IN2000 – Software Engineering med prosjektarbeid

Case 1. Rakettoppskytning

Prosjektrapport Team 48 16.05.2024

ROCKET LAUNCHER

Peter Larsen Olaisen (*peterlol),* Johan Rondestvedt (*joharond),* The Dat Le (*thedl),* Gard-Henrik Haftor Bjerkelund Gimse (*ghgimse),* Wiktor Blaszkowski (*wiktorbl),* Oskar Wiig Melvold (*oskarwm)*

Veileder:
Fredrik Kin Shing Ma

INNHOLDSFORTEGNELSE

1.1 INTRODUKSJON

I dette prosjektet har team48 utviklet en applikasjon dedikert til å assistere med planlegging av rakettoppskytninger ved Universitetet i Oslo. Formålet med appen er å gi brukerne oppdatert og relevant værinformasjon som er kritisk for å gjennomføre sikre og vellykkede oppskytninger. Appen tar for seg flere værfaktorer som sikt, nedbør, og vindhastighet ved forskjellige høyder, som alle kan påvirke oppskytningsvinduet betydelig.

Denne rapporten dokumenterer utviklingsprosessen av appen fra konseptualisering til ferdig produkt. Gjennom rapporten vil leseren få innsikt i hvordan teamet tilnærmet seg prosjektet, inkludert valg av teknologier, håndtering av tekniske og ikke-tekniske utfordringer, og tilpasninger gjort basert på tester og tilbakemeldinger. Detaljer om teamets samarbeid og bruk av smidige utviklingsmetoder vil også bli presentert, sammen med en evaluering av appens funksjonalitet og brukervennlighet.

1.2 PRESENTASJON AV PROSJEKTET

Vi har valgte case 1, som består av å bygge en app for planlegging av rakketoppskytning for rakettbyggegruppen Portal Space. De ønsker en tjeneste for å se værdata og dens utvikling, slik at de kan vite om det er trygt å skyte opp en rakett. Dette vil gi rakettbyggegruppen et estimat for om de ulike faktorene stemmer.

De funksjonelle kravene for om det er forsvarlig å skyte opp i løpet av de neste timene er: sikt, nedbør, vindhastighet ved forskjellige høyder og juridiske områder. Sikten er viktig fordi det er ønskelig å se raketten under oppskytning. Dette hjelper til med sikkerhet og at det er mulig å ta opptak av oppskytningen. Det er viktig å holde styr på nedbør, fordi det jobbes med mye elektronikk og kryogenisk væske. Fukt kan bli gjort om til is, og vil skade eventuelle ventiler andre deler av raketten. Under oppskytning vil raketten være mindre stabil på grunn av en lavere fart. Det er derfor viktig å vite hvordan tilstanden til vinden er på bakkenivået. Ved fallskjerms fasen kan vinden i de ulike luftlaga føre raketten vekk, og landingsområdet til raketten vil bli større, dette ønskes å være minimalt grunnet sikkerhetsårsaker. I tillegg kan

sterke vindkast i ulike retninger i de ulike laga føre til rotasjon. Til slutt må oppskytningen skje på en godkjent plass, og derfor ha kontroll på hvilke områder det er.

1.3 BESKRIVELSE AV LØSNINGEN

Vår løsning av caset er en enkelt å bruke app som gir et estimert svar på om det er forsvarlig å skyte opp en rakett for de neste tre timene. De funksjonelle kravene blir vist om de er oppfylt på hjemskjermen til appen. Der vises all værdata som trengs i tillegg til om det er innafor de juridiske områdene. For å finne ut om nedbør, vind og sikt er gunstige, har appen normalverdier for hver data. Disse verdiene kan brukeren selv endre hvis det er ønskelig. For å finne ut om oppskytningsplassen er i et juridisk område har vi valgt og ta utgangspunkt i om område befinner seg i et "luftsportsrom/air sport boxes". Dette er fordi for å finne ut om området er en godkjent plass, må man ha en individuell avtale med grunneier. I tillegg trenger luftfartstilsynet å kontaktes lang tid i forveien og de har klare retningslinjer. Så vi har simplifisert løsningen.

1.4 INTRODUKSJON TIL TEAMET

Vi i team 48 består av seks elever ved Universitetet i Oslo, som alle tar bachelorprogrammet Informatikk: Programmering og systemarkitektur som sitt studieprogram. I tillegg har alle sammen på gruppa IN2140 – Introduksjon til operativsystemer og datakommunikasjon som studieemne i sitt 4. semester. I starten av prosjektet hadde hver enkelt person på gruppa, en person de kjente fra før av. Altså en gruppe av tre par som kjente hverandre.

BRUKERDOKUMENTASJON

2.1 FUNKSJONALITET

Denne applikasjonen er spesialutviklet for å assistere Portal Space, en studentorganisasjon ved Universitetet i Oslo, i deres arbeid med planlegging av rakettoppskytninger. Appen tilbyr løpende oppdateringer om værforhold og evaluerer om disse forholdene oppfyller de nødvendige sikkerhetskravene for en rakettoppskytning.

Appens hovedfunksjon er å overvåke værdata. Den samler inn informasjon om vindforhold både på bakkenivå og i ulike høyder basert på isobarisk trykk, og gir også data om fuktighet, nedbør og sikt. For å sikre at alle oppskytningsaktiviteter foregår innenfor trygge rammer, har appen en funksjon hvor brukerne kan justere sikkerhetsparametrene etter behov. Dette gjøres gjennom en dedikert innstillingside som er enkel å bruke.

En annen viktig funksjon i appen er valg av oppskytningssted. Appen inneholder kartfunksjonalitet som gjør det mulig for brukerne å velge og vurdere ulike oppskytingssteder basert på de gjeldende værforholdene og områdets egenskaper. Videre gir appen brukeren oversikt over juridiske bestemmelser som er relevante for de valgte oppskytningsområdene. Dette vil hjelpe brukerne å forstå og overholde alle lokale regler og forskrifter som gjelder for rakettoppskytninger.

Figur 1: viser oppstartsskjerm

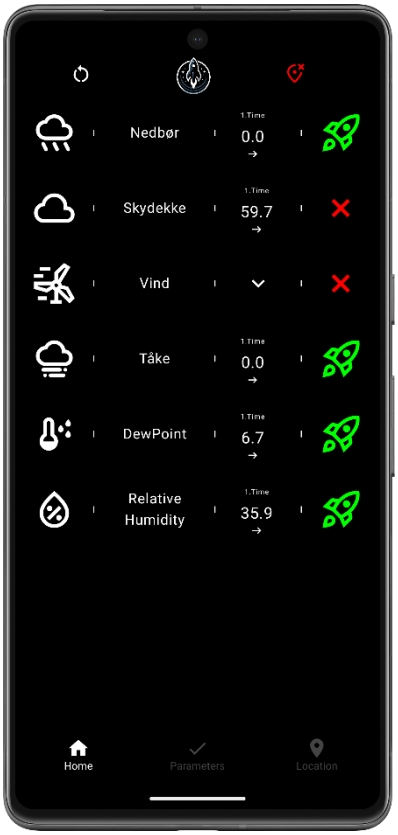
OPPSTARTSSKJERM

Under oppstart vises denne splashskjermen med logoen til appen for å gi en visuell tilstedeværelse mens appen laster nødvendige data i bakgrunnen. Splashskjermen vises i ca. 3 sekunder før du automatisk tas videre til hovedmenyen. Dette sikrer at alt er klart for en sømløs brukeropplevelse. (Figur 1)

HOVEDSKJERM I

Når appen starter, kommer du direkte til hovedskjermen. Her gir vi deg en rask oversikt over viktige værvariabler som nedbør, skydekke, tåke, luftfuktighet og vind. Disse variablene er vist for forskjellige deler av atmosfæren.

For å gjøre det enklere å tolke dataene, bruker vi fargekoder. Røde indikerer at variablene er utenfor de valgte parameterverdiene, mens grønne viser at de er innenfor. Dette gjør det mulig for deg å raskt se og vurdere værforholdene. (figur 2)



Figur 2: viser hovedskjermen

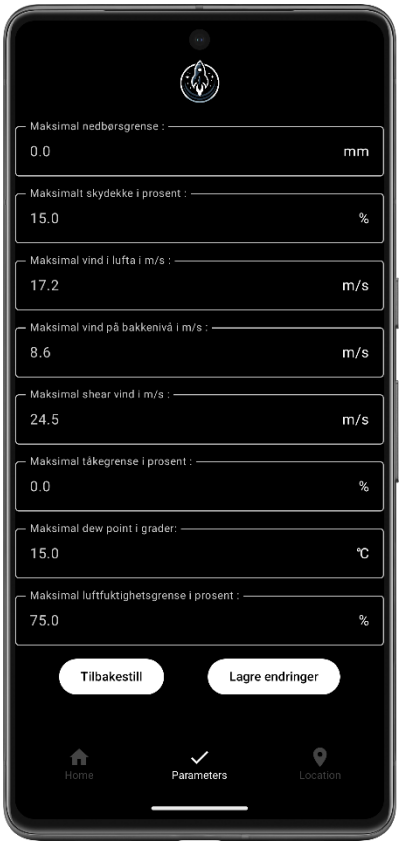
Figur 3: viser hovedskjermen når vindraden er ekspandert

HOVEDSKJERM II

Når du trykker på 'Vind' i hovedskjermen, vil en utvidet liste åpne seg med detaljert informasjon om vindforholdene på forskjellige høydemeter. Listen viser både vindstyrke om vind shear mellom de ulike høydenivåene. Vi bruker fargekoder for å indikere sikkerhetsnivåene: røde farger betyr at vindforholdene er utenfor trygge parameterverdier, mens grønne indikerer sikre forhold. Dette detaljnivået gir deg en presis forståelse av de dynamiske vindforholdene. (figur 3)

PARAMETERSKJERM

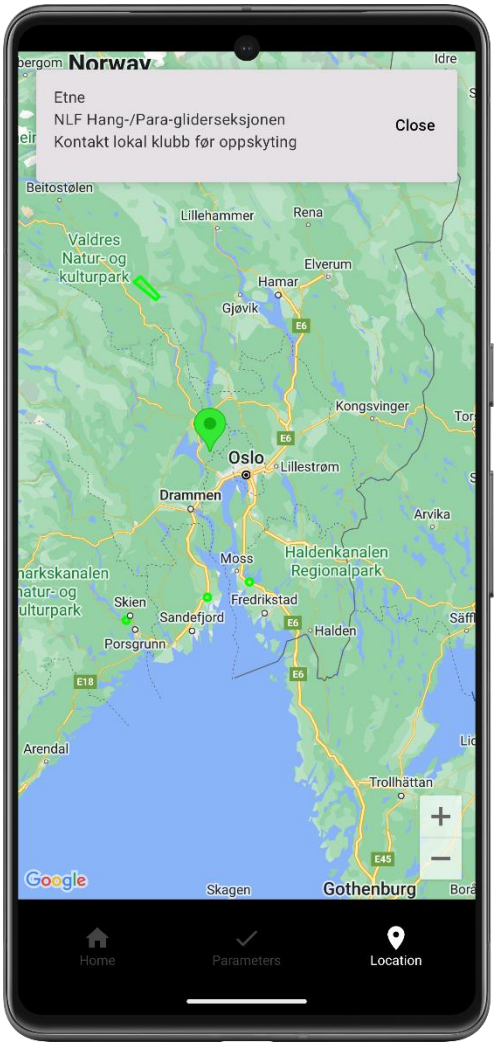
Fra hovedskjermen kan du enkelt navigere til parameterskjermen ved å bruke navigasjonsbaren. På denne skjermen har du muligheten til å justere verdier for forskjellige parametere etter dine spesifikke behov og hvilken type prosjekt (f.eks., rakett) du arbeider med. Dette lar deg tilpasse appen nøyaktig til de kravene du har. (figur 4)



Figur 4 – Viser parameterskjerm

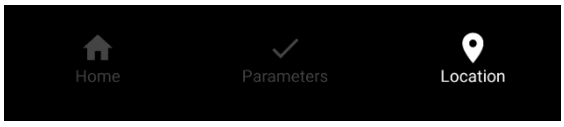
KARTSKJERM

På denne skjermen kan du velge lokasjoner direkte på et interaktivt Google-kart. Dette er essensielt for å sikre at oppskytningsvilkårene er innenfor nødvendige sikkerhetsmarginer. Du kan også få detaljert informasjon om juridiske bestemmelser som gjelder for forskjellige oppskytningsområder. Bruk kartet til å planlegge og validere oppskytingssteder etter både meteorologiske og juridiske kriterier. (figur 5)



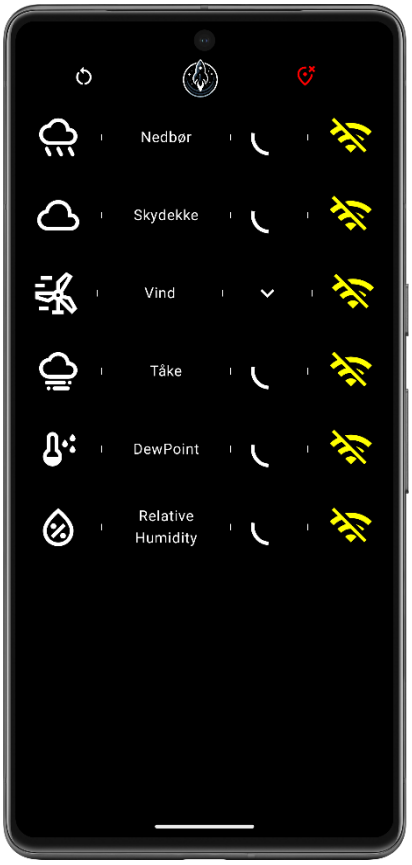
Figur 5 : Viser kartskjermen

NAVIGASJONSBAR



Figur 6 : Viser navigasjonsbar

Navigasjonsbaren gir brukerne en intuitiv måte å veksle mellom ulike skjermene i appen. Dette oppsettet sikrer at du raskt og enkelt kan navigere gjennom appens funksjoner uten forstyrrelser. (Figur 6)



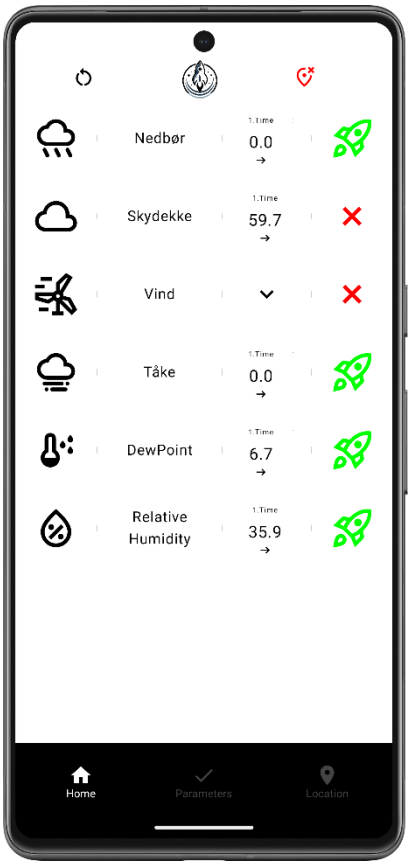
Figur 7- viser appen uten internettilkobling

INTERNETTSJEKK

Appen vil automatisk varsle deg hvis du mangler internettforbindelse eller hvis internett ikke er aktivert. Ikoner vises i brukergrensesnittet for å indikere manglende tilkobling, slik at du enkelt kan se når du trenger å sjekke dine nettverksinnstillinger. (Figur 7)

LIGHTMODE

Når lightmode er aktivert, tilpasser appen seg automatisk etter systemets tema, og gir et lyst og klart grensesnitt. Dette sikrer god lesbarhet under lyse forhold. Alle viktige værvariabler som nedbør, skydekke, og vind, er fremhevet med intuitive fargekoder—grønt for normale og rødt for unormale verdier—slik at du enkelt kan vurdere værforholdene. (figur 8)



Figur 8: *viser appen i lightmode*

TOPPAPPBAR

Figur 9: viser toppappbar

I toppbaren av appen finner du en knapp for å oppdatere dataene, slik at du alltid har tilgang til de nyeste værinformasjonene. Ved siden av oppdateringsknappen vises din valgte lokasjon, noe som gjør det enkelt for deg å se hvilket geografisk område værdataene gjelder for. Dette intuitive designet sikrer at du raskt kan få en oppdatering av værforholdene med bare et trykk. (figur 9)

2.2 MÅLGRUPPE

Hovedbrukerne av denne appen er medlemmene av Portal Space, som inkluderer studenter, lærere, og forskere innen feltene rakettvitenskap og romforskning ved Universitetet i Oslo. Disse brukerne vil benytte appen til å planlegge og gjennomføre rakettoppskytninger ved å dra nytte av tilgjengelig og offentlig værdata. Appen er designet for å hjelpe med å ta informerte beslutninger om oppskytningsvindu basert på nøyaktige værprognoser og sikkerhetsvurderinger. I tillegg kan appen være nyttig for hobbyister og amatør-rakettbyggere som ønsker tilgang til lignende data for å sikre vellykkede oppskytninger utenfor akademiske miljøer.

2.3 PLATTFORMTILGJENGELIGHET

Appen er tilgjengelig på Android-enheter og støtter operativsystemer opp til Android API-nivå 33. Dette sikrer at appen er kompatibel med de nyeste Android-versjonene og gir en jevn brukeropplevelse på de mest aktuelle enhetene. Brukere bør sjekke at deres Android-enhet opererer innenfor eller under dette API-nivået for optimal ytelse og funksjonalitet.

2.4 TILGJENGELIGHET

Selv om appen primært er utviklet for bruk av medlemmene i rakettgruppen ved Universitetet i Oslo, erkjenner vi verdien av universell utforming for å gjøre teknologien tilgjengelig for alle brukere. Vi har implementert grunnleggende tilgjengelighetsfunksjoner, slik som større

berøringsmål, tydelig kontrast, og bruk av farger kombinert med ikoner, text to speech, for å forbedre brukerinteraksjonene. Appen er generelt designet for å være logisk og intuitiv.

Vi erkjenner imidlertid at appen ikke fullt ut oppfyller alle kravene i WCAG-standardene. Spesifikke aspekter, som tekstlige beskrivelser for alle visuelle funksjoner og fullstendig tastaturnavigasjon, har ikke vært prioritert i denne utviklingsfasen. Dette skyldes delvis at appen er skreddersydd for en spesifikk brukergruppe med spesifikke tekniske ferdigheter og behov, noe som har ledet til antagelser om brukernes evne til å navigere og forstå appen uten disse tilgjengelighetstilpasningene.

Skulle appen publiseres eller utvides til en bredere brukergruppe i fremtiden, vil det være nødvendig å gjennomgå og tilpasse den i henhold til WCAGs prinsipper, retningslinjer og testkriterier for å sikre at den møter de internasjonale standardene for tilgjengelighet. Vi er forpliktet til å vurdere disse tilpasningene seriøst for å garantere at alle mulige brukere kan ha en best mulig opplevelse av appen.

PRODUKTDOKUMENTASJON

3.1 EGENSKAPER

De viktigste kvalitetsegenskapene til appen er funksjonalitet og pålitelighet. Det er viktige at appen oppnår de funksjonelle kravene til appen: Å vise fram forholdene til sikt, nedbør og vind og at oppskytningen skjer på et godkjent område. Dette er for å få kunnskap om det er trygt for oppskytning. Det er i tillegg viktig at denne dataen er pålitelig, slik at forholdene faktisk er slik det vises i appen. For å evaluere egenskapene ble det utført åpne intervjuer i form at brukertesting. Her fikk vi direkte tilbakemeldinger på hvordan funksjonaliteten til appen fungerte. Hva som trengte å bli endret, hva som var bra og hva som var forvirrende for brukeren. For å få innsikt til maksverdiene til dataen hentet fra API, og hvordan de påvirket raketten, ble det utført et åpent intervju som kommunikasjon via e-post med Haakon Offernes, Chief Electrical Officer.

3.2 API

I appen vår tar vi i bruk tre ulike API-er. Locationforecast og Isobaricgrib er to av API-ene som vi bruker, som er fra meteorologisk institutt som leverer værdata. Disse var essensielle

for de funksjonelle kravene til appen vår. Den siste API-en vi bruker er en ekstern API for å få vist fram og ta i bruk et kart. Dette gjør at det er enklere å visualisere lokasjonen til brukeren, i tillegg blir appen mer brukervennlig fordi man ikke lenger trenger å skrive inn “lat” og “lon” koordinater manuelt.

Locationforecast

Locationforecast gir globalt værvarsel for de neste ni døgnene. API-et kan bli hentet via tre ulike måter: compact, complete og classic. Classic leverer et xml-format, compact leverer en kortere versjon med bare de mest brukte parameterne som JSON, og complete som gir alle verdiene som JSON. Her får appen tilgang til viktige variabler som gir informasjon til nedbør, vind under 10 meter, sikt og temperatur (api.met/Locationforecast, n.d). Noe som var viktig når vi brukte Locationforecast API-et var å bruke en ifi proxy server. Dette ble gjort for å forhindre overbelastning av met.api sine nettsider, med tanke på at resten av IN2000 også jobber med produksjon av app og bruker Locationforecast.

Isobaricgrib

Isobaricgrib leverer GRIB-filer på GRIB-2 format, med data fra trykkflater fra 850pHa til 100pHa. Dette gir variablene for vind over 10m. En kombinasjon av disse API-ene gir oss kunnskapen til å vite om forholdene til en rakketoppskytning vil være trygt. I motsetning til Locationforcast som gir regelmessig værdata med timers intervaller, gir Isobaricgrib en gjennomsnitts verdi for værdata i de ulike trykkflatene for 3 timer.

En utfordring med Isobaricgrib er at den ikke leverer JSON-filer, i tillegg til at filene kan være tungvint å lese inn på appen på grunn av størrelsen. For å håndtere dette er en løsning å bruke en backend-server for å levere behandlet data til appen. I tillegg er Isobaricgrib API-en som er brukt, i en beta-versjon. Isobaricgrib gir også bare data fra søndre Norge, og appen får dermed ikke presis vind-data andre steder (api.met/Isobaricgrib, i.n).

Google maps:

Google Maps APIet gir oss muligheten til å integrere Google Maps' omfattende kartfunksjonalitet direkte i vår applikasjon (Google Developers, n.d). Gjennom Google Cloud-konsollen har vi anskaffet en unik API-nøkkel som gir oss tilgang til å benytte Google Maps' tjenester.

I vår applikasjon har vi implementert APIet for å vise et interaktivt kart i kartvisningen. Dette gir brukerne en intuitiv opplevelse ved å kunne navigere og zoome over kartflaten. Videre har vi utnyttet APIets funksjonalitet for markering, som lar brukerne plassere markører på kartet. Disse markørene benyttes for å indikere spesifikke posisjoner, samt gi muligheten til å bytte mellom ulike posisjoner.

PROSESSDOKUMENTASJON

4.1 UTVIKLINGSMETODER

Vi tok brukte smidige teknikker fra scrum som vår utviklingsmetode til appen. Vi hadde "sprinter" på rundt 2-3 uker, hvor vi satte våre hovedmål. To til tre uker ga oss nok tid til å arbeide med oppgaver og var med på å bidra til en mer nyttigverdig sprint review og retrospektiv, da vi hadde en del å reflektere over. I sprint-planningen satte vi oss et mål for sprinten, og fordelte ulike oppgaver for å oppnå målet (Berntzen, 2020). I sprint-periodene hadde vi standups hver gang vi hadde møte, disse antall møter varierte hver uke etter behov. Etter hver sprint hadde vi en sprint review og en retrospektiv. Her reflekterte vi over hva vi hadde fått til, om arbeidsmengden var for mye eller for lite og hva som kunne forbedres. Disse møtene utviklet oss som et team, og førte til et bedre arbeid. Etter første sprint retrospektiv implementerte vi en virtuell tavle som fungerte som et scrum table ved bruk av verktøyet trello. Dette ga oss mere oversikt over arbeidsoppgavene. Etter andre sprint ble vi enige om at oftere standup møter bidra positivt til å forstå hvordan det gikk med prosjektet og resten av gruppa. Til slutt endte vi opp med en utviklingsmetodikk som tok utgangspunkt i scrum sitt rammeverk, med sprinter, sprint-planning, tre til fem standups, sprint review, sprint retrospektiv og et virtuelt scrum board.

4.2 VERKTØY

Kommunikasjon

For kommunikasjon i gruppa har vi tatt bruk av verktøyene messenger og discord. Messenger har blitt brukt til hyppig og kort informasjon mellom gruppemedlemmer, her har det blitt

planlagt teammøter, kommentert problemer og generell kommunikasjon. Discord har blitt brukt som et samarbeidsområde hjemme, der kan man gå inn i en gruppesamtale, sende bilder, sende filer og kommenter til filene.

Smidig praksiser

Til scrum planlegging tok vi i bruk av trello. I trello fikk vi laget en digital tavle hvor vi kunne bedre organisere og spore arbeidsoppgaver, samt for å tydelig identifisere hvem som var ansvarlig for hva. Dette ville bidra til å unngå bortkastede ressurser og sikre en mer effektiv arbeidsflyt.

Modellering av diagram

For modelleringen benyttet vi mermaid-diagrammer, et intuitivt verktøy for å visualisere arkitekturen og flyten i applikasjonen. Fordelen med mermaid-diagrammer er at den genererte koden vises direkte i MODELING.md filen på GitHub, noe som gjør det enkelt å dele og oppdatere modellene.

Vi startet modelleringsprosessen med å lage et flytdiagram som illustrerte vår opprinnelige tanke om hvordan data skulle strømme fra kildene, via viewmodellen og til de ulike skjermene i appen. Dette flytdiagrammet ble justert underveis etter hvert som vi fikk bedre forståelse av arkitekturen.

Videre utviklet vi klassediagrammer, use case-diagrammer og sekvensdiagrammer for å spesifisere applikasjonens oppbygging, funksjonalitet og interaksjoner på en mer detaljert måte. Å starte med flytdiagrammet var nyttig, da det ga hele teamet en felles forståelse av den overordnede arkitekturen før vi gikk dypere inn på de ulike delene.

Gjennom denne iterative modelleringsprosessen med mermaid-diagrammer fikk vi en solid og visuell representasjon av applikasjonen vår, noe som lettet implementeringen og samarbeidet betraktelig.

Konstruksjon av prosjekt

For å lage appen brukte vi Android studios og VSCode som tekst editorer. Android studio gir oss tilgang til en emulator som gjør testing og kjøring av appen lettere på ulike Android enheter. GitHub ble brukt for å holde orden på filer og oppdateringer av appen, slik at vi jobba i samme generasjon av appen. Til rapportskriving og prosesslogg av teamaktiviteter tok vi bruk av Microsoft Office sine tjenester, dette lot oss dele dokumenter og skrive de ned samtidig.

4.3 OVERSIKT OVER PROSJEKTETS GANG

Oppstart

I vårt team har vi utforsket ulike aktiviteter for å optimalisere vår utviklingsprosess og dra nytte av erfaringer for å identifisere hva som virker best innen smidig utvikling, og hva som kanskje ikke er like verdifullt. Fra begynnelsen av prosjektet var vi enige om å følge retningslinjene og konseptene som smidig utvikling tilbyr.

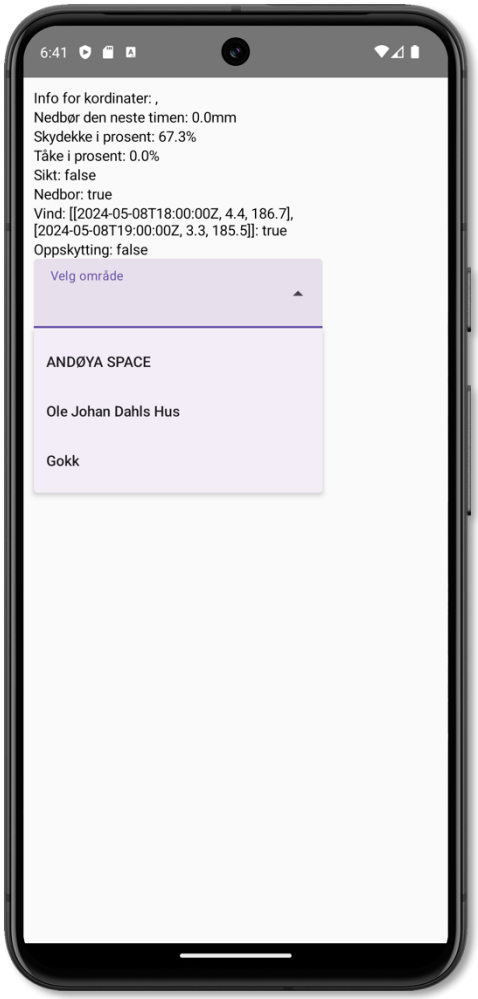
Vi startet utviklingsprosessen med et møte der vi spiste lunsj, gikk gjennom case-mulighetene våre og diskuterte prosjektet. Vi som gruppe ble enige om å ta i bruk smidig utvikling, spesifikt rammeverket Scrum for å levere et høykvalitetsprodukt, men på en effektiv måte. Vi valgte Rakettoppskyting-casen, og opprettet en type produktbacklog, med krav vi ønsker til appen vår. Tidlig i utviklingsfasen bestemte vi oss for å fokusere på MVP (Minimum Viable Product) og satte klare mål for dette. Vi utarbeidet arkitektoniske skisser og dataflytdiagrammer ved hjelp av MermaidChart, og gjennomførte grundige diskusjoner om hvilke krav vi ønsket at MVP skulle oppfylle.

Vi opprettholdt kontinuerlig kommunikasjon med Haakon Offernes, Chief Electrical Officer hos Portal Space, for å sikre at våre krav og funksjonaliteter var i tråd med Portal Spaces behov, både for MVP og for appen som helhet.

Etter å ha fastsatt kravene og funksjonaliteten for vår app, begynte vi å utforske designmulighetene. Vi gjennomførte en workshop, spesifikt en "Crazy 8" -økt, hvor vi genererte en mengde ideer for appens utseende og brukeropplevelse. Dette resulterte i flere

konsepter som vi deretter diskuterte og delte for tilbakemeldinger om design og presentasjon av data.

Utvikling og demonstrasjon av MVP



Figur 10 : Hjemskjermen ved demo av MVP

Med klare mål for vår uformelle "sprint" frem mot demonstrasjonen av MVP, gikk vi i gang med selve utviklingsarbeidet. Vi valgte å praktisere mobb-programmering i denne perioden for å styrke de sosiale relasjonene i teamet og for å gi innsikt i arbeidsprosessene til alle gruppemedlemmene. Dette ga oss også verdifull innsikt i hvordan MVP-en ville kunne utvikle seg.

Denne kombinasjonen av nøye planlegging, kontinuerlig kommunikasjon med interessenter og effektive arbeidsmetoder som mobb-programmering, har vært avgjørende for vårt teams suksess med å utvikle en vellykket app.
Hovedmålet med vår MVP var å vise frem alt relevant av data, og fremvise det vi har gjort med dataen.

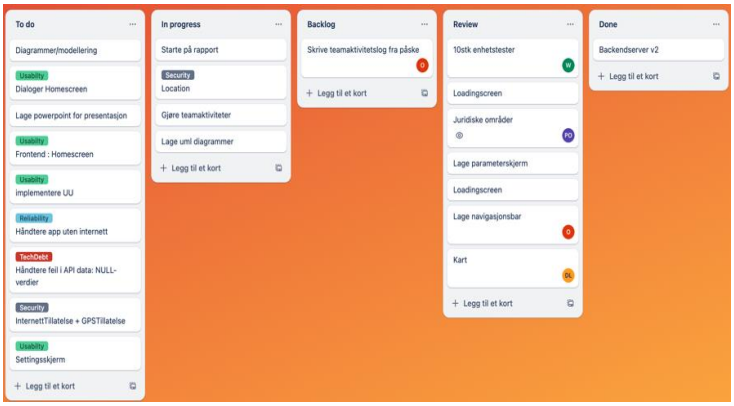
Første sprint – planlegging, gjennomføring og gjennomgang

Etter å ha demonstrert vår MVP og fullført den første uformelle "sprinten", holdt vi en retrospektiv for å evaluere arbeidsperioden. Diskusjonen brakte frem mye nyttig innsikt, inkludert elementer som fungerte bra og områder der vi så rom for forbedring.

Vi var enige om at å sette klare mål og oppnå dem var en av de positive aspektene ved vår tilnærming. På den andre siden identifiserte vi også noen områder for forbedring, spesielt når det gjaldt å fordele arbeidsoppgaver mer presist. For å adressere dette behovet foreslo vi å implementere en Trello-tavle.

Implementeringen av Trello-tavlen viste seg å være en vellykket løsning for oss gjennom resten av utviklingsprosessen. Det ga oss en tydelig oversikt over oppgaver og bidro til bedre samarbeid og kommunikasjon i teamet.

Etter å ha fullført vår første uformelle "sprint" og samlet verdifulle erfaringer, var vi klare for å gå videre med planleggingen av vår første formelle sprint. Vi innledet denne prosessen med et sprint-planleggingsmøte, hvor vi identifiserte målene for sprinten og satte opp retningslinjer for et fremtidig sprint-retrospektivt møte.



Figur 11- Trello-tavle

Med Trello-tavlen på plass, som ga oss et strukturert rammeverk for vår backlog, kunne vi enkelt organisere og fordele arbeidsoppgavene for den kommende sprinten. Hvert teammedlem fikk muligheten til å komme med forslag og ønsker, og vi tok hensyn til disse når vi fordelte oppgavene.

Ved slutten av planleggingsmøtet hadde vi etablert klare mål for sprinten, som skulle fungere som vår rettesnor og motivasjon gjennom arbeidsperioden. Disse målene skulle også være

fokusområder vi kunne evaluere og diskutere ved avslutningen av sprinten, i det planlagte retrospektivmøtet.

I denne sprinten fokuserte vi på å optimalisere effektiviteten og kvaliteten av arbeidsprosessen gjennom ulike smidige aktiviteter, ved bruk av flere ulike verktøy.. Vårt første steg var å forbedre backlogen, og forbedre kodekvalitet ved å benytte verktøy som Trello, Kotlin refactor og Qualitytagger.

Vi identifiserte et kodesnitt som oppfylte kriteriene for å være rotete, kort og enkelt å teste. For å forbedre kodekvaliteten og klarheten i koden, valgte vi å benytte verktøyet Kotlin Refactor. Målet var å rydde opp i koden, forbedre kommentarer og variabelnavn, samt skape en mer oversiktlig kodestruktur.

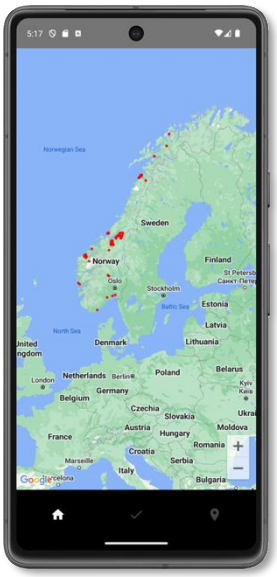
Imidlertid opplevde vi at denne prosessen med Kotlin Refactor ikke fungerte så bra som vi hadde håpet på. Tvert imot førte den til flere feil enn vi startet med, og den genererte også nye repositories uventet. Selv om verktøyet var effektivt når det gjaldt å forbedre kodeskikk og kommentarer, var det ikke en praksis vi valgte å benytte oss av igjen i løpet av utviklingsprosessen.

Denne erfaringen minnet oss om viktigheten av å være forsiktige med å benytte automatiserte verktøy for kodeopprensing, spesielt når de kan føre til uventede konsekvenser eller feil. Det understreket også behovet for manuell kontroll og testing etter slike endringer for å sikre at koden forblir stabil og funksjonell.

Vi gjennomgikk hver historie, problem og oppgave og tildelte dem relevante "tagger" før vi førte dem inn i Qualitytagger for å analysere resultatene.

Selv om Qualitytagger ikke ga de forventede resultatene, var prosessen med å tildele "tickets" en verdifull tilføyelse til vår utviklingspraksis. Det bidro til en mer strukturert og oversiktlig tilnærming som vi valgte å opprettholde.

I løpet av sprinten vår satte vi oss som mål å forbedre koden vår og introdusere ny funksjonalitet. For å oppnå dette valgte vi å bruke parprogrammering, der vi jobbet i par og byttet på å være "sjåfør" og "navigator" regelmessig. Denne tilnærmingen viste seg å være svært vellykket både for å skape ny kode og for å styrke teamets relasjoner.



Figur 12 - Videreutvikling av Kart-skjermen ved parprogrammering

Imidlertid oppdaget vi at når vi utførte enklere oppgaver, som å rydde opp i eksisterende kode, kunne det være mer effektivt å jobbe individuelt. Dette skyldtes både ressursbruk og enklere oppgavehåndtering når man jobbet alene.

Til tross for dette var opplevelsen med parprogrammering svært lærerik og effektiv når det gjaldt å utvikle ny kode og løse komplekse problemer.

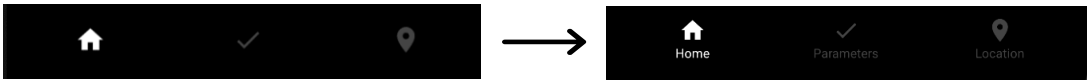
Etter omtrent fire uker med sprintaktivitet, nærmet vi oss slutten av sprinten. På dette punktet holdt vi en uformell sprint review-møte med veileder, der vi gjennomgikk alle aspekter av sprinten. Vi tok oss tid til å vurdere hva som gikk bra og hva som kunne vært bedre. Under denne gjennomgangen ble det tydelig at vi kunne ha vært enda mer spesifikke under planleggingsmøtet for å optimalisere ressursbruken bedre. Likevel konkluderte vi med at sprint-formatet i sin helhet fungerte svært godt for oss.

Figur 13 – Hjemskjermen ved enden av sprint

I tillegg til å reflektere over hva som fungerte bra, identifiserte vi også områder der vi kunne forbedre oss. Vi diskuterte nøye hva som manglet og hvilke tiltak som måtte tas for å møte prosjektets frister og målsettinger. Denne refleksjonsprosessen var avgjørende for å sikre at vi kunne lære av våre erfaringer og kontinuerlig forbedre våre arbeidsmetoder.

Avsluttende fase

For å avslutte og finpusse prosjektet, gjennomførte vi ulike tester på ulike kandidater. Vi gjennomførte først en brukertest, der vi opprettet et samtykkeskjema, lot kandidat lese over og gi samtykke. Vi stilte da spørsmål ved hva kandidat tenkte om appen, både negativt og positivt. Vi fikk både designorientert og funksjonalitetsorientert tilbakemelding, og vi gjorde endringer samsvarende. Et eksempel på en av endringene gjort er navigasjonsbaren, da vi fikk tilbakemelding om at den var vanskelig å bruke.



Figur 14 : Endring i navigasjonsbar etter brukertest

Etter brukertestene gjennomførte vi endringer med hensyn til tilbakemeldingene. Dette fungerte godt for å få innsikt til elementer av appen som vi som utviklerne kan ha oversett, og det er definitivt et av de mest produktive og lærerike elementene i utviklingsprosessen.

Til slutt gjennomførte vi en geriljatest der vi ba en tilfeldig deltaker om å gi tilbakemelding på appen vår, både når det gjaldt design og funksjonalitet. Vi mottok mange positive tilbakemeldinger, for eksempel om appens oversiktlige og rene design, samt at det ble verdsatt at appen tilpasset seg enhetens valgte tema. Samtidig fikk vi nyttige forslag til forbedringer, som å inkludere en knapp for oppdatering av data, redusere fontstørrelsen og legge til tilbakemelding ved knappetrykk for å bekrefte handlingen. Vi implementerte disse endringene i appen, og resultatet var en betydelig forbedring. Derfor konkluderte vi med at geriljatesten var en svært effektiv utviklingsaktivitet for oss, og hvis vi skulle videreutvikle appen, ville vi definitivt gjenta denne prosessen flere ganger.

Figur 15: Endringer i toppbar etter geriljatest

4.4 ITERASJONER

1. Iterasjon

Den første iterasjonen la grunnlaget for appens kjernefunksjonalitet altså alt backend som også var MVPen, i denne iterasjonen var det ikke noe fokus på frontend. Etter å ha valgt case, definert kravspesifikasjon og funksjonelle krav, splittet vi opp teamet i tre grupper for å håndtere ulike deler av MVPen.

En gruppe fokuserte på å hente og behandle data fra Locationforecast API til en egen datasource klasse. En annen gruppe fokuserte på å hente og behandle data fra IsobaricGRIB APIen til en egen datasource klasse.

Den siste gruppen tok ansvaret for å sette opp hjemmeskjermen som skulle vise dataen i MVPen samt viewModel og repositoriene som skulle tilrettelegge dataen fra datasourcene

Etter at alle gruppene hadde fullført deres del satte vi sammen komponentene slik at hjemmeskjermen skulle presentere dataen fra API ene riktig. I denne iterasjonen fikk vi også hjelp fra Haakoon Offernes fra Portal Space angående relevante vær parametere i en rakketoppskyting.

2. Iterasjon

Andre iterasjonen var det fokus på hvilken funksjonalitet vi ville ha utenom kravene til MVPen, her kom vi blant annet frem til at vi ville ha en kartskjerm og en parmeterskjerm.

Kartskjermen skulle gi brukeren et visuelt kart der de kan se sin egen posisjon, endre posisjon og se områder der man potensielt kan skyte opp raketter i Norge. Noen av oppgavene i kartskjerm var blant annet integrasjon av Google Maps APIen, implanteringa av juridiske områder fra Avinor (avinor, 2024) og displaying av informasjon angående hvert område samt GPS sjekk og implementering av Google Maps compose elementer.

Parameterskjermen ble designet slik at brukeren skal kunne endre parametere for oppskytning til deres egne krav. Her omfatter oppgavene blant annet å endre og legge til variabler i repositories og oppdatering av viewModelen slik at dataen oppdaterer seg. Vi hadde i første iterasjon hardkodet inn parameterne og lokasjoner.

Etter sprintreviewen fra første iterasjon bestemte vi oss for noen endringer i utviklings prosessen blant annet integrasjonen av et Trello tavle for bedre oversikt over oppgaver som skal løses. Med denne løsningen meldte hvert medlem av teamet seg selv på en oppgave som var i sammenheng med kartskjermen eller parameterskjermen.

3. Iterasjon

Tredje iterasjon, her var det fokus på frontend, å gjøre appen mer intuitiv og andre kravspesifikasjoner som ikke implementert enda. Her var det viktig for oss at alle var på

Figur 16 : Prototype av design

samme side når det kom til hvordan appen så ut, så vi fikk lagd noen prototyper til designet og designet vi ble enige om ble slikt.

Trello tavlen ble videreført fra forrige iterasjon slik at oppgaver ble effektiv fordelt og sporet. Noen av oppgavene i denne iterasjonen var blant annet å designe hjemskjermen, designe parameterskjerm, sette opp navigasjonsbar, implementere internettsjekk.

Videre i denne iterasjonen hadde vi også brukertester slik at vi kunne få gode tilbakemeldinger angående hvor intuitiv appen vår er og hva som eventuelt skulle endres. Vi fikk veldig mye god informasjon fra disse testene og fikk fikset dette under denne iterasjonen også.

Ved å fokusere på frontend og brukeropplevelse, fikk appen et mer polert og konsekvent utseende i tråd med de utarbeidede prototypene. Implementeringen av navigasjonsbaren og internettsjekken bidro også til å øke appens brukervennlighet.

4.5 KRAVSPESIFIKASJONER

De opprinnelige funksjonelle kravene omfattet visning av værinformasjon som nedbør, fuktighet og vindhastighet for ulike høyder for de neste to timene, samt kart over godkjente oppskytingssteder (se tabell 1).

Under utviklingsprosessen identifiserte vi behovet for å tilpasse visningen av vindhastigheten. Vår opprinnelige spesifikasjon støttet visning av vinddata for de neste to timene. Imidlertid, på grunn av begrensninger i de isobariske grib-filene fra vårt API, som presenterer vindhastighetsdata over tre timers intervaller, utvidet vi visningen til å omfatte de neste tre timene for vind over 10 m/s. Denne tilpasningen sikret at dataene var så nøyaktige som mulig, noe som er kritisk for våre brukere ved planlegging av rakettoppskytninger.

Vår initiale mangel på detaljert informasjon om juridiske bestemmelser for oppskytingssteder viste seg også å være en utfordring. Etter grundig undersøkelse innså vi at oppskyting krever spesielle tillatelser og er begrenset til spesifikke godkjente lufthavner. For å forenkle denne prosessen for våre brukere, integrerte vi et interaktivt Google-kart i appen, som viser alle godkjente oppskytingssteder med detaljert kontaktinformasjon for å skaffe de nødvendige tillatelsene.

Endringene i kravspesifikasjonen demonstrerer viktigheten av fleksibilitet og tilpasningsevne i programvareutvikling, spesielt når det gjelder å håndtere uforutsette tekniske og juridiske utfordringer. Denne erfaringen har styrket vårt team sin evne til å forutse og håndtere lignende situasjoner i fremtiden.

KRAVSPESIFIKASJON		
Type krav	#	Beskrivelse
Funksjonelle	1.1	Vise nedbør for neste to timer
	1.2	Vise fuktighet for neste to timer
	1.3	Vise vindhastighet for ulike høyder for neste to timer
	1.4	Vise oversikt over godkjente oppskytningsplasser
Ikke-funksjonelle	1.5	Systemet skal ikke krasje
	1.6	Systemet skal fungere på tvers av alle enheter
	1.7	Systemet skal tilby et brukervennlig grensesnitt
	1.8	Systemet skal ha robust feilhåndtering

Tabell 1: *Oversikt over de opprinnelige funksjonelle og ikke-funksjonelle kravene.*

4.6 TESTING

Android Studio tilbyr et godt testrammeverk som vi brukte til å gjennomføre enhetstesting for appen, samt andre biblioteker som mockk.io og junit4. Det er to ulike typer tester man kan kjøre i Android Studio, disse er instrumenterte tester og vanlige enhetstester.

De instrumenterte testene bruker emulatoren til å utføre sin jobb og fungerer direkte på ulike UI-komponenter. Selve denne testingen fikk oss til å skrive mer robust kode som gir oss mere oversikt for kodingen. Vi implementerte derfor contentDescriptions for alle funksjoner som lot oss aksessere de lettere underveis i testingen og var hjelpsomme for kodemiljøet vårt. Testingen går ut på at Android Studio bytter mellom skjermene og endrer på og leser verdier og sjekker om disse er som forventet.

Den vanlige enhetstesten bruker mockk.io til å lage simulerte data klasser slik at vi kan finne ut om disse er satt opp ordentlig. Testene bruker junit4 sin runTest for å kunne lage et simulert coroutine og kjøre et manipulert API kall slik at vi kan gjenkjenne klassen vi lager med en assert. (junit4, n.d)

Testingen var gjennomført i flere forskjellige faser av appen vår, og gjennom utviklinga måtte også testene våre endres til å tilpasse disse endringene. Ettersom at testene ble korrekte var vi mer trygge på at koden vår trygg å kjøre. Gjennom testingen finner vi ut mulige svakheter som finnes i koden vår og ordner opp i disse slik at de ikke gjenspeiler sluttproduktet vårt.

5.1 REFLEKSJONER

I vårt team, Team 48, har vi jobbet tverrfaglig ved å integrere ulike ferdigheter og perspektiver fra informatikk, design og prosjektledelse. Gjennom hele prosjektet har vi hatt jevnlige møter og workshops, hvor vi har delt innsikt og erfaringer for å sikre en helhetlig tilnærming til utviklingen av vår app. Denne tverrfaglige tilnærmingen har vært essensiell for å adressere både tekniske og brukervennlighetsmessige utfordringer.

For eksempel, i uke 10 da vi arbeidet med innsiktsarbeid og arkitekturskisse, kombinerte vi teknisk forståelse med brukerbehov, som vi hadde identifisert gjennom åpne intervjuer. Dette samarbeidet mellom utviklere og de som jobbet med innsikt hjalp oss med å definere en klar struktur for appen vår. I senere uker, som under Workshop/Crazy 8 i uke 11, arbeidet designere og utviklere sammen for å generere og evaluere ideer til appens brukergrensesnitt.

Utfordringer i utviklingsprosessen

En av de største utfordringene vi møtte var kommunikasjon på tvers av fagområder. Spesielt i begynnelsen, som under kickoff i uke 8, var det vanskelig å sørge for at alle forstod hverandres terminologi og mål. Dette førte til noen misforståelser og behovet for ekstra møter for å rette opp i feil. Vi fant ut at det var viktig å ha klare og dokumenterte beslutninger samt regelmessige oppdateringer for å holde alle på samme side.

En annen utfordring var å holde prosjektet innenfor tidsrammene. Forsinkelser i enkelte deler av prosjektet påvirket fremdriften i andre deler, som vi opplevde under

mobbprogrammeringen i uke 12. Denne arbeidsmetoden viste seg å være ineffektiv for oss på grunn av ulike kodestiler og rotasjonstider som skapte avbrudd. Vi lærte at det er viktig å tilpasse arbeidsmetoder etter teamets dynamikk og prosjektets behov.

Til slutt var en stor utfordring å balansere arbeidsmengden med andre fagområder, spesielt rundt eksamenstider, som vi så i uke 17. Vi måtte justere vår arbeidsplan og fokusere på bedre planlegging for å sikre at prosjektet ikke ble påvirket negativt av eksterne faktorer. Dette innebar å prioritere oppgaver mer effektivt og bruke verktøy som Trello for å holde oversikt over arbeidsflyten.

Refleksjoner over løsninger og forbedringer

Gjennom retrospektiven, som den vi hadde før påske i uke 12, har vi kontinuerlig vurdert hva som har fungert godt og hva som kan forbedres. Vi har lært at klare målsettinger, strukturert arbeidsflyt, og hyppig kommunikasjon er nøkkelen til suksess. Bruk av verktøy som Trello for å organisere oppgaver har hjulpet oss med å få bedre oversikt og struktur i arbeidet vårt.

Videre, brukertesting og geriljatesting i uke 19 ga oss verdifulle tilbakemeldinger som vi kunne bruke til å forbedre appen. Strenge, men konstruktive tilbakemeldinger fra brukere hjalp oss med å identifisere områder som trengte forbedring og justere design og funksjonalitet deretter. Dette har vært avgjørende for å skape en mer brukervennlig app.

Alt i alt har vår erfaring vist at tverrfaglig samarbeid, selv med utfordringer, er avgjørende for å lykkes med komplekse prosjekter. Ved å kontinuerlig evaluere og forbedre våre arbeidsprosesser, har vi klart å overvinne mange av de utfordringene vi har møtt og oppnådd våre mål.

AVSLUTNING

6.1 ERFARINGER FRA PROSJEKTET

Gjennom prosjektarbeidet i IN2000 har vi lært mye om tverrfaglig samarbeid, kommunikasjon og fleksibilitet. Her er våre viktigste lærdommer:

- **Oskar Wiig Melvold:** Et positivt arbeidsmiljø bidrar til et komfortabelt arbeidsområde hvor man kan ha det gøy og jobbe effektivt.
- **Johan Rondestvedt:** God planlegging og regelmessig arbeid gir de beste resultatene.
- **Peter Larsen Olaisen:** Teamaktivitetene gjorde at gruppearbeidet ble spennende.
- **The Dat Le:** Samarbeid er viktig!
- **Gard-Henrik Haftor Bjerkelund Gimse:** Kommunikasjon mellom pushing og merging i github er vikitg for at det ikke skal oppstå merge conflicts.
- **Wiktor Blaszkowski:** Det er gøy å jobbe med andre medstudenter på et prosjekt.

6.2 TIPS TIL NESTE ÅRS STUDENTER

Lær effektivt bruk av GitHub og planlegg med hensyn til eksterne faktorer som kan påvirke prosjektarbeidet, slik som andre eksamener, sykdom og pc-problemer.

REFERANSER OG KILDEFØRING

Meteorologisk institutt. (n.d.). *Locationforecast*. https://api.met.no/weatherapi/locationforecast/2.0/documentation (Hentet 13.05.2024)

Meteorologisk institutt. (n.d.). *Isobaricgrib*. https://api.met.no/weatherapi/isobaricgrib/1.0/documentation (Hentet 13.05.2024)

Avinor. (2024). *ENR 5.5 Aerial sporting and recreational activities*. https://ais.avinor.no/no/AIP/View/130/2024-03-21-AIRAC/html/eAIP/EN-ENR-5.5-en-GB.html (Hentet 14.05.2024)

JUnit. (n.d.). *JUnit 4: Testing framework for Java and Kotlin*. https://junit.org/junit4/ (Hentet 14.05.2024)

Google Developers. (n.d.). *Google Maps*. https://developers.google.com/maps (Hentet 14.05.2024)

Berntzen, M. (2020). *Smidige teknikker for teamarbeid*. Universitetet i Oslo. https://www.uio.no/studier/emner/matnat/ifi/IN1010/v20/ressurser/smidigeteknikkerforteamarbeid.pdf (Hentet 14.05.2024)