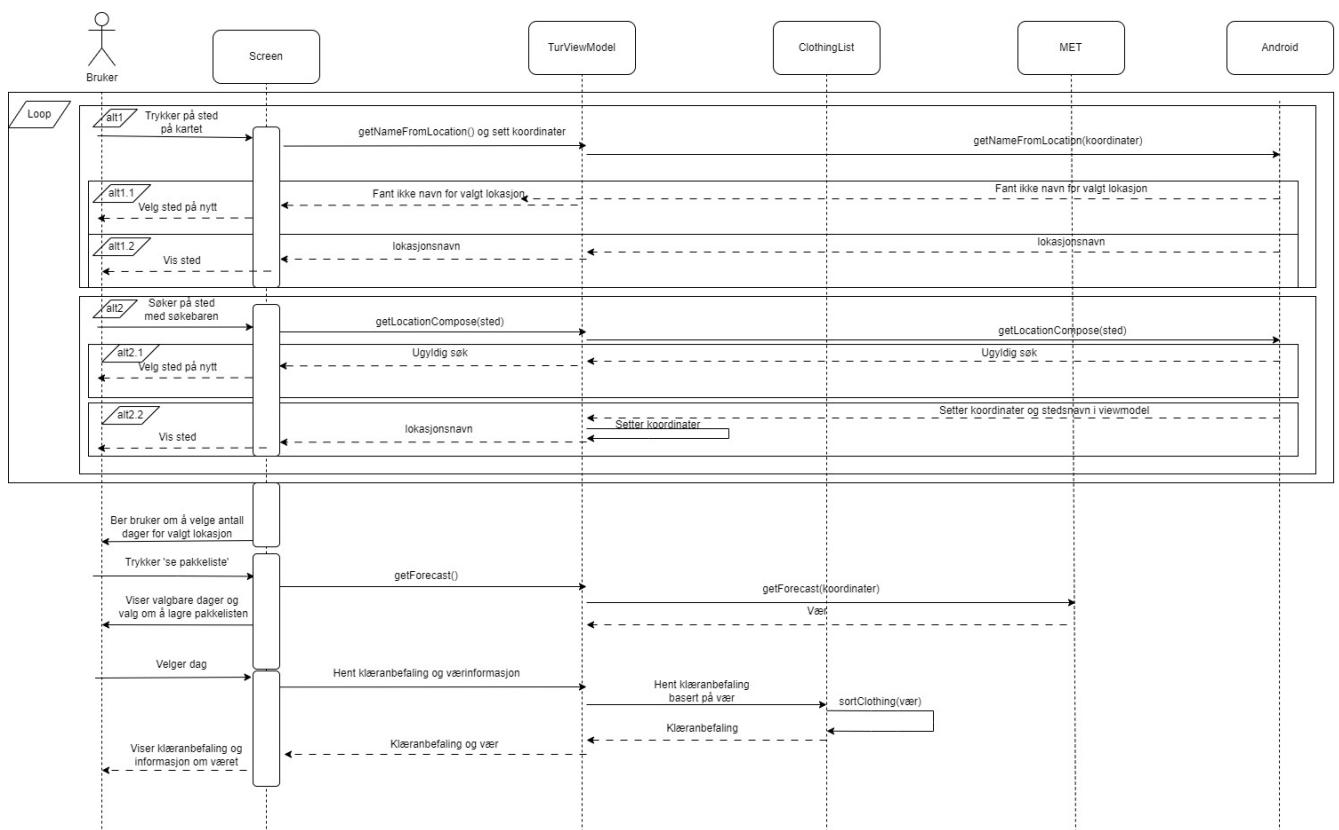
Klassediagrammer er en visuell fremstilling av sammenhengen mellom klasser, og fungerer som et supplement til tekstlig kode (NTNU, 2019). Systemet *PackMate* inneholder mange klasser og et helhetlig klassediagram blir derfor for omfattende for å kunne leses. Derimot er det fremstilt et kompakt klassediagram som viser frem noen av de mest vesentlige klassene og forholdet dem imellom:



Et mer omfattende klassediagram er vedlagt rapporten (Vedlegg: 7.3 Klassediagram).

3.6 Sekvensdiagram

Et sekvensdiagram viser hvordan objekter samhandler, eller sekvensen for metodekall mellom objekter. Sekvensdiagrammer har en implisitt tidslinje som viser forholdet mellom brukerens handlinger og samspillet mellom komponenter (NTNU, 2019). I sekvensdiagrammet har vi beskrevet bruksflyten for et use case, hvor brukeren velger sted og ser en pakkeliste for det valgte stedet. For bedre lesbarhet, er diagrammet også lagt til som vedlegg, inkludert en tekstlig beskrivelse av bruksflyten (Vedlegg: 7.1 Sekvensdiagram).



3.7 Objektorienterte prinsipper og design patterns

Kompleksitet i systemarkitektur oppstår som følge av antallet komponenter og forholdene de imellom (Sommerville, 2021, s.104). Når man skal endre et program må man forstå disse forholdene og hvordan komponentene påvirker hverandre. I arbeidet med å utforme systemarkitektur er det viktig å unngå unødvendig kompleksitet, slik at man enklere kan ivareta oversikten.

Eksplisitt kan man kontrollere kompleksiteten i systemarkitekturer ved å tenke på Separation of Concerns (Sommerville, 2021, s.105). Dette er et prinsipp som fastslår at man bør identifisere komponenter i grupper, relatert etter funksjonalitet. På et lavere nivå, innebærer prinsippet at ulike komponenter, ideelt sett, skal ha ett enkeltstående ansvarsområde.

I applikasjonen *PackMate* kan dette prinsippet konkretiseres gjennom design mønsteret MVVM (Model-View-ViewModel). Dette er et design pattern som anvendes innen utviklingen av webapplikasjoner for å mediere ansvar mellom klasser. Hensikten er å separere brukergrensesnittet (View) fra logikken i applikasjonen (Model) og presentasjonen (ViewModel) (Microsoft, 2022).

Det er også viktig å ivareta objektorienterte prinsipper slik som høy kohesjon og lav kobling. Dette er et prinsipp som er tett tilknyttet Separation of Concerns. Høy kohesjon handler om at objekter har moderat ansvar og utfører et begrenset antall oppgaver innenfor ett funksjonelt område. På sin side handler lav kobling om at objekter skal ha et begrenset antall avhengigheter. Dette er grunnleggende prinsipper for god kodeproduksjon, uavhengig av språk, rammeverk eller plattform (Runde, 2017).

3.7.1 Prinsipper for kodeproduksjon

For å ivareta grunnleggende prinsipper for god kodeproduksjon, kan disse punktene være hensiktsmessige å følge:

- Definere tydelige ansvarsområder, slik det beskrives gjennom prinsippet Separation of Concerns (SoC), men også høy kohesjon og lav kobling. I applikasjonen *PackMate* er dette spesielt konkretisert gjennom design mønsteret MVVM.
- Holde funksjoner så små som mulig. Dette gjør det enklere å kontrollere hva funksjoner gjør og hvilke avhengigheter de har. Slik øker man lesbarheten, vedlikeholdbarheten og oppmuntrer til hensiktsmessig gjenbruk av kode.
- Bruke deskriptive navn for variabler, funksjoner og klasser. Dette gjør koden mer selvdokumentert og lettleselig.
- Unngå unødvendig globale variabler. Globale variabler kan gjøre det vanskelig å forstå hvorfor kode leder til feil.

4.0 Produktdokumentasjon

Produktdokumentasjonen forutsetter grunnleggende kjennskap til *PackMate*. Det er fordelaktig å ha lest brukerdokumentasjonen før man leser produktdokumentasjonen, slik at man har et overblikk over applikasjonens bruksområde og funksjonalitet.

4.1 Oversikt – Løsningen

Til deg som skal videreutvikle eller driftet *PackMate* er følgende punkter vesentlige:

- Prosjektet er utviklet med Kotlin og Android Studio har vært det foretrukne utviklingsverktøyet.
- For å designe UI-elementene har man hovedsakelig benyttet designbiblioteket *Jetpack Compose*. I tillegg er det anvendt kode fra utgiverne av *Material 3*, som i nåtiden er den mest oppdaterte versjonen av Google sitt open-source design system (Google, 2023d).
- Det er nødvendig med API-nøkkel for å anvende Google Maps.
- Gradle har vært benyttet gjennom Android Studio for å styre avhengigheter, konfigurere innstillinger og automatisere byggeprosessen.
- Arkitekturen bygger på MVVM, som knytter sammen både front og backend.
- Prosjektet bruker minimum API-nivå 26.
- Lokal databaseløsning som bruker Room.
- Pinpoint-location-algoritmen.
- SortClothing-algoritmen.

4.2 Konkretisering

4.2.1 Kotlin

Som nevnt i oversikten er prosjektet utviklet i Kotlin, som er et kodespråk basert på Java. I den forstand er Java kompatibelt med Kotlin, men Kotlin har mindre boilerplate kode (JetBrains, 2023). For ordens skyld vil det være hensiktsmessig å fortsette utviklingen i Kotlin for å ivareta en oversiktig kodebase.

4.2.2 Jetpack Compose

I konteksten av *Jetpack Compose* har man benyttet ”composables” for UI-design. Composable refererer til en funksjon som genererer og forvalter et UI-element (Google, 2023a). Composables er å regne som fundamentale byggestener innen *Jetpack Compose* biblioteket. De er enkle, gjenbruksbare og de kan kombineres sammen for å skape mer komplekse brukergrensesnitt. I kodebasen står disse funksjonene annotert med ”@Composable”. Denne koden tar imot data som input og returnerer en beskrivelse av UI-elementet som skal fremvises. Beskrivelsen er ikke en representasjon av UI i seg selv, men en definisjon av hvordan brukergrensesnittet burde se ut, gitt visse input data. Det er å foretrekke at alt vedlikehold eller utvikling av applikasjonen benytter det samme biblioteket, fremfor XML. *Jetpack Compose* oppdateres stadig og kode kan fort bli utdatert, derfor må man aktivt sjekke det oppdaterte biblioteket (Google, 2023a).

4.2.3 Material 3

I applikasjonen *PackMate* anvendes det kode fra *Material 3*, som er den mest oppdaterte versjonen av Google sitt open-source design system. I nåtiden benytter man Version 1.1.0-alpha08, som ble utgitt 8.mars, 2023 (Google, 2023b). Det kritiske er å ikke anvende en tidligere versjon av biblioteket, ettersom det i denne oppdateringen mottok støtte for såkalte “BottomSheets”. Denne typen composable er implementert i kart-skjermen på applikasjonen, som vil tilsi at man ikke kan anvende dette UI-elementet, dersom man går tilbake i versjon. Alternativt kan det fremstilles en egendefinert composable som overtar for nåværende “BottomSheet” fra *Material 3*.

4.2.4 Google Maps

Kartet som er i applikasjonen anvender API fra Google Maps. Spesifikt representeres kartet gjennom en standardisert composable med tilnavnet “GoogleMaps”. For å kunne bruke Google Maps-API må man ha tilgang på en API-nøkkel. Dersom man går til “local.properties” som inngår i “Gradle Scripts”, skal man ha en variabel definert på formen:

GOOGLE_MAPS_API_KEY = (EGENDEFINERT API-NØKKEL)

Her skal det være en eksisterende API-nøkkel i feltet “(EGENDEFINERT API-NØKKEL)”. Dersom det ikke er en API-nøkkel i dette feltet vil ikke kartet fungere. Derimot skal det være inkludert nøkkel ettersom prosjektet er i en zippet mappe.

4.2.5 Gradle

Gradle-filene i prosjektet, har kort fortalt, ansvaret for å styre hvordan Android Studio prosjektet er bygget og innpakket. Disse filene inndeles etter ”Project-level” og ”Module-level”. Project-level filer definerer innstillingar som påvirker hele prosjektet. Det inkluderer også avhengigheter for byggesystemet. Module-level filer har innstillingar som er spesifikke for modulen eller applikasjonen som bygges, slik som avhengigheter og informasjon om versjoner. For å forvalte avhengigheter slik som implementasjonen av *Material 3*, må man inn i ”build.gradle” på Module-level. Her vil avhengigheter stå annotert under dependencies, hvor man kan forvalte ulike avhengigheter som må oppdateres eller innhentes.

4.2.6 MVVM

Arkitekturen i *PackMate* baserer seg delvis på design-mønsteret MVVM. Her må det understrekkes at man kun har én ViewModel, ved navn ”TurViewModel”. Applikasjonen i seg selv er ikke så omfattende, derfor har man bare én ViewModel, fremfor mange små. Likevel, kan det gjøre det vanskeligere å videreutvikle løsningen og på lang sikt gjøre den mindre modulær. I tillegg kan dette være i konflikt med prinsippet om høy kohesjon og lav kobling. For fremtidig utvikling bør man vurdere om det er hensiktsmessig å ha flere definerte ViewModel, for å kunne ivareta prinsipper for god kodeproduksjon.

4.2.7 API-nivå

Opprinnelig hadde *PackMate* minimum API-nivå 23. Derimot krever visse funksjoner i applikasjonen minimum API-nivå 26, derfor er dette det oppdaterte minimum API-nivået.

"Target SDK" refererer til API-nivået appen er ment til å kjøre på. For å gjøre prosjektet mer modulerbart på sikt, har appen et høyt "Target SDK" på API-nivå 33. En annen grunn for at appen har så høyt "Target SDK" er fordi at fra og med 31.august 2023, må alle apper som skal ut på Google Play Store være rettet mot API-nivå 33 (Google, 2023c).

4.2.8 Databasen

For lagring av pakkelister benytter *PackMate* en lokal database. Formålet er at brukeren skal kunne bruke deler av applikasjonen uten internetttilgang. Brukeren får valget om å lagre pakklisten etter at den er generert. Databasen tar kun vare på informasjonen som er nødvendig for å generere pakklisten på nytt. Brukeren har også, i den nåværende versjonen, kun muligheten til å slette alt i databasen, fremfor individuelle pakkelister. For fremtiden kan det være hensiktsmessig å gjøre det mulig å fjerne individuelle lister, fremfor alle lagrede lister. For å skille mellom elementene i databasen, brukes datoен som primærnøkkel. Dette kommer med en antagelse om at en bruker ikke kan dra på flere turer i løpet av det samme døgnet.

Løsningen bruker Room. Det er et innebygd bibliotek i Kotlin for håndtering av SQL-spørninger. Room er et abstraksjonsnivå over SQLite, som implementerer en frittstående SQL-databasemotor (Google, 2023e). Dette gjør det enkelt å gjøre spørninger under kjøring av applikasjonen.

4.2.9 Pinpoint-location-algoritmen

PinpointLocation er algoritmen som har ansvaret for å finne ut om den valgte lokasjonen til en bruker er innenfor området til et farevarsel. MetAlerts inneholder et polygon som indikerer området som en fare gjelder for. Algoritmen finner den laveste og høyeste verdien for lengde- og breddegraden til polygonet, som danner området faren gjelder for. Videre sjekkes det om den valgte lokasjonen befinner seg innenfor dette området. PinpointLocation blir brukt i en løkke av alerts, slik at det sjekkes om det er flere farevarsler for et område.

4.2.10 SortClothing-algoritmen

For å anbefale klær til brukeren benytter man en liste med plagg som er lagret lokalt i systemet. Her er hvert plagg lagret med unike verdier for hvilket kleslag det tilhører, varme, vannresistens og vindresistens. Algoritmen sortClothing() tar i mot en streng som bestemmer hvilket lag man skal anbefale klær for. I tillegg mottar algoritmen verdier hentet ut fra Locationforecast slik som temperatur, vind og nedbør.

Ved bruk av hjelpe-metodene chooseOuterClothingRequirements() og chooseInnerClothingRequirements(), fremstiller algoritmen minimumsverdier for kles-lagene, innerlag og ytterlag, basert på værdataen. Det fremstilles altså minimumsverdier for varme, vann og vind resistens som må oppfylles for de ulike kles-lagene. Deretter sorterer algoritmen gjennom plaggene som er lagret i hvert lag og stanser først når man tilfredsstiller minimumsverdiene.

4.3 Kvalitetsegenskaper og evaluering

Beskrivelsen av viktige kvalitetsegenskaper ved *PackMate* tar utgangspunkt i *ISO/IEC 25010* (ISO, 2022). Applikasjonen ivaretar følgende kvalitetsegenskaper:

- Functional Suitability: Gjør systemet det det skal?
- Reliability: Er det feil (bugs) i systemet?
- Usability: Er systemet greit å bruke?

4.3.1 Functional Suitability

“Functional Suitability” er i henhold til *ISO/IEC 25010* en beskrivelse av hvordan et system ivaretar funksjoner som møter uttalte og implisitte behov, under spesifikke forhold (ISO, 2023). Dette målet fokuserer på å vurdere om funksjonene i applikasjonen er hensiktsmessig utformet og imøtekommer funksjonelle krav.

Applikasjonen benytter enhetstester for å sikre at viktige funksjoner gjør som de skal og ivaretar oppførsel, selv når kodebasen oppdateres (Sommerville, 2021, s.274). Totalt er det fremstilt 12 enhetstester. Disse testene utgjør grunnlaget for å kontinuerlig utføre funksjonell testing, slik det beskrives i *Engineering Software Products - An introduction to Modern Software Engineering* av Ian Sommerville (Sommerville, 2021 , s.272). Ved å anvende ulike typer enhetstester, slik som å fremtvinge edge-cases og error-meldinger, kan man redegjøre for om viktige funksjoner fungerer.

I nåtiden er det fremstilt såkalte "Unit" og "Feature" tests (Sommerville, 2021, s.274).

Altså, enhetstestene slik de eksisterer nå, er begrensete i omfang og tar kun stilling til spesifikke deler av kodebasen. Eksempelvis er det fremstilt flere tester som tar stilling til poengsystemet som fastslår anbefalte klær, gitt spesifikke værforhold. Dette er en kjernefunksjon som er særdeles viktig å teste.

På et mer overordnet nivå eksisterer det tester for å sjekke at kjernefunksjoner samhandler slik de skal, men de er ikke automatiserte. Disse testene kan beskrives gjennom såkalte *Interaction tests* og *Usefulness tests*, hvor man forsøker å integrere funksjoner sammen og tester resultatet. Spesifikt kan man bruke enkle user stories som utgangspunkt for tester. Her er et utdrag som er tilpasset testing av funksjonelle krav for *PackMate*:

- Som en bruker ønsker jeg å kunne søke opp lokasjon for tur, men også bytte valgt lokasjon, dersom jeg ombestemmer meg.

Denne beskrivelsen er et utgangspunkt for å teste om søker-og-trykke funksjonen fungerer slik de skal på kartet, og hvordan disse samhandler. Dette kan være utfordrende å teste med en formell enhetstest, derfor er enkelte tester som dette ikke automatisert, men manuelle. For videre utvikling bør det være mer omfattende og automatiserte tester, som kartlegger hvordan funksjoner samhandler på et overordnet systemnivå.

4.3.2 Reliability

"Reliability" er i henhold til *ISO/IEC 25010* en beskrivelse av hvordan et system, produkt eller komponent utfører funksjoner under spesifikke forhold, for en spesifisert tidsperiode (ISO, 2023). Dette målet fokuserer på applikasjonens stabilitet, tilgjengelighet og robusthet under ulike forhold.

Testing og vurdering av "Reliability" i applikasjonen *PackMate* forekom avslutningsvis i utviklingsprosessen. Under utvikling har man kontinuerlig kartlagt error-meldinger som påvirker robustheten og tilgjengeligheten til applikasjonen, men en systematisert gjennomgang forekom først senere, dersom feilmeldingene ikke var kritiske eller hindret videre utvikling. Dette momentet er ytterligere redegjort for i prosess-dokumentasjonen.

For brukeropplevelsen tar applikasjonen høyde for feilmeldinger ved å gi synlige tilbakemeldinger, slik som om man ikke har tilgang til internett. For dette eksisterer det en egen error-skjerm, som beskriver den gjeldende feilmeldingen for brukeren.

For videre utvikling bør det være et fokus på å oppklare resterende feilmeldinger. Spesifikt er det identifisert en timeout-error som kan oppstå ved bruk av LocationForecast, når man søker etter lokasjon på kartet. Applikasjonen kræsjer ikke regelrett, men man blir ført til error-skjermen og må starte applikasjonen på nytt. I tillegg eksisterer det feilmeldinger for valg av lokasjon, dersom man velger lokasjon uten stedsnavn. Applikasjonen fungerer og *PackBuddy* vil gi beskjed til brukeren om at de må velge en ny lokasjon.

4.3.3 Usability

“Usability” er i henhold til ISO/IEC 25010 en beskrivelse av hvordan et produkt eller system kan brukes av spesifikke brukere, for å oppnå spesifikke mål, effektivt og tilfredsstillende, i en definert kontekst (ISO, 2023). For å forbedre usability kan det utføres brukertester der brukeren utfører oppgaver i systemet og oppdager potensielle problemer.

For å evaluere “Usability” kan man henvende seg til ressurser på “Usability.gov” (U.S. General Services Administration, 2023). Brukertesting, i flere steg av utviklingsprosessen, er grunnlaget for å sikre at software-systemer er brukervennlige og oppfyller spesifiserte krav. Ressursene fra “Usability.gov” fremhever ulike måter man kan moderere brukertester, deriblant “Concurrent probing”. Denne metoden er anvendt for å evaluere usability i *PackMate*. Spesifikt går det ut på å observere en bruker og dokumentere om de møter på utfordringer ved bruk av systemet. Under testing stiller man spørsmål for å sikre at man får hensiktsmessige tilbakemeldinger og data som kan anvendes for å forbedre brukeropplevelsen. Derav fanger man opp problemer og vanskeligheter med én gang slik at ingenting blir glemt. Konkrete resultater fra brukertester som evaluerer usability i *PackMate* er vedlagt prosjektet og utdypet i prosessdokumentasjon.

4.4 API fra Meteorologisk Institutt

I applikasjonen *PackMate* har man benyttet følgende API fra Meteorologisk institutt:

- **MetAlerts** – API fra Meteorologisk institutt med farevarsel inndelt etter regioner i Norge.
- **Locationforecast** – API fra Meteorologisk institutt med sanntids værdata inndelt etter lokasjoner i Norge.

PackMate benytter MetAlerts fordi:

- *MetAlerts* gir brukeren tilgang på viktig informasjon vedrørende lokale værforhold. Farevarsel er svært relevant og viktig informasjon fordi det kan påvirke den fysiske sikkerheten til brukeren. Et farevarsel vedrørende ekstremvær kan være grunnlag for å fraråde brukeren til å gå på tur.
- Det kan gi viktig informasjon som påvirker valg av utstyr. Farevarsel kan inngå som en del av beslutningsgrunnlaget når man anbefaler brukeren turutstyr og klær.

PackMate benytter Locationforecast fordi:

- Locationforecast gir svært nøyaktige værdata som styrker beslutningsgrunnlaget for å anbefale klær. Spesifikt gir Locationforecast tilgang til informasjon som vindhastighet, temperatur og nedbør.
- Locationforecast gir tilgang på værdata for flere dager frem i tid. Dette gir brukeren mulighet til å planlegge lengre turer som kan vare over flere dager. Flerdagers værvarsel sikrer at anbefalingene i pakkelisten har et sterkere beslutningsgrunnlag. Dermed kan PackMate fremstille mer treffende anbefalinger for pakkelisten.

4.5 Utfordringer – API fra Meteorologisk Institutt

Rent teknisk er det få utfordringer med MetAlerts og Locationforecast. Derimot er det mest vesentlige, for den videre driften av *PackMate*, at man vurderer en overgang fra den kompakte versjonen av Locationforecast, til den komplette. Dette er for å oppfylle deler av den funksjonelle kravspesifikasjonen som ikke er oppfylt i den nåværende versjonen av *PackMate*. Den kompakte versjonen av Locationforecast har ikke data slik som UV, men det eksisterer i den mer omfattende komplette versjonen av Locationforecast.

5.0 Prosessdokumentasjon

I prosessdokumentasjonen skal vi beskrive det overordnede arbeidet som ledet frem til *PackMate*. Her redegjør vi for hvilke valg som ble tatt underveis, hva som fungerte og hva som ikke fungerte.

Først skal vi beskrive de formelle rammene for prosjektet, som vi i gruppen ble enige om i oppstartfasen. Deretter vil vi redegjøre for erfaringene våre med å fremstille de ulike iterasjonene av *PackMate*. Til slutt, vil vi redegjøre for hvordan vi har oppfattet prosjektarbeidet og hva vi har lært, i en kortfattet konklusjon.

5.1 Formelle rammer for prosjektet.

Lørdag 4. Mars 2023 hadde gruppen sitt første møte i forbindelse med kick-off for årets IN2000-prosjekter. Dette arrangementet var en god mulighet for å bli kjent med hverandre og det la grunnlaget for resten av prosjektarbeidet.

Innledningsvis ønsket vi å skape en god tone og stadfeste hva vi forventet av hverandre. Dette ble formalisert gjennom team-avtalen vår som er vedlagt prosjektoppgaven (Vedlegg: 7.10 Teamavtale). Som en gruppe var det svært viktig for oss å ha grundige samtaler i forkant av prosjekt-spesifikt arbeid slik som diskusjon om vision eller koding, for å unngå uklarheter og mulige konflikter.

5.1.1 Hvor ofte skal teamet møtes?

Som en gruppe hadde vi faste fysiske møtetidspunkt på mandag, onsdag og fredag. Deriblant var fredagsmøtene lengre, med fokus på korte retrospektiver.

5.1.2 Hvordan skal oppgavene og rollene i teamet fordeles?

I team-avtalen fremstilte vi en uforpliktende rolleinndeling basert på ønsker og selvoppnevnte egenskaper (Vedlegg: 7.10 Teamavtale). Grunnen til at vi betegnet rolleinndelingen som uforpliktende er fordi vi ønsker at folk skal få erfaring med forskjellige arbeidsoppgaver i løpet av prosjektet. Rolleinndelingen er førende med tanke på ansvarsområder, men den er ikke eksklusiv. Det betyr at ingen har enerett på å utføre visse arbeidsoppgaver som programmering, design eller rapportskriving. Det viktigste med prosjektet er å skape en arena som gir rom til hvert eneste medlem for å lære. I praksis fikk teammedlemmene prøvd seg på mange ulike deler av prosjektet, selv om vi hadde hver vår rolle.

5.1.3 Valg av smidig-arbeidsmetodikk

Kjernen i prosjektarbeidet er den smidige arbeidsmetodikken Scrumban. Det er flere grunner for at vi valgte Scrumban:

- For det første, er Scrumban en kombinasjon av Scrum og Kanban (ProductPlan, 2023). Det betyr at vi hadde muligheten til å tilegne oss erfaring med en spennende hybrid arbeidsmetodikk.
- For det andre, så var det flere aktiviteter i prosjektet som skulle foregå parallelt. For å holde oversikt over arbeidsoppgavene og mulige flaskehalsar ønsket man å fremstille prosjektfremgang visuelt. Kanban har et fokus på arbeidsflyt og visuelle fremstillinger av prosjektfremgang. Scrum har på sin side et fokus på rolleinndeling og tidsperioder for arbeid. Scrumban sammenfatter strukturen og forutsigbarheten til Scrum, med Kanban sitt fokus på fleksibilitet og arbeidsflyt. Dette var en balanse som vi forsøkte å tilstrebe.

I prosjektarbeidet erfarte vi at man ikke alltid følger rammene som er fastsatt, slik som å følge Scrumban rammeverket til punkt og prikke. Uforutsette problemer, ulike timeplaner blant teammedlemmer og nye erfaringer førte til at vi kontinuerlig tilpasset hvordan vi arbeidet. Likevel, ønsker vi å spesifisere hva Scrumban skulle innebære for gruppen, uavhengig av hvordan det ble fulgt opp.

Spesifisering av Scrumban

Nedenfor er det en visuell fremstilling av stegene som inngår i Scrumban, på et overordnet nivå:

How Does Scrumban Work?

- 1** Develop a Scrumban board
- 2** Set your work-in-progress limits
- 3** Order the team's priorities on the board
- 4** Throw out your planning-poker cards
- 5** Set your daily meetings

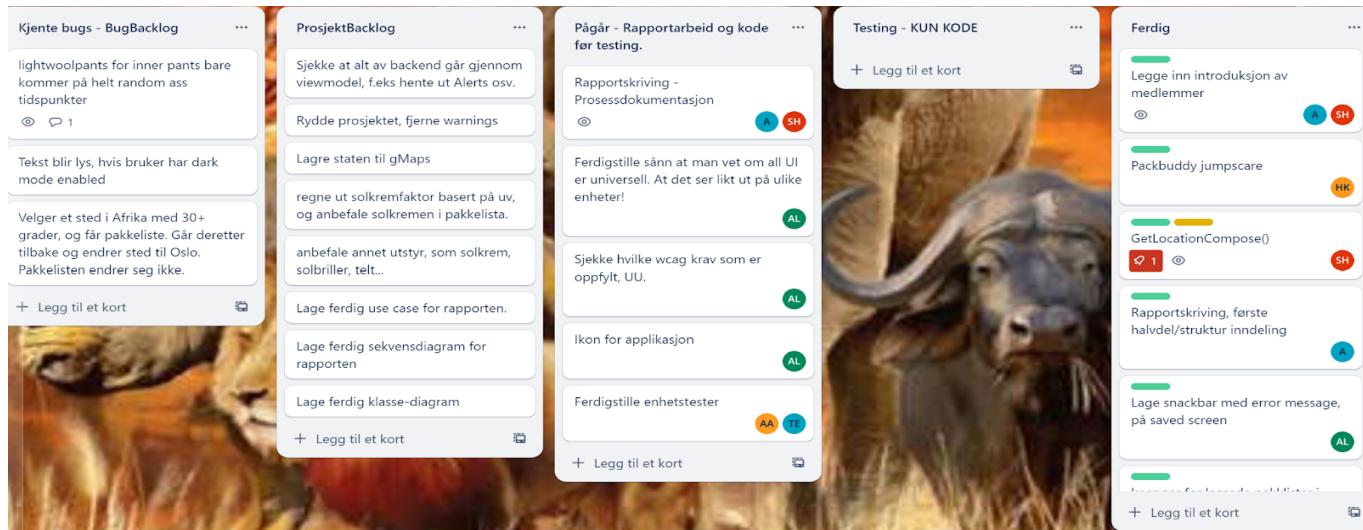


(ProductPlan, 2023)

Scrumban involverer å anvende Kanban prinsipper – som visualisering av arbeid og arbeidsflyt, kombinert med deler av Scrum rammeverket. I tillegg fjerner Scrumban noen av de mest rigide elementene fra Scrum og tillater hvert team å skape en unik fremgangsmåte for utvikling (ProductPlan, 2023).

Scrumban brettet

Et Scrumban brett er liknende et Kanban brett hvor man fremstiller backloggen for prosjektet. Det brukes som et verktøy for å ivareta arbeidsflyt, derfor legger man til så mange kolonner som er hensiktsmessig, for å markere ulike steg av utviklingsprosessen. Nedenfor er et bilde som viser vårt Scrumban brett:



Trello-board 10. Mai 2023

Work-in-progress limit

Scrum fastsetter tidsbegrensninger for hver sprint. Kanban har på sin side et fokus på kontinuerlig arbeidsflyt. I Scrumban er det viktigst å etablere en arbeidsbegrensning, som i vårt tilfelle var på "to kort" eller arbeidsoppgaver, som man kan påta seg fra brettet samtidig. Det er vanskelig å estimere hvor omfattende en oppgave ville være, derfor satte vi dette tallet relativt lavt.

Prioritering av oppgaver

Scrumban fokuserer på at man etablerer prioriteringer for arbeidsoppgaver på brettet. Teamet vil kontinuerlig avgjøre hvilke personer som burde håndtere de ulike oppgavene.

Møteordning og tidsbegrensninger

Selv om Scrumban formelt sett ikke har sprintplanlegging, reviews eller retrospektiver, ønsket vi å ha fastsatte øyeblikk for å betrakte arbeidet som har skjedd og som skal utføres. Derfor hadde vi korte retrospektiver på fredagsmøtene for å diskutere hvordan ukens arbeid hadde gått.

I tillegg opplevde vi, gitt tidsbegrensningene for prosjektet, at man burde ha en viss tidsbegrensning for deler av prosjektarbeidet, slik at man kan sikre kontinuerlig fremgang. Derfor definerte vi et fåtall av tidsbegrensete arbeidsperioder som omfattet en fremstilling av MVP, ferdigstilling av kritiske kjernefunksjoner og en lengre vedlikeholdsperiode. Dette blir fremvist tydelig i prosjektplanen nedenfor.

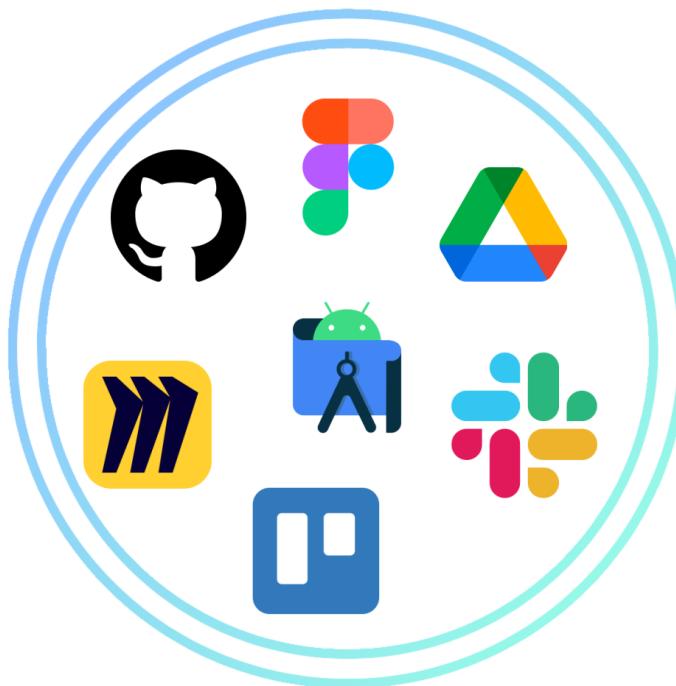
Scrumban oppsumering og formelle rammer

Vi har få faste tidsfrister å forholde oss til i prosjektet, foruten fremstillingen av MVP, ferdigutviklet applikasjon og rapport. Derfor skal man kontinuerlig prioritere arbeidsoppgaver visuelt og fordele oppgavene våre etter en uformell rolleinndeling. I henhold til Scrumban vil ikke arbeidet foregå i sprinter, men fokusere på arbeidsflyt og prioritering av oppgaver. Fokuset er å overholde begrensninger for hvor mye arbeid teammedlemmer kan påta seg innen de overordnede tidsfristene, samtidig som vi gjennomfører retrospektiver for utført arbeid, og innehar en uformell rolleinndeling for prosjektarbeidet.

5.1.4 Hvilke verktøy ble benyttet?

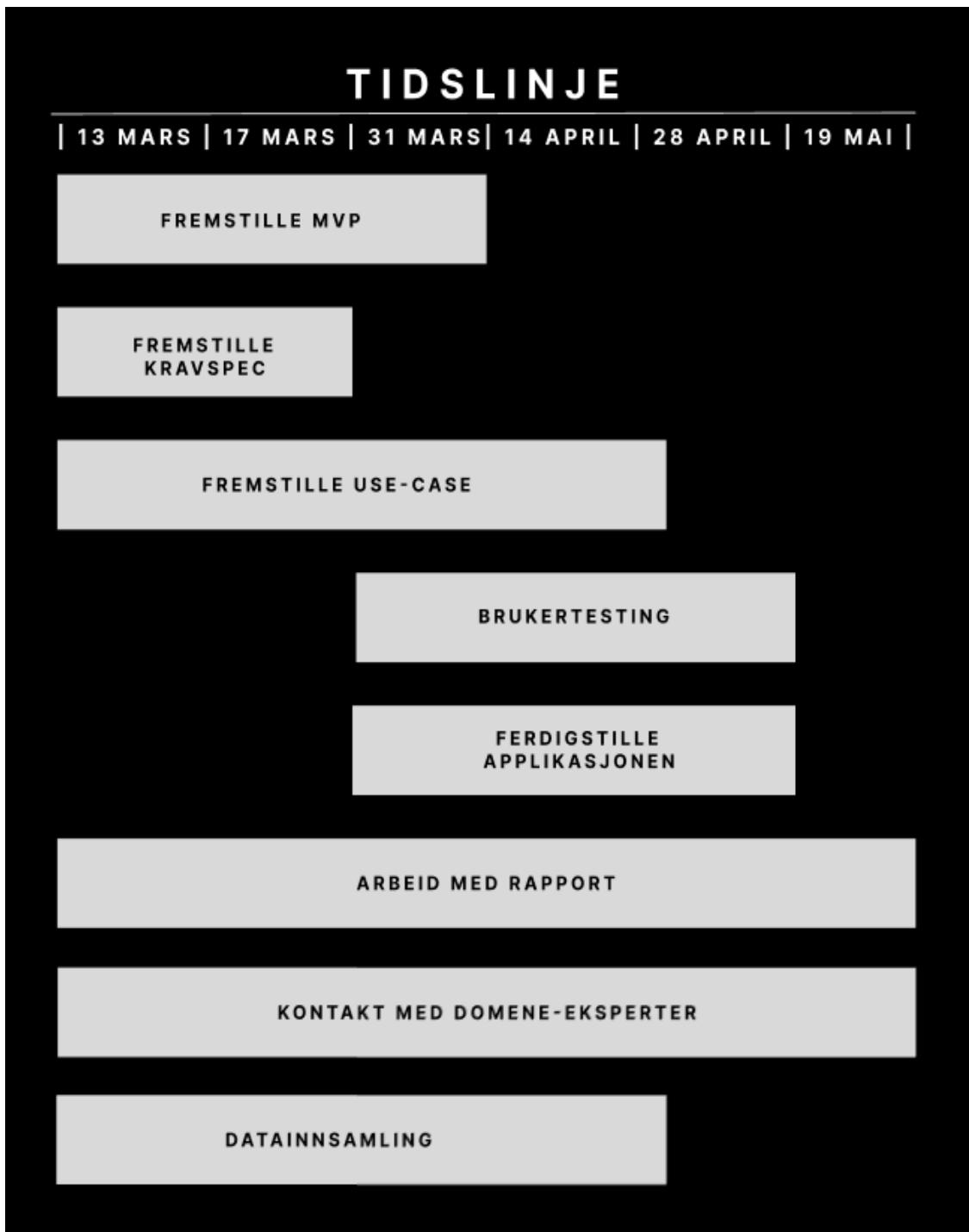
I prosjektarbeid benyttet man følgende verktøy:

- Figma: Designverktøy for å fremstille prototyper.
- Trello: Arbeidsbrett for visuell fremstilling av arbeidsoppgaver.
- GitHub: Verktøy for å dele kode mellom teammedlemmer.
- Android Studio: Utviklingsverktøy (IDE) for å kode prosjektet.
- Slack: Det foretrukne kommunikasjonsverktøyet for gruppen.
- Google Drive: Plattform for å dele relevante dokumenter, slik som prosjektrapporten.
- Miro: Verktøy for visuelle fremstillinger. Det ble benyttet for å gjennomføre retrospektiver og planlegging.



5.2 Prosjektløp

5.2.1 Opprinnelig prosjektplan

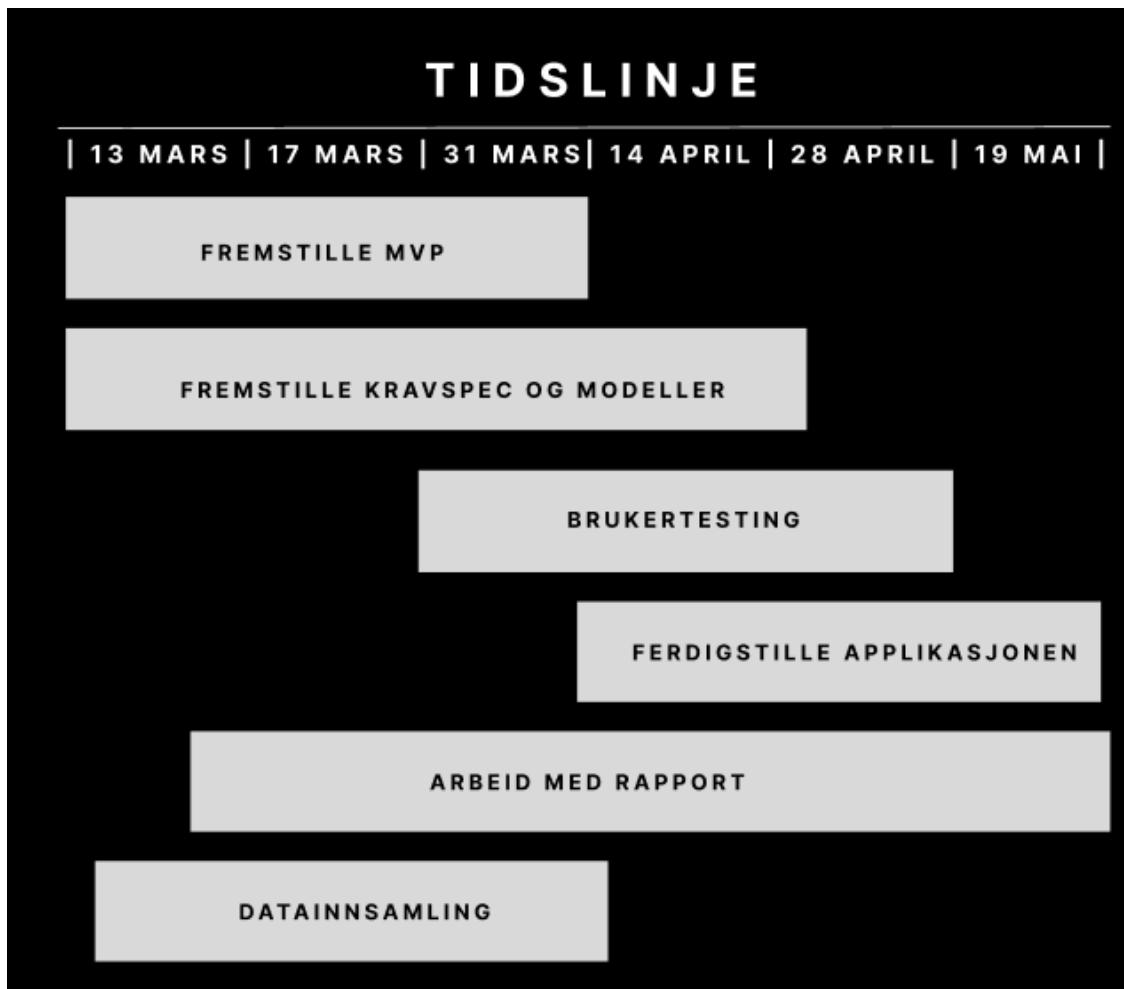


Prosjektplanen slik den ser ut ovenfor ble fremstilt i oppstartsfasen av prosjektet. Den fremstiller flere av kjerneaktivitetene og tidsfristene som man planla for:

- Vi hadde som mål å fremstille en MVP innen 31 mars. Parallelt med å lage MVP skulle vi fokusere på å tilspisse applikasjonen vår, både med tanke på funksjonalitet, men også design.
- Som et overordnet mål så vi for oss å ha en ferdigutviklet applikasjon innen 28.april. Denne selvpålagte tidsfristen eksisterte for å kunne fokusere på rapporten i slutfasen av prosjektet.

Ettersom teamet har en større faglig tyngde innen programmering enn design, ønsket vi å jobbe konsentrert i oppstarten for å bli kjent med viktige designprinsipper og verktøy. Konkret ønsket vi å forbedre forståelsen vår for rammeverket WCAG 2.1 og aktivt fremstille prototyper innen Figma. Designtenking og et tverrfaglig fokus var spennende å utforske, særlig med tanke på at vi ikke har noen spesifikk design-erfaring.

5.2.2 Gjennomført prosjektplan



Denne nye prosjektplanen er en ørlig vurdering av hvordan prosjektarbeidet faktisk gikk for seg. Slik man kan tyde, gikk ikke alle iterasjonene for seg slik som planlagt og tidsfrister ble forflyttet. Nedenfor vil vi ta for oss de ulike momentene i prosjektplanen og redegjøre for endringene som fremkom gjennom iterasjonene.

5.3 Iterasjonene

5.3.1 Pre-iterasjon - Oppstart

Vi startet sterkt og vi bestemte oss for *Case 4, åpent case- i lufta* uten noe kragling.

Mot slutten av kick off-dagen hadde vi satt opp kommunikasjonskanal på Slack, fastsatt møtetidspunkt på mandag, onsdag og fredag, og vi ble enige om pizza-straff som sanksjon, dersom man kom for sent til møter. I tillegg ble vi enige om den smidige arbeidsmetodikken Scrumban, som skulle være selve kjernen i prosjektarbeidet.

Mandag 06.03.2023 markerte starten på utvikler-reisen vår. Det at vi har fire prosa-studenter i gruppen påvirket oss alle. Det kriblet i fingrene etter å kode, men først måtte vi vite hva vi skulle lage. Derfor gjennomførte vi presentasjoner av app-ideene våre, i PowerPoint-format, foran gruppa. Etter de 45 minuttene vi hadde satt av for presentasjoner og diskusjon, endte vi opp med et valg mellom en flyvær/spill-app eller en pakkliste-app. Det var heftige diskusjoner og sterke meninger, derfor ble vi ikke enige på dagen. Vi la frem ideene for veilederne senere i uken og begge idéene ble godkjent. Dette hjalp lite for å løse konflikten, men etter at vi hadde vurdert hvor gjennomførbare de ulike ideene var, endte vi opp med å velge pakkliste-applikasjonen. Nedenfor er et utdrag fra én av de glødende PowerPoint appellene, til fordel for å lage pakkliste-applikasjonen:



5.3.2 Iterasjon 1

Hensikt og mål

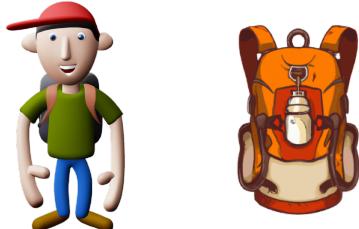
Tidsperioden for første iterasjon var fra 13.03 til 31.03. Hovedfokuset for første iterasjon var å ferdigstille en MVP og tilspisse visjonen for *PackMate*, slik at vi hadde en klar retning for resten av prosjektet. Vi begynte tidlig med design og prototyping i Figma, før vi begynte å programmere. Her måtte man eksperimentere mye og forsøke så godt man kunne å ta utgangspunkt i den tiltenkte brukergruppen vår. Backloggen ble kontinuerlig revidert gjennom iterasjonene, men på dette tidspunktet var det en svært enkel backlog, som i stor grad fokuserte på å få grunnleggende funksjonalitet i applikasjonen til å fungere. Deriblant ønsket vi å lage et fungerende kart, gi brukeren værmeldingen og farvarsels, i tillegg til en enkel anbefaling av klær for en dagstur.

Prototyping

Den første prototypen i Figma beskrev en enkel brukerflyt for *PackMate* og inneholdt hovedfunksjoner som vi anså som nødvendige for en ferdig kodet MVP. Tross alt, skal MVP være et bevis på at ideen vår er teknisk gjennomførbar. Prototypen i Figma besto av en kart-skjerm, værvarsel-skjerm og en pakkliste-skjerm for klær. Den fullstendige prototypen er avbildet i vedlegg med alle skermene (Vedlegg: 7.5 Prototype Iterasjon 1 og 7.6 Prototype Iterasjon 2).

På dette stadiet var det en spennende utfordring om hvordan vi skulle fremstille anbefalinger av klær for brukeren. Skulle det være en enkel liste, eller en spennende visuell fremstilling?

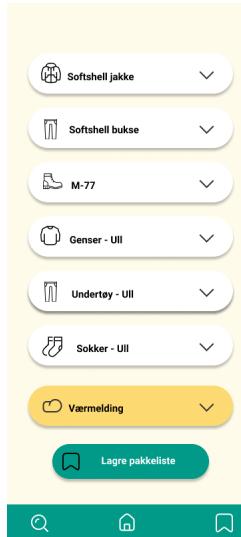
Vi lekte oss med AI-generasjon av kunst på tjenesten Dall-E 2 og fremstilte en avatar, som kunne være en form for mannequin for aktuelle klær i *PackMate* (OpenAI, 2023). Tanken var at vi også kunne AI-generere klær til avataren og på den måten spare tid. Derimot var dette vanskelig å gjennomføre i praksis. Etter flere mislykkede forsøk på AI-generasjon av en avatar, ble vi trollbundet av denne vakre avataren, som vi døpte *PackBuddy*, og vi fikk skapt denne logoen til *PackMate*:



De andre mislykkede avatarene våre er vedlagt prosjektet (Vedlegg: 7.9 Mislykkede avatarer). Det vi ønsket å gjøre var å kle på *PackBuddy*, noe som er tydelig fra den første kles-skjermen i Figma prototypen:



Dette er slik kles-skjermen ble fremstilt i den siste Figma prototypen, som gjaldt for MVP:



Hovedmomenter

Hovedoppgavene for å fremstille MVP var:

- Implementasjonen av kart ved hjelp av Google Maps.
- Lage klesalgoritmen for å velge hensiktsmessige klær.
- Kommunikasjon med API-ene fra Meteorologisk institutt.

Fordi vi var ivrige etter å se appen få ferdig funksjonalitet, var vi svært løsslupne med å teste applikasjonen vår grundig, før vi ”pushet og committed” individuelle endringer til kodebasen. Dette skapte teknisk gjeld for senere iterasjoner og et betydelig etterslep for å ha en teknisk robust applikasjon.

Google Maps

I MVP valgte vi å implementere Google Maps ved hjelp av XML, til tross for at vi hadde som mål å skape UI-elementer kun ved hjelp av Jetpack Compose. Dette gjorde vi først fordi vi fant flere ressurser på internett for å fremstille en XML-versjon av kartet, enn Jetpack Compose. Likevel endret vi på dette i senere iterasjoner, til tross for en stiavhengighet, som skapte betydelig teknisk gjeld. Vi merket at XML-versjonen hadde mye utilsiktet oppførsel som var vanskelig å ta høyde for. Dette momentet blir ytterligere utdypet i neste iterasjon.

Kles-algoritmen

For å lage kles-algoritmen bestemte vi oss for å lage et poengsystem basert på offentlig tilgjengelige ressurser fra selskapet Norrøna (Norrøna, 2023). Vi anså dette som et naturlig valg ettersom vi ikke er domeneeksperter, verken på materialegenskaper eller turklær generelt.

En vurdering som vi ikke har tatt godt nok stilling til og som utgjør en betydelig svakhet for en universelt anvendelig pakkeliste-applikasjon, er at man ikke har bestemt klær ut fra kjønn og vekt. Menn og kvinner har ulik kjernetemperatur som kan påvirke hvilke klær som er hensiktsmessig å ha på seg i ulike vær- og vindforhold (Pfizer, 2023). Kroppsvekt og fettprosent kan også sterkt påvirke kroppstemperaturen og med det hvilke klær man bør ha på seg (Speakman, 2018). I tillegg har man kun fremstilt et innerlag og ytterlag av klær for brukeren, men det mest hensiktsmessige, dersom vi hadde hatt mer tid, ville vært å ta større høyde for *flerlagsprinsippet*. Flerlagsprinsippet er et begrep som særlig anvendes i Forsvaret, hvor man vektlegger en sammensetning av flere lag med

klesplagg for å dekke varme-og beskyttelsesbehov. Plaggene deles inn i tre lag; det innerste laget, det mellomste laget og det ytterste laget:

Flerlagsprinsippet

- | | | |
|--------------------|---|---------------------|
| 1. Innerste laget | → | fuktighetstransport |
| 2. Mellomste laget | → | isolasjon |
| 3. Ytterste laget | → | mot vær og vind |
| Forsterkningsplagg | → | ekstra isolasjon |

(Forsvaret, 2009)

Det mellomste laget er svært viktig, men dessverre ble det ikke inkludert i senere iterasjoner. Likevel har det vært hensiktsmessig for prosjektet, gitt tidsbegrensningene, å holde seg til verdiene fra Norrøna og fremstille en blanding av både mannlige og kvinnelige klær, som blir anbefalt om hverandre.

API-valg

I applikasjonen benyttet man Locationforecast og MetAlerts fra Meteorologisk institutt. Det var relativt uproblematisk å implementere disse i applikasjonen. Vi utformet en algoritme, relativt kjapt, som vurderte hvor i landet man hadde farevarsel. Locationforecast derimot, opplevde vi noe utfordringer med, for å få det til å samhandle med algoritmen vi hadde utformet for å velge klær. Det var en del error-meldinger som ble kartlagt, men de var ikke kritiske og hindret ikke videre utvikling. Vi skjønte også at det var enklest for oss å benytte en kompakt versjon av data fra Locationforecast, men da manglet vi data som var viktige for å oppfylle deler av kravspesifikasjonen, slik som UV for å anbefale solkrem. Dette

var en avveining som kan føre til teknisk gjeld ved senere utvikling, men for prosjektet i seg selv, var det relativt uproblematisk.

Møtevirksomhet og samarbeid

Vi oppfattet dette som en god første iterasjon, men arbeidsperioden bar preg av at vi ikke fulgte Scrumban reglene. Vi benyttet Scrumban-brettet, men vi ble noe revet med, folk glemte work-in-progress-limits og kortene i Scrumban-brettet ble ikke alltid flyttet til hensiktsmessige kolonner. I tillegg hadde vi det veldig gøy på de fysiske møtene, men kanskje litt for gøy. Vi var effektive til å programmere, men i møtene våre manglet vi definerte tider for å programmere, ha pauser eller diskutere prosjektarbeid. I det siste retrospektivet som vi hadde for iterasjon 1 ble vi enige om at man måtte ha tydeligere definerte tidsinndelinger for møtene våre (Vedlegg: 7.8 Siste retrospektiv – Iterasjon 1).

5.3.3 Iterasjon 2

Hensikt

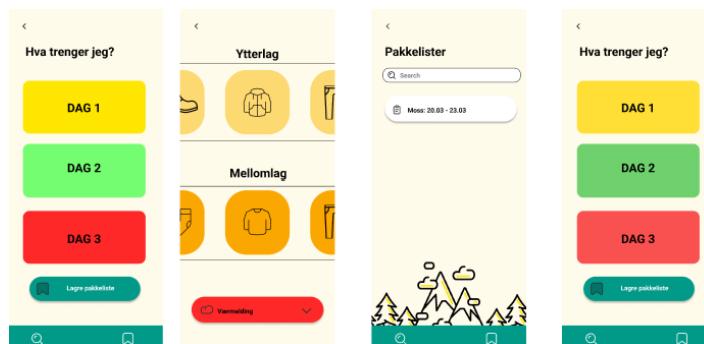
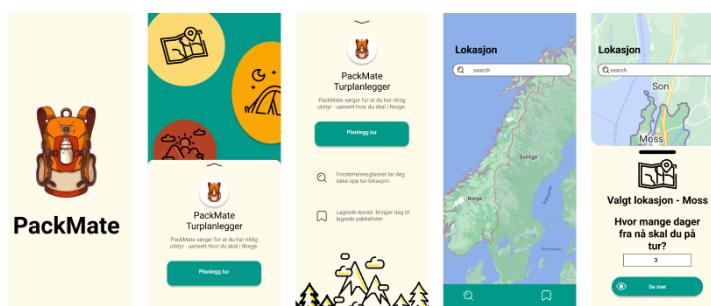
Til denne iterasjonen hadde vi en ferdig MVP som fundamentet for applikasjonen. Målet for denne iterasjonen var å fremstille en applikasjon som oppfylte flere av de funksjonelle og ikke-funksjonelle kravene i spesifikasjonen. Tidsperioden for iterasjon 2 ble satt fra 10. April – 28. April. Her måtte vi ta hensyn til at teamet kom til å ha redusert kapasitet til å jobbe med prosjektet i store deler av mai – på grunn av individuelle timeplaner.

Prototyping

Mellan forrige iterasjon og den inneværende var det påskeferie. I ferien ble det gjennomført brukertester blant familie og venner hos gruppemedlemmene, med utgangspunkt i MVP. Ut av disse fikk vi mange nyttige tilbakemeldinger om design og funksjonalitet. Tilbakemeldingene var blant annet:

- Mangelfullt og lite intuitivt design.
- Problematiske feil i applikasjonen som ødela for brukeropplevelsen.
- Lite tilfredsstillende funksjonalitet – gjennom akseptansestest.
- Ideer til nyttige funksjoner.

Disse brukertestene er ytterligere konkretisert i Vedlegg: Brukertesting iterasjon 2. Disse testene bidro til at vi endret deler av kravspesifikasjonen og utformingen av prototypene våre i Figma. Vi gikk gjennom to ulike prototyper for iterasjon 2, som begge er vedlagt for prosjektet (Vedlegg: 7.6 Prototype iterasjon 2). Avbildet nedenfor er den siste og gjeldende Figma-prototypen for PackMate:



Hovedmomenter

Denne iterasjonen var påvirket av mye teknisk gjeld. Overordnet var de viktigste momentene:

- Tilpasning av algoritmer for å støtte turer over flere dager.
- Overgangen fra XML til Compose, for Google Maps.
- Implementasjon av database for å lagre pakkelister lokalt.

Tilpasning av algoritmer

Vi oppdaget at dataen til complete-versjonen av Locationforecast hadde minimum og maks verdier, for både temperatur og nedbør i løpet av en dag. Dette hadde vi ikke tilgang til ettersom vi brukte den kompakte versjonen av Locationforecast og dette kunne ha optimalisert algoritmen vår. Likevel valgte vi å ikke bytte versjon for å spare tid og prioritere andre oppgaver.

Med utgangspunkt i hvor lenge værdata fra Meteorologisk institutt forblir nøyaktige, valgte vi å begrense lengden på turer som brukere kan dra på, til maks 3 dager.

Google Maps-implementasjon.

Underveis i iterasjonen bestemte vi oss for å implementere Google Maps med Jetpack Compose, fremfor XML. Dette ledet til en diskusjon i teamet, hvor vi drøftet følgende fordeler og ulemper:

Fordeler

- Google maps gjennom jetpack compose blir kontinuerlig oppdatert. Dette kan gi fordeler som at den er mer effektiv og letthåndterlig over tid.
- Om alt det visuelle i appen er laget ved hjelp av Jetpack Compose, vil vi redusere kompleksiteten for utvikling i fremtiden.
- Til tross for en allerede-implementert Google Maps, var det noen andre utfordringer som vi kunne mitigere dersom vi fikk byttet. Eksempelvis var det utfordringer med å manipulere markørene på kartet med XML.
- Google Maps, som både XML og jetpack compose, implementeres modulært. Dette tyder på at man kan endre disse uten å risikere at andre deler av programmet blir påvirket.

Ulemper

- Om vi skulle bytte, ville det bety å legge inn ressurser i dette fremfor noe annet. Vi hadde en XML-versjon som allerede fungerte i applikasjonen.

Etter at én i teamet utforsket compose-versjonen en kort stund, lærte vi at å bytte til Jetpack compose, ikke ville være veldig ressurskrevende. Dette ble derfor gjort fortløpende og relativt sømløst.

Valg av database

Vi ønsket å tilby brukerne en måte å se tilbake på tidligere pakkelister. Vi mente dette var passende ettersom man ikke alltid har tilgang til internett på tur. For å få til dette implementerte vi en database med "Room". Denne lagret informasjon om vær for gitte dager, slik at sortClothing()-algoritmen kunne kjøres og anbefale klærne til brukeren.

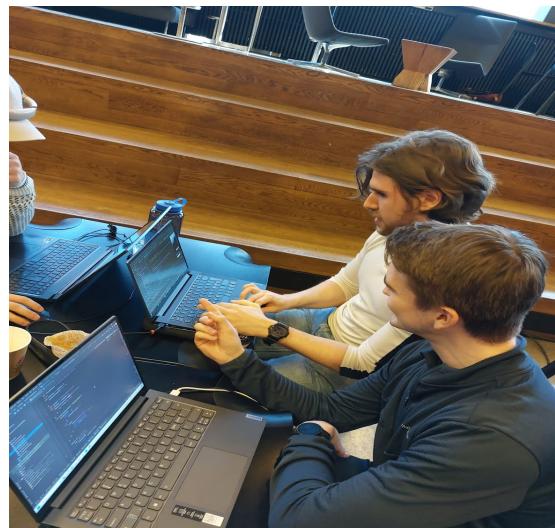


Her diskuterte vi hva som var en hensiktsmessig måte å lagre pakklistene våre som databaseobjekter. Ved hjelp av datamodellering og en tydelig visualisering av problemet, kom vi frem til en hensiktsmessig løsning.

Parprogrammering og bugfiksing

Mye av arbeidet i denne perioden bestod av parprogrammering. Dette var en effektiv måte å oppdage skrivefeil og logiske brister ved hverandres kodesnutter. Dette ble benyttet ved flere anledninger og var spesielt nyttig når flere personer hadde jobbet på ulike deler av én løsning. Når vi kombinerte kunnskapen vår fikk vi løst problemer mye mer effektivt.

Her jobber Aron og Sander med å sikre en sømløs overgang fra Google Maps med XML til Jetpack Compose. Dette fikk vi til ved å kombinere kunnskap som begge satt med om de ulike versjonene.



Forklaring av bildet: "Who let them cook?!" sier Alexander i skrek, etter at gutta pusher en algoritme med kjøretid $O(2^n)$ direkte til master.

Parprogrammering var også en essensiell del av arbeidsprosessen når vi skulle merge ulike versjoner av kodebasen. Her var det kritisk at man forsto hvilken kode som skulle inngå i applikasjonen, derfor var det hensiktsmessig at dette var en prosess som ble gjort i plenum. For små endringer kunne man "pushe og comitte" relativt problemfritt, men ved større endringer, slik som implementasjonen av nytt kart eller database, måtte vi alle ha oversikt.

Teamarbeid og møtevirksomhet

I denne iterasjonen hadde teamet blitt mer kjent med hverandre og vi ble mer vant til møtevirksomheten. Gjennom tidligere retrospektiver, hadde vi lært at man måtte ha tydeligere skiller mellom koding, pauser og diskusjon. Dette ble fulgt opp passe godt. Derimot hadde vi også hatt utfordringer med å teste koden vår godt nok. Enhetstestene våre var ikke ferdigstilt, som betød at testene våre for å evaluere kode var veldig uformelle og ikke automatiserte. Dette førte til et ytterligere teknisk etterslep som vi måtte ta igjen i vedlikeholdsperioden vår, iterasjon 3.

5.3.4 Iterasjon 3

I forrige iterasjon skulle appen ha blitt ferdigstilt og planen var å legge den fra seg for å fokusere på rapportskriving. Vi fant fort ut at dette var lettere sagt enn gjort. På mange måter er denne siste iterasjonen med kodearbeid et resultat av overhead-kostnader og teknisk gjeld. Derfor måtte vi dele fokuset vårt mellom rapport og koding. En god blanding av oppfylte krav, bugs og ideer gjorde at vi kontinuerlig gikk tilbake til appen for vedlikehold.

Vedlikehold

Hovedfokuset for denne iterasjonen var å finpusse appen, sørge for at den oppfyller kravene for vurdering i IN2000, at den er robust og oppfyller det viktigste i kravspesifikasjonen. Vi gikk systematisk gjennom denne og gjorde overordnede vurderinger om hva som var realistisk å få til, gitt den begrensede tiden som vi hadde igjen. Dette er utgangspunktet for kravspesifikasjonen som ble vist innledningsvis i prosjektet.

De resterende enhetstestene ble ferdigstilt og testet mot prosjektet vårt for å sikre at man tar høyde for ulike edge-cases. Disse testene for “Functional Suitability” og “Reliability” ble først automatisert i vedlikeholdsfasen, mens testing av dette var mer uformelt under tidligere utvikling. “Reliability”, slik det var påpekt i produktdokumentasjonen, var et større fokus i denne vedlikeholdsperioden. Vi hadde en systematisk gjennomgang av dokumenterte feilmeldinger som ble oppdaget i utviklingsprosessen. Disse var organisert i Trello, i en egen kolonne ved navn ”Kjente bugs- Backlog”. Foruten brukertestene som ble gjennomført i iterasjon 2, forsøkte vi å ivareta ”usability” ved å systematisk gjennomgå de ulike WCAG 2.1 kravene fra spesifikasjonen og vurdere om de var oppfylt. Særlig benyttet vi kontrast, farge og tekst sjekkere på internett for å sikre at man ivaretok ulike WCAG 2.1 krav (WebAIM, 2023).

TurpakklisteUiState ble utvidet til å innebefatte ‘error’ og ‘offlineMode’.

Om applikasjonen ikke klarte å hente data, ville den i uiState bli i tilstanden error. Dette sendte brukeren videre til ErrorScreen.

ErrorScreen ble laget med hensikten om å dekke utilsiktet oppførsel av PackMate som for eksempel å generere en pakkliste uten internett. Om brukeren startet appen uten å være tilkoblet til internett ville uiState bli oppdatert til å være offlineMode som begrenser mulighetene brukeren har i appen, som for eksempel å gjøre kall på API.



ViewModel og modularitet

Når det kom til valget om å ha én ViewModel tidligere i prosjektet, gikk det opp for oss at dette gjorde appen mindre mottakelig for videreutvikling. Beslutningen om å ha én ViewModel ble gjort mens applikasjonen var mindre omfattende og det virket overflødig med flere ViewModels. Etterhvert som funksjonalitet for lokal lagring, frakoblet modus og oppdatering av været i sanntid ble implementert, opplevde vi at det ble utfordrende å ha samme ViewModel hele veien. I retrospekt hadde det heller vært optimalt med flere ViewModels med mindre og mer spesifikke ansvarsområder. Dette hindret oss i å ivareta høy kohesjon og lav kobling, slik vi ønsket.

Teamarbeid og møtevirksomhet

For denne iterasjonen var arbeidsoppgavene som gjenstod tydelig definerte og perioden besto i store deler av individuelt arbeid. På grunn av den reduserte kapasiteten i teamet som følge av andre akademiske forpliktelser, tilpasset vi møtevirksomheten vår til å fasilitere for dette. Vi hadde kun møte på mandag og onsdag. Ettersom oppgavene som gjensto ikke krevde mye diskusjon, ble gjøremål plukket opp etter hvor mye kapasitet enkelpersonen hadde.

5.4 Oppsummering



Hva har jeg lært i dette prosjektet? Jo, det skal jeg fortelle deg!

- Test regelmessig og ikke utsett problemer for lenge!
- Det er lov å ha det gøy på møter, men pass på. Det kan fort bli for mye av det gode!
- Sørg for å planlegge, men ikke vær for tilknyttet planen. Det er utfordrende å unngå overhead kostnader og ting kan endre seg kjapt! Det viktigste er ikke planen, men planleggingen i seg selv.

6.0 Bibliografi

Forsvaret. (2009). *Veiledning i Vintertjeneste – Personlig bekledning.*

<https://s3-eu-west-1.amazonaws.com/turistforeningen/files/f4e16339721cf582ab807725485682818660bd8b.pdf>

Google. (2023a). *Composable.*

<https://developer.android.com/reference/kotlin/androidx/compose/runtime/Composable>

Google. (2023b). *Compose | Material 3.*

<https://developer.android.com/jetpack/androidx/releases/compose-material3>

Google. (2023c). *Krav til API-målnivået for Google Play-apper.*

<https://support.google.com/googleplay/android-developer/answer/11926878?hl=no>

Google. (2023d). *Material Design.* <https://m3.material.io/>

Google. (2023e). *Save data in a local database using Room.*

<https://developer.android.com/training/data-storage/room>

ISO. (2022). *ISO/IEC 25010.*

<https://iso25000.com/index.php/en/iso-25000-standards/iso-25010>

JetBrains. (2023). *Get started with Kotlin.*

<https://kotlinlang.org/docs/getting-started.html>

Meteorologisk institutt. (2023). *Case 4 åpent case—I lufta.*

Microsoft. (2022). *Model-View-ViewModel (MVVM)*.

<https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm>

NDLA. (2023). *Skrive brukerdokumentasjon*.

<https://ndla.no/nb/subject:l:1352b19e-e706-4480-a728-c6b0a57ba8ae/top>

<https://ic:l:e48b7498-b2d3-4ad0-b797-bd153728ad47/resource:l:109223>

Norrøna. (2023). *Bak sommene: Performance rangering*.

<https://www.norrona.com/nb-NO/hjelp-meg/performance-rangering/>

NTNU. (2019). *Sekvensdiagrammer—NTNU wiki*.

<https://www.ntnu.no/wiki/display/tdt4100/Sekvensdiagrammer>

OpenAI. (2023). *DALL-E 2*. <https://openai.com/product/dall-e-2>

Pfizer. (2023). *Research explains why most women feel cold more intensely than men*.

https://www.pfizer.com/news/articles/cold_wars_why_women_feel_the_chill_more#:~:text=But%20a%20University%20of%20Utah,90%20degrees%20F%20for%20men

ProductPlan. (2023, mars 17). *Scrumban*. ProductPlan.

<https://www.productplan.com/glossary/scrumban/>

Runde. (2017). *Objektorientert design*.

<https://www.uio.no/studier/emner/matnat/ifi/INF1010/v17/lysark/oodesign1-2017.pdf>

Sommerville, I. (2021). *Engineering Software Products An introduction to Modern*

Software Engineering.

Speakman, J. (2018). *Obesity and thermoregulation.*

<https://pubmed.ncbi.nlm.nih.gov/30454605/>

UiO. (2023). *IN2000 – Emneoversikt.*

<https://www.uio.no/studier/emner/matnat/ifi/IN2000/>

U.S. General Services Administration. (2023). *Running a Usability Test.*

<https://www.usability.gov/how-to-and-tools/methods/running-usability-tests.html>

Visual Paradigm. (2022). *What is Use Case Diagram?*

<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>

W3C. (2023). *Web Content Accessibility Guidelines.*

<https://www.w3.org/TR/WCAG21/>

WebAIM. (2023). *Contrast Checker.*

<https://webaim.org/resources/contrastchecker/>

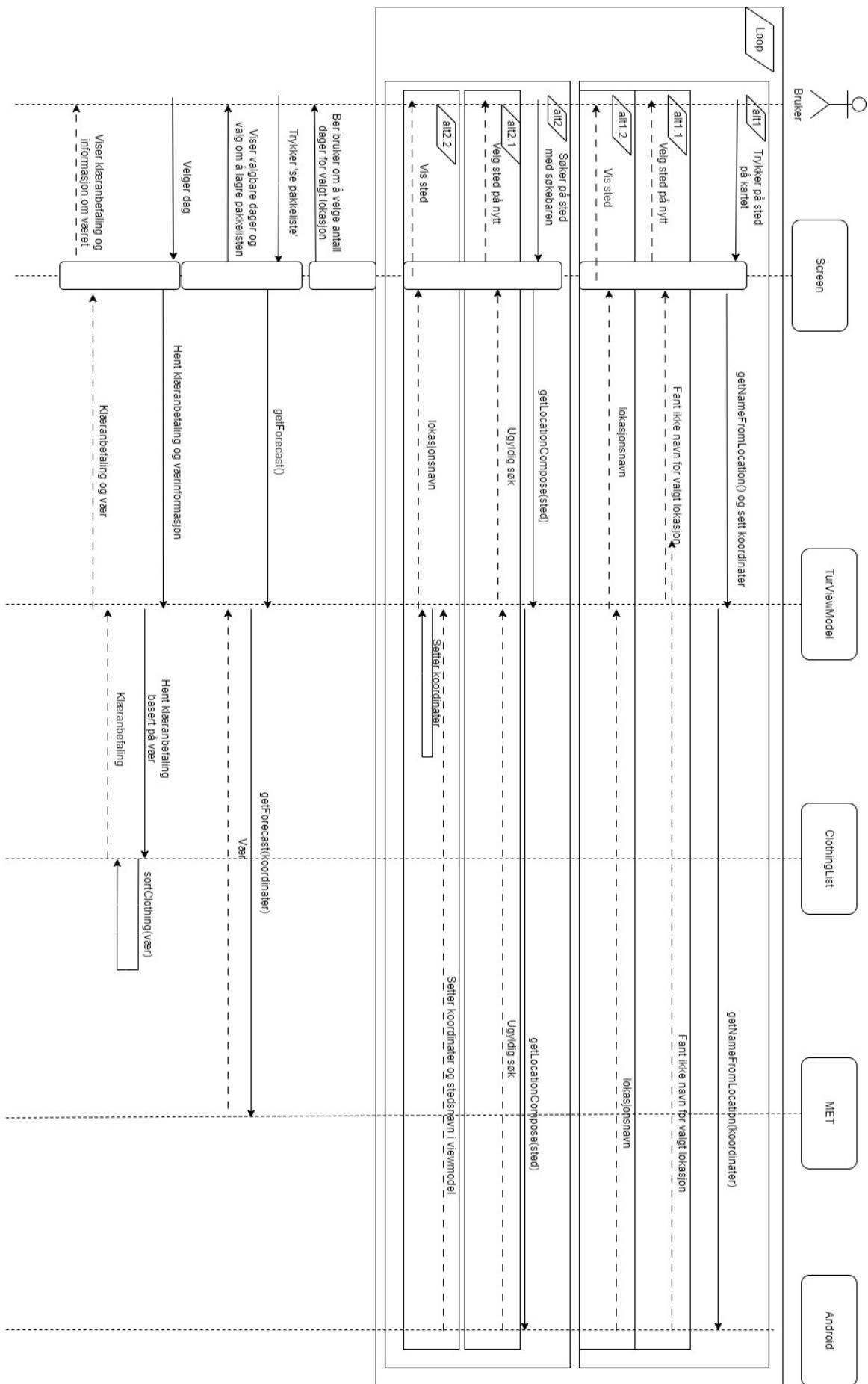
7.0 Vedlegg

7.1 Sekvensdiagram

Hovedflyt:

1. Bruker trykker på sted i kartet
 - a. koordinatene lagres i viewmodel
 - b. Viewmodellen henter navnet for den valgte lokasjonen gjennom Android
 - c. Om det ikke er noe navn for lokasjonen, må bruker velge lokasjon på nytt (alt1.1). Dersom et lokasjonsnavn finnes, går man ut av loopen.
2. Bruker søker på sted med søkebaren
 - a. Viewmodellen tar imot dette og beregner koordinatene for stedet gjennom Android
 - b. Om den ikke finner et passende sted, må bruker velge sted på nytt (alt2.1).

Dersom den finner koordinater for stedet, lagres disse i viewmodel, lokasjonsnavnet går til skjermen og man går ut av loopen.
3. Lokasjonsnavnet vises på skjermen og bruker bes velge antall dager for valgt lokasjon
4. Bruker velger antall dager og trykker 'se pakkeliste'
5. Skjermen ber viewmodel om å hente været
6. Viewmodel kaller getForecast() med koordinatene som utgangspunkt
7. Været lagres dermed i viewmodel
8. Bruker kan velge hvilken dag hen vil se pakkeliste for
9. Skjermen ber om klæransbefaling og værinformasjon fra viewmodel.
10. Viewmodel henter klæransbefaling fra ClothingList
11. Informasjonen sendes tilbake til skjermen



7.2 Brukertestguide MVP

Det har blitt gjennomført mange uformelle tester i løpet av prosjektet, men dette er et eksempel på en formell brukertest av MVP som vi utførte i påskeferien.

Ved å studere interaksjonen mellom en bruker og vår mvp, forventer vi å finne ut av om den er enkel og praktisk å bruke for målgruppen til appen og om brukeren finner den informasjonen de er ute etter. Målgruppen er alle som kunne tenke seg å forberede seg til å dra på tur utendørs. Deltakeren kommer til å være anonym, fordi det eneste vi lurer på om deltakeren er hvor mye erfaring deltakeren har med å dra på tur utendørs. Vi tenker å be deltakeren om å utføre følgende oppgaver:

- Sjekk været for stedet du befinner deg på.
- Velg stedet der du er nå. Hvis du er i Oslo, velg et annet sted i Oslo.
- Lag en pakkeliste for stedet som er valgt
- Studer pakkelisten og kom med din subjektive mening om pakkelisten passer for stedet du skal til basert på værmelding.
- Gå til hjemskjermen, og se om du klarer å finne tilbake til pakkelisten du nettopp laget.

Vi skal notere ned relevante erfaringer og observasjoner under brukertesten, og evt tilbakemeldinger brukeren kommer med.

Tilbakemelding fra brukere:

Deltaker 1: litt over gjennomsnittlig turgåer i 20- årene.

Ønsket å kunne lagre tidligere pakkelister for å se tilbake på dem om de fungerte bra.

Ønsket knapp på plaggene som ga nest beste jakka hvis man ikke har den anbefalte jakka.

Ønsket anbefaling om solkrem, solbriller eller caps, og sko.

Logisk navigasjon.

Deltaker 2: litt over gjennomsnittlig turgåer i 20- årene.

Ønsker mulighet for å velge flere steder i løpet av dagene pakkelisten skal gjelde for. fra A til B til C foreks.

Det må være noe som møter brukeren med én gang slik at appen ikke er forvirrende for brukeren. At appen forteller deg hva du skal gjøre i den. Tenk symboler, fortellende tekst, ting som er gjenkjennbart. Splash screen burde komme med én gang når den laster.

En meny med flere ting som Map, Logg med for eksempel navn på turen med liten tekst som gir litt mer info om hvilken tur det er foreks. dato – antall dager, steder turen går gjennom. m.o.h forskjell. avstand på turen. Speiderbok tutips/fjellvettregler.

Deltaker 3: Litt over gjennomsnittlig turgåer i 20- årene.

Appen er designet slik at det er enkelt å navigere seg rundt. Værmeldinga kan være litt bredere ved å vise været over en lengre tidsperiode. Kan også vite litt mer informasjon om valgt destinasjon som kan være relevant for planlegging og pakking. For eksempel type natur osv, og hva slags utstyr og klær som kan være relevant å pakke.

Sosial plattform. Se hvilke turer andre har gått og anbefaler. følge folk osv, like når andre legger ut turen sin.

Deltaker 4: gjennomsnittlig aktiv turgåer i 20-årene:

Bruker syntes ikke navigasjonen er helt intuitiv fra startskjermen. Brukeren får assistanse i hvor det skal trykkes på skjermen for å komme til kart-skjermen. Det blir valgt sted for tur og bruker får anbefalt en pakkeliste

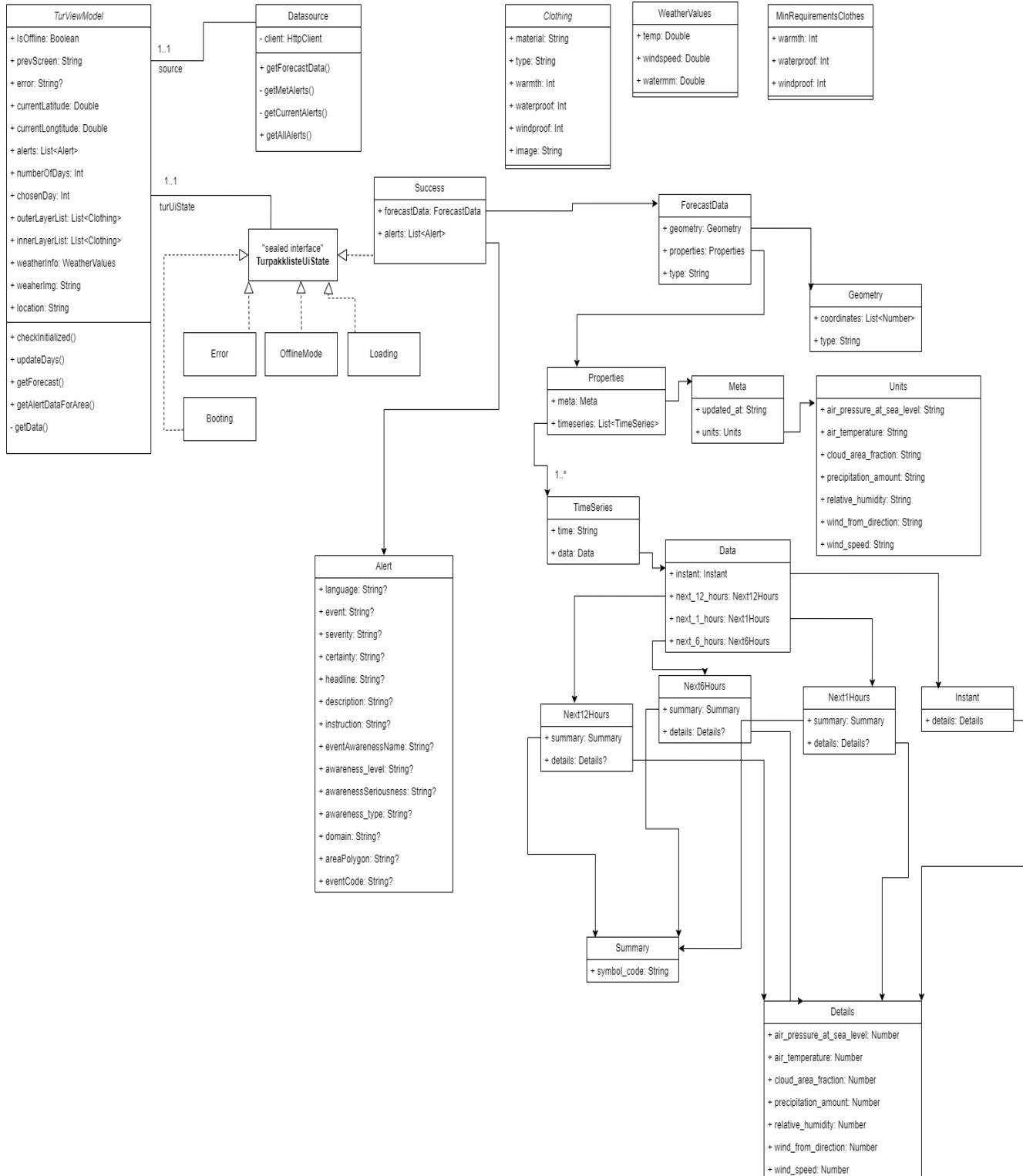
med klær. Brukeren gir positiv tilbakemelding på anbefalte klær for været basert på tidligere erfaringer. Brukeren ønsker en beskrivelse av poengsystemet for klærne. Bruker også en funksjonalitet for å lagre turer og inkludere bilder

Deltaker 5: Erfaren turgåer i 40-årene: Svært positiv for ideen før selve testingen. Brukeren liker designet og valg av farge, men gir en sterk høstfølelse. Dette ga bruker forslag om farge basert på årstid. Velger område som bruker befinner seg i og får anbefalt klær for regnværsdag som stemmer. Bruker er fornøyd med anbefalte klær og synes det er passende for været. Ønsker derimot bedre informasjon om klesplaggets stoff og ikke navn fra norrøna.

Deltaker 6: Uerfaren turgåer i 40-årene: Bruker gir god tilbakemelding om design for appen og finner fort frem til kart-skjermen for å velge seg et område. Velger både Oslo og Dovre. Bruker blir skeptisk når knappen for å motta pakkelista vises på skjermen. Brev-ikonet fikk brukeren til å lure på om pakkelista kom på mail, noe som ikke var aktuelt. Bruker ønsker mer direkte visuell tilbakemelding for anbefalte klær og ikke måtte trykke knapp for hvert plagg.

Deltaker 7: Uerfaren turgåer tenåring: Bruker synes det bør være mer tydelig at man skal trykke på map-knappen for å gå videre i appen. Brukeren kommer så med et forslag om å gi en tekstlig beskrivelse for å vise dette. Velger området vi befinner oss på og får anbefalt klær bruker tror er passende. Ønske om å legge til venner for å ha som en sosial greie at man er på tur.

7.3 Klassendiagramm

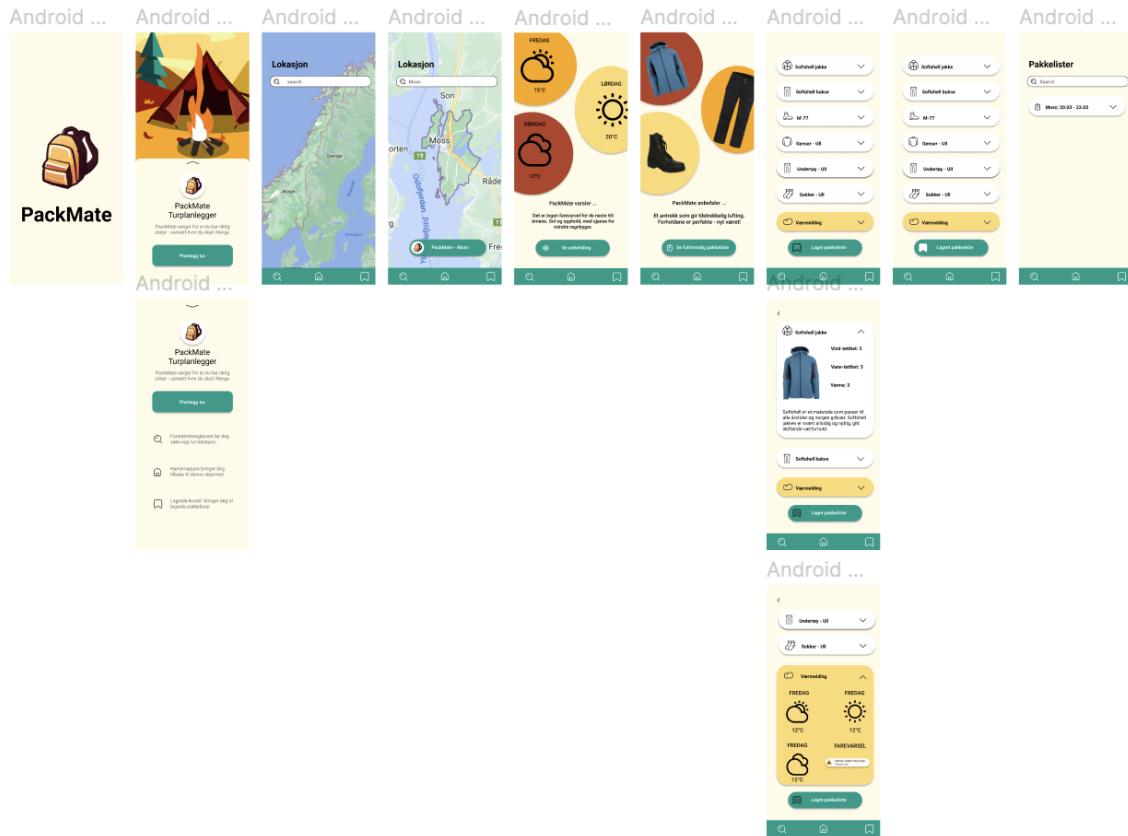


7.4 Testing av WCAG krav med WebAIM

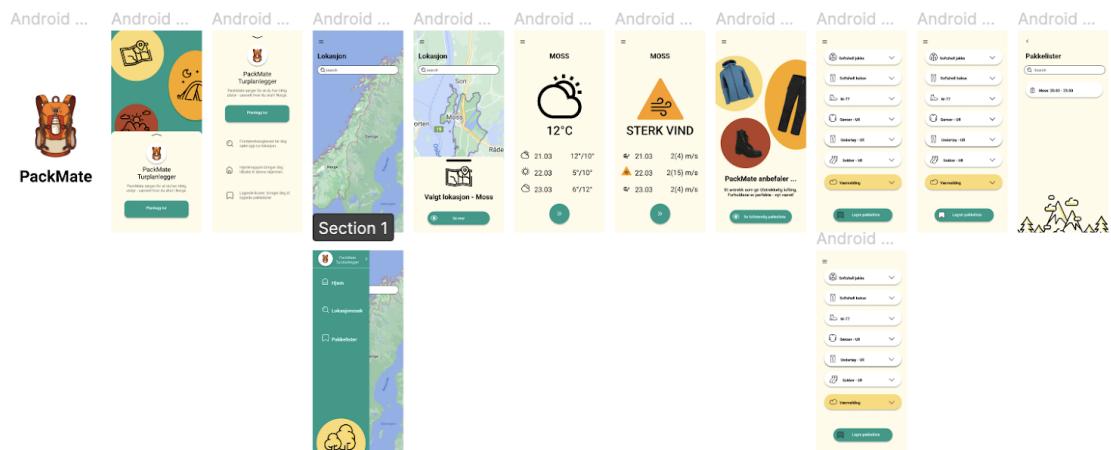
Ifølge WebAIM , så var ikke kontrastforholdet stort nok mellom noen av de fargene vi brukte mest i appen. Vi brukte deres “ContrastChecker” for å sikre at fargene hadde et tilstrekkelig stort kontrastforhold. Vi fant blant annet ut av at vår grønnfarge måtte bli mørkere for å tilfredsstille WCAG kravene.

En annen ting vi måtte endre på for å tilfredsstille kravet om stor nok tekst, var å endre alle tekstelementene i appen til å være minimum skriftstørrelse 14.sp og bold fontWeight, eller skriftstørrelse 18.sp uten bold fontWeight.

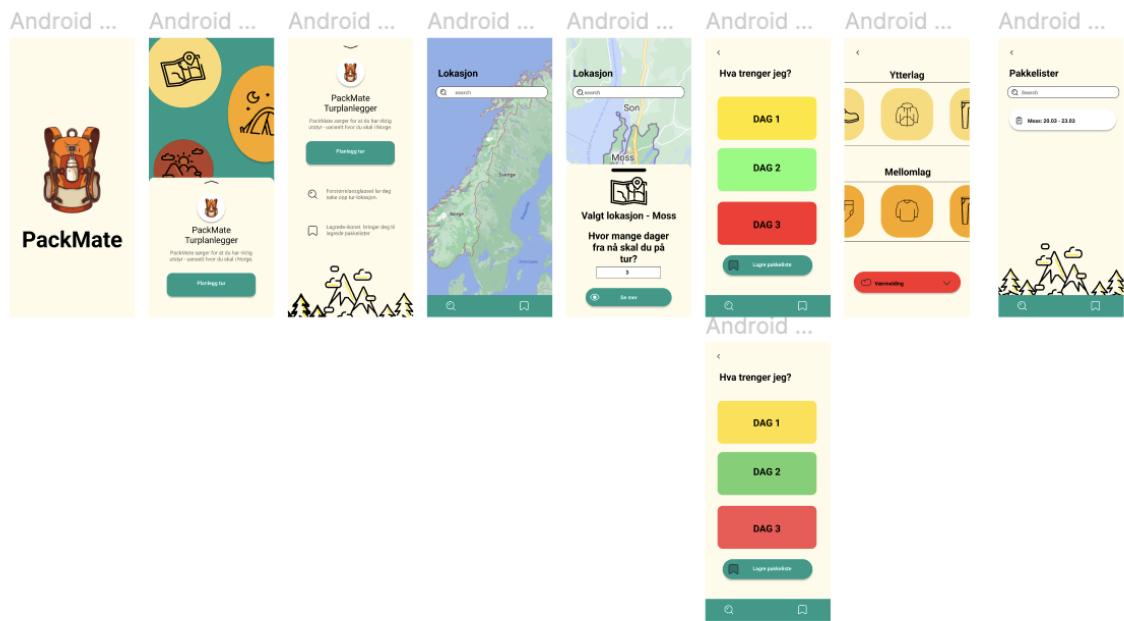
7.5 Prototype iterasjon 1



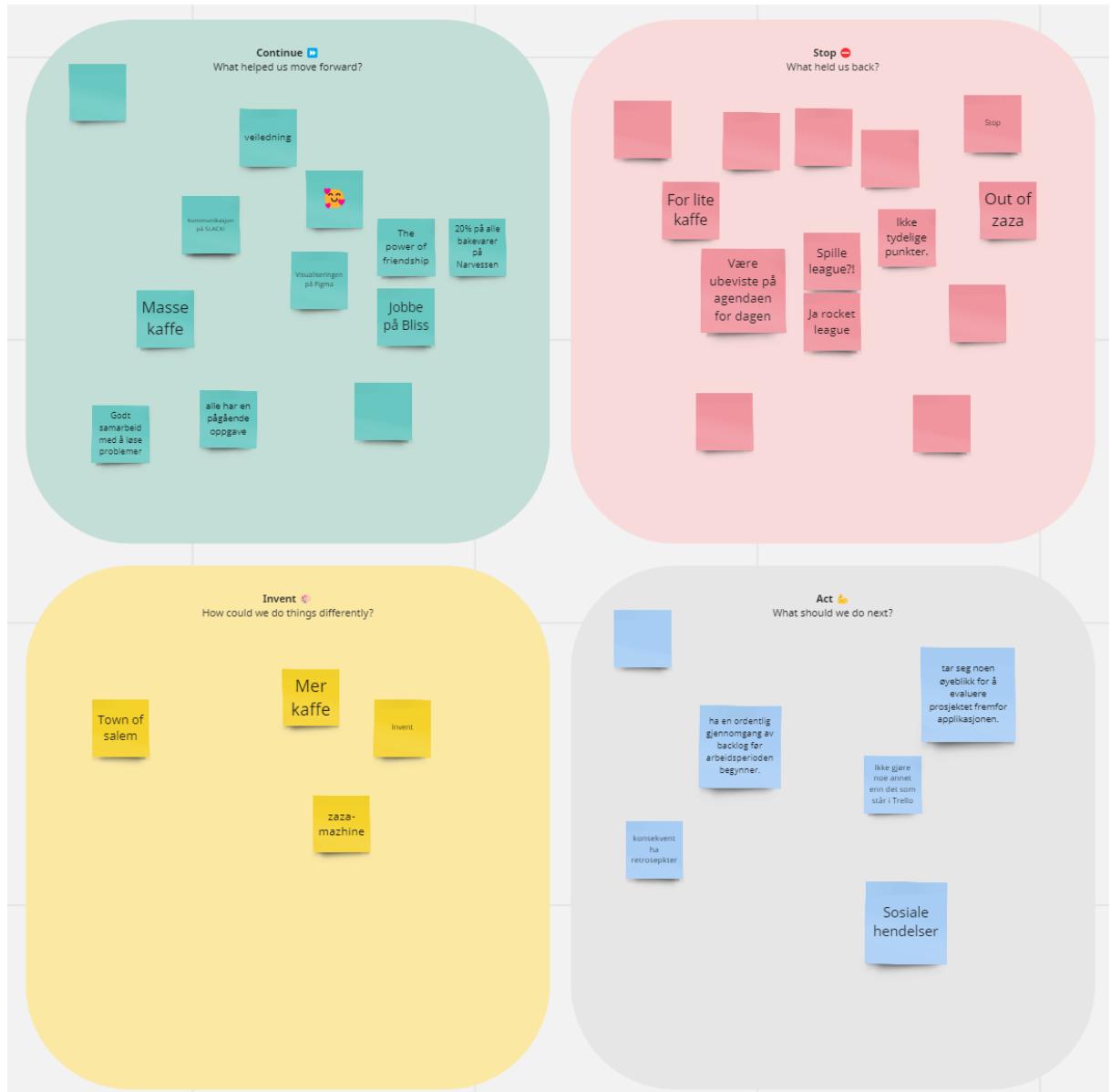
7.6 Prototype iterasjon 2



7.7 Prototype iterasjon 3



7.8 Siste retrospektiv – Iterasjon 1



7.9 Mislykkede avatarer



7.10 Teamavtale

Navn: BonkBoiz. **Dato:** 17.03.2023.

<p>HENSIKT</p> <p>Hvorfor eksisterer dette teamet? Hva ønsker vi å oppnå?</p> <p>Hvorfor er prosjektarbeid i team viktig for læringsprosessen?</p>	<p>Vi ønsker å lage en nyttig og kul applikasjon som utfordrer oss kreativt og teknisk. Videre ønsker vi å få erfaring med ulike verktøy og arbeidsmetodikker som er arbeidslivsrelevante.</p> <p>Prosjektarbeid er viktig i læringsprosessen fordi: vi lærer oss å samarbeide, det tilrettelegger for meningsutveksling og det gir praktisk erfaring i bruken av nyttige arbeidsverktøy og metoder.</p>
<p>MÅL</p> <p>Hvilke konkrete resultater ønsker vi å oppnå som team? Hvilket kvalitetsnivå?</p> <p>Hvilke kunnskap og ferdigheter ønsker vi å utvikle?</p> <p>Hvilken karakter?</p>	<p>Resultater: Som team ønsker vi å skape en funksjonell applikasjon, med en lettfattelig utforming og et konkret bruksområde.</p> <p>Kunnskap og ferdigheter: Vi ønsker å bli bedre på samarbeid, AndroidStudio, Kotlin, Git, Trello, bruk av API, kommunikasjon over Slack, kodeskikk og bruk av smidige utviklingsmetoder.</p> <p>Karakter: Vi skal selvfølgelig få A!</p>

ROLLER	Rolleinndeling:
Hvordan fordeler vi arbeidet? Hvilke oppgaver har den enkelte?	Git-mester (Støtte-ansvarlig for Git): Ahmed.
Hvem kan ta beslutninger? Hvordan skal vi ta beslutninger?	Señor-UI (SUI) (UI-ansvarlig): Alex Rapportguru (Rapportansvarlig): Aron
	Brukertester-mester (Brukertester): Trym Algoritme-lord (Backend): Haakon
	Figma-sigma (Visuell profil): Sander
	Beslutninger: Vi har desentralisert beslutningstakning med tanke på hurtighet for mindre beslutninger, eksempelvis ordlyd når man skriver til rapporten. Valg av API eller større strukturmessige betraktninger som fundamentalt påvirker applikasjonen diskuteres i gruppen. Dette blir en avveining mellom effektivitet og oversikt på gruppenivå, som vi vil forsøke å balansere gjennom hyppige gruppemøter.

FORVENTNINGER OG POLICIES	
<p>Hva forventer vi av hverandre når det gjelder oppmøte, deltagelse, kommunikasjon, innsats osv.?</p>	<p>Oppmøte: Vi forventer at folk møter til avtalte møtetider. Unntak vil kunne gjøres dersom man gir tydelig beskjed i forkant. Dersom man er forsinket med mer enn 10 minutter til 3 møter, uten at man gir tydelig beskjed i forkant, må synderen kjøpe pizza til gruppen.</p>
<p>Hvor ofte og når skal vi møtes?</p> <p>Hvilke felles forpliktelser ønsker vi å få på plass for å sikre at teamet fungerer godt? (f.eks. hvordan deler vi informasjon, møter, hvordan ta avgjørelser osv.)</p>	<p>For individer i gruppen tilsvarer altså 3 strike, pizza-kjøp.</p>
<p>Hvilke regler kan vi være enig om slik at vi når målene og forventningene?</p>	<p>Deltakelse og innsats: folk må gjøre oppgaver innen avtalte frister. Dersom man ikke har gjort en tilstrekkelig innsats så vil det tilsvare en "strike". Dersom man ikke oppgir noen grunn og ikke møter opp til møte så vil det tilsvare automatisk pizza-plikt.</p>
	<p>Kommunikasjon: Det skal være rom for å stille spørsmål og søker hjelp. Vi har kjernetid fra 8-16 i ukedagene, der vi hovedsakelig kommuniserer på Slack. Folk burde være oppmerksomme på chatten og følge med aktivt i kjernetiden.</p>

<p>KONSEKVENSER</p> <p>Hva gjør vi dersom en eller flere teammedlemmer bryter med mål, forventninger og regler?</p> <p>Bør vi ha et sanksjonssystem? Hvordan bør det fungere?</p> <p>Kan det være en ide å ha retrospektmøte?</p>	<p>Spesifisering av sanksjon: Vi bruker pizza som hoved-sanksjon i gruppa vår. Ved 3 "strikes" vil det medføre pizza-plikt for den skyldige.</p> <p>Retrospekt: Vi planlegger å ha korte retrospektmøter på fredagene, der vi også legger slagplan for neste uke.</p>
<p>STØTTE</p> <p>Hva slags støtte (f.eks. veiledning, ressurser, informasjon osv.) tror vi at vi trenger fra andre for å lykkes med å nå målene våre?</p>	<p>Forventing til hverandre: Vi forventer åpen kommunikasjon blant teammedlemmer og at alle er behjelpelege. Vi ønsker å ha en aktiv dialog med veilederne.</p> <p>Vi henvender oss til avtalte kommunikasjonsverktøy som Slack for støtte, samt de avtalte møte-tidspunktene våre.</p>

Med denne teamavtalen erklærer vi oss enig følgene mål og forventninger, policies, prosedyrer og konsekvenser.

Teammedlem navn: Trym Audun Utne Eriksen

Teammedlem navn: Sander Halvorsen

Teammedlem navn: Aron Berger Johannessen

Teammedlem navn: Alexander Døviken Løvold

Teammedlem navn: Ahmed Abdulahi Ahmed

Teammedlem navn: Haakon Korslund

7.11 Kvittering for besvart undersøkelse fra Met

IN2000 - Tilbakemelding API

IN2000 – Software Engineering med prosjektarbeid Vår 2023

Project group details

This assignment uses hard deadlines. You will not be able to write comments or upload files after the deadline has expired.

DOWNLOAD:

[first deadline](#)

Deadline: Friday May 26, 2023, 23:59

Attempt 1

waiting for deliveries

Haakon Korslund (haakonpk) [Edit](#)

Wednesday May 24, 2023, 19:08

(student)

[responsskjema2.xlsx](#) 