

Prosjektrapport

IN2000 - Vår 2022



Case 1 - Sykkelruter i Oslo

Gruppe 3

Edis Kukuruzovic (edisk)

Emir Kukuruzovic (emirkuk)

Erling Holte (erlinhol)

Jehan Khodabocus (jehank)

Michiru Tamura Nygaard (michirut)

Uvejs Sadiki (uvejss)

Veiledere: Emil Eriksmoen Stensland og Emil Damsgård Knutsen

Institutt for informatikk

Det matematisk-naturvitenskapelige fakultet

UNIVERSITETET I OSLO

20. mai 2022

1.0 Presentasjon	1
1.1 Oppbygging	1
1.2 Grunnlag for valg av case	1
1.3 Problemstilling og mål	2
1.4 Presentasjon av teamet	3
2.0 Brukerdokumentasjon	4
2.1 Plattform og tilgjengelighet	4
2.2 Målgruppe	5
2.3 Struktur og funksjonalitet	6
2.3.1 “Map”	7
2.3.2 “Routes”	8
2.3.3 “My Profile”	9
3.0 Kravspesifikasjon og modellering	10
3.1 Brukerhistorier	10
3.2 Krav	11
3.2.1 Systemkrav	11
3.2.2 Funksjonelle krav	12
3.2.3 Ikke-funksjonelle krav	13
3.3 Use-case	14
3.3.1 Tekstlig beskrivelse av use-case	15
3.4 Sekvensdiagram	18
3.5 Klassediagram	18
4.0 Produktdokumentasjon	19
4.1 Apparkitektur	21
4.1.1 Design patterns	21
4.1.2 UI-lag	22
4.1.3 Datalag	22
4.2 Kvalitetsegenskaper	23
4.3 Dokumentasjon av kode	23
4.3.1 UI	23
Intro	23
Map	24
Profile	26
Route	29
4.3.2 Datalag	30
4.4 Universell utforming	30
4.4.1 Definisjon på universell utforming	30
4.4.2 WCAG	30
4.4.3 Bruk av farge	31

4.4.4 Konsekvent navigering og sidetitler	31
5.0 Testdokumentasjon	32
5.1 Enhetstester	32
5.1.1 API-test	32
5.1.2 Enhetstesting av ViewModel og Fragment	33
5.2 Integrasjonstester	34
6.0 Prosessdokumentasjon	35
6.1 Smidig utvikling	35
6.1.2 Scrumban	35
6.2 Arbeidsverktøy	36
6.2.1 Arbeidsstruktur	36
6.2.2 Git	37
6.2.3 Brukerundersøkelser og design	37
6.2 Planleggingsfase 26. februar- 14. mars	37
6.2.1 Arbeidsdeling	39
6.3 Kravhåndteringsprosess	39
6.3.1 Forstudiefasen	39
6.3.2 Kravinnsamling og -analyse	40
6.3.3 Validering av kravspesifikasjoner	42
6.4 Sprintene	43
Sprint 1 - 14. - 21. mars	43
Sprint 2 - 21.-28. mars	43
Sprint 3 - 28. mars- 4. april	45
Sprint 4 - 4. april - 19. april	45
Sprint 5 - 19. april - 10. mai	46
6.5 Evaluering og håndtering av kravendring	47
6.5.1 Brukbarhetstesting og intervju	47
6.5.2 Evaluering av visuelt design	50
6.5.3 Utførte krav	51
6.6 Avsluttende fase - 10. mai - 20. mai	53
7.0 Refleksjon	53
8.0 Videre arbeid	54
8.1 LocationForecast & Stedstjenester	54
8.2 Bugs & loading	54
8.3 Profil	55
9.0 Konklusjon	56
10.0 Litteraturliste	58
11.0 Vedlegg	62

Vedlegg 1: Teamavtale	62
Vedlegg 2: Klassediagram	63
Vedlegg 3: Innledende spørreundersøkelse	63
Vedlegg 4: Intervjuguide	77
Vedlegg 5: Samtykkeskjema	79
Vedlegg 6: Høyoppløselig prototype	82
Vedlegg 7: Retrospektive møter	82

1.0 Presentasjon

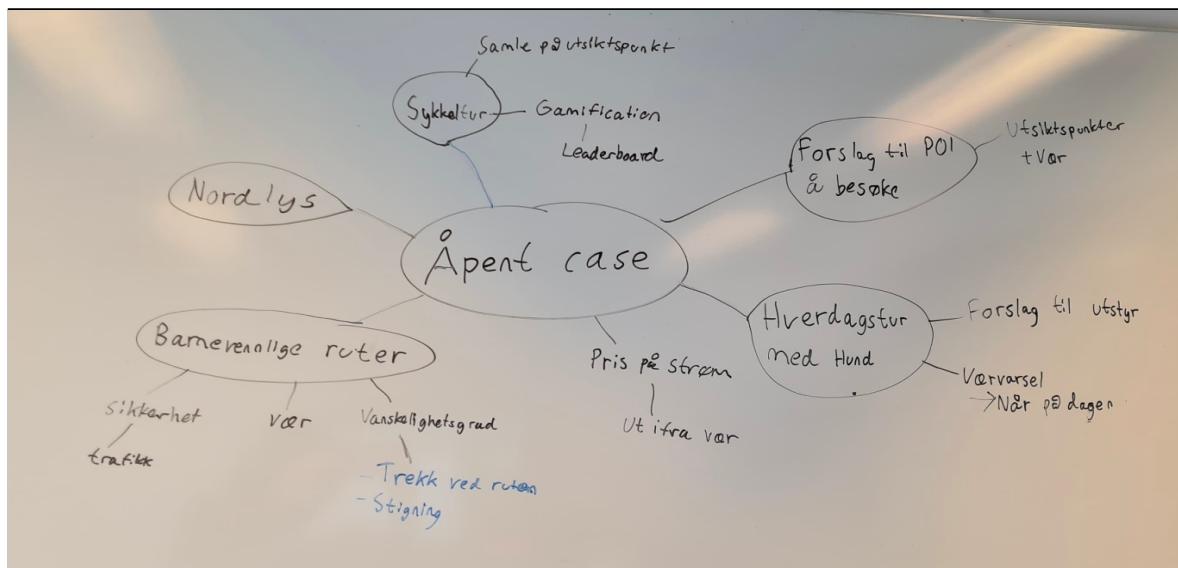
1.1 Oppbygging

Rapportens struktur er bygget opp i en kronologisk rekkefølge, der vi beskriver den trinnvise produktutviklingsprosessen. Innledningsvis presenterer vi gruppens medlemmer og valg av case, der vi forteller om våre målsettinger med applikasjonen. Den neste delen omhandler brukerdokumentasjon som er skrevet med hensyn på personer uten teknisk kunnskap. Her tar vi for oss applikasjonen spesifikasjoner og funksjonalitet samt målgruppen vi har rettet oss mot. Videre går vi dypere inn i denne målgruppen, hvor vi gjennom ulike typer intervjuer samlet inn informasjon med hensyn på utforming av kravspesifikasjoner. Kapittel 4 vil skrives med hensyn på mennesker med teknisk bakgrunn. Her vil vi ta opp applikasjonsarkitekturen, dokumentasjon av kode og universell utforming. Med hensyn på utviklingen av applikasjonen har det også blitt utført enhetstester og integrasjonstester av programvaren. På De siste kapitlene våre dokumenterer prosessen bak det ferdige produktet og eventuelle utfordringer.

1.2 Grunnlag for valg av case

Prosessen knyttet til valg av case (problemstilling) var tidskrevende, men ansett som svært viktig. Årsaken til dette er at prosjektet er langvarig, derfor mener vi at det er viktig at alle er enige om case for at interessen og motivasjonen skal opprettholdes gjennom hele prosjektet. Til å begynne med noterte vi noen stikkord om de ulike casene prosjektet tilbyr, samt utviklet vi hver noen frie case som vi skulle dele i plenum. Dette var ikke noe omfattende, men heller noe kort for å få et innblikk i gruppemedlemmernes ønsker og tanker. Denne prosessen fungerte som en form for idémyldring, der vi la vekt på lite innvendinger og kritikk mot andre forslag. Disse verdiene var viktige i denne fasen, da det bidro til en større mengde forslag å velge mellom. De ideene som fikk størst oppslutning blant gruppemedlemmene etter denne prosessen er oppsummert i figur 1.1. Blant alternativene vi vurderte var åpent case i form av nordlys-app, app for hundeeiere (anbefaling av turer etter vær) eller en turplanlegger, eller case 1.

Ut i fra disse ideene i figur 1.1 tok vi i bruk "vurderingskart" som et verktøy for rangering av ulike case basert på preferanse. På denne måten kunne vi se hvilke av de ulike casene som var foretrukket og basert på disse resultatene foreta en eliminasjonsprosess. Da kom vi raskt frem til at valget falt på case 1.



Figur 1.1: Idemydring av åpent case

1.3 Problemstilling og mål

Sykkel har lenge vært en viktig form for transportmiddel og har den siste tiden økt i popularitet på lik linje med den stadig urovekkende miljøkrisen. Denne økningen har myndighetene stimulert gjennom utbygging av sykkelinfrastruktur som sykkelparkeringer og sykkelbynettverket. Dette kan man særlig se til i hovedstaden som gjennom deres store satsning har klart å heve seg på listen over verdens mest sykkelvennlige byer (Steen, 2020). Slike forbedringer av sykkelinfrastrukturene gir syklister økte muligheter, men krever også mer av kommunen med tanke på informasjonsdeling, slik at syklister i Oslo for eksempel får med seg utbyggingen av nye sykkelruter.

Med hensyn på at sykling er en aktivitet som foregår ute vil ulike værforhold være en viktig faktor for syklister. Disse forholdene vil ha innvirkning på om mennesker velger å sykle i det hele tatt. Dette kan også være forhold som er spesifikke for sykkelruten, da forhold påvirker hvorvidt syklistene sykler visse ruter. Å ha andre alternativer til ruter vil være viktig for å motivere og engasjere mennesker til å sykle mer.

I sammenheng med det som er skrevet ovenfor har vi formulert et hovedmål for prosjektet;

Målet med prosjektet er å skape en applikasjon som motiverer til sykling, opplyse mennesker om Oslos ulike sykkelruter og gi syklister nytte væropplysninger knyttet til rutene de ønsker å sykle.

1.4 Presentasjon av teamet



Figur 1.2: Fra venstre til høyre: Uvejs Sadiki, Emir Kukuruzovic, Edis Kukuruzovic, Jehan Khodabocus, Michiru Tamura Nygaard og Erling Holte

Edis Kukuruzovic

Går på studieprogrammet informatikk: Digital økonomi og ledelse. Har lenge hatt et ønske om å jobbe med økonomi og programmering, og begynte derfor på IFI. Liker å programmere, og har derfor drevet mye med koding i løpet av prosjektet. Utenfor studiene er han en aktiv person som liker å drive med idrett.

Emir Kukuruzovic

Går på studieprogrammet informatikk: Digital økonomi og ledelse. Er som person struktureret, kreativ og er glad i å programmere. Gjennom prosjektet ønsker han å tilegne seg mer erfaring om smidig metodikk i praksis, mens han utenfor studiene er glad i å trenne.

Erling Holte

Studerer programmering og systemarkitektur. Liker godt å programmere, og er ansvarlig for Github i gruppa. I tillegg til studier liker Erling å være fysisk aktiv og spiller kontrabass i et studentorkester.

Jehan Khodabocus

Studerer informatikk: Design, bruk og interaksjon. Har en mastergrad innen helse. Etter studiet mitt har jeg utviklet en interesse i IT og programmering og begynte derfor på IFI. I løpet av dette prosjektet har jeg jobbet med både en del kode og design. Utover studier liker jeg å lese bøker og bruke tid med familien.

Michiru Tamura Nygaard

Studerer informatikk: design, bruk, interaksjon. Lidenskap i brukersentrert design, og liker å tenke på muligheter i design basert på konkrete data fra undersøkelse og brukertestning. Utover studier liker Michiru å gå på tur i skog med bestevennen sin; en 4 år gammel corgi.

Uvejs Sadiki

Studerer bachelor i informatikk: programmering og systemarkitektur. Kreativ, løsningsorientert og glad i å programmere. Når han ikke studerer liker Uvejs å tilbringe tid på treningssenteret og være sosial med venner.

2.0 Brukerdokumentasjon

2.1 Plattform og tilgjengelighet

Applikasjonen er utviklet for Android og benytter seg av API-nivå 23 (Marshmallow). Dette betyr at appen kan kjøres på Android-enheter og Android-emulatorer som bruker Android 6.0 og nyere, det utgjør 94,1% av hele Android-enhet økosystemet (Wang, 2021). Bakgrunnen for valg av API nivå er at vi ville nå så mange brukere som mulig, samtidig som vi får god funksjonalitet og sikkerhet. Det er også et API-nivå vi har god kjennskap til på grunn av arbeid med dette nivået på tidligere obligater.

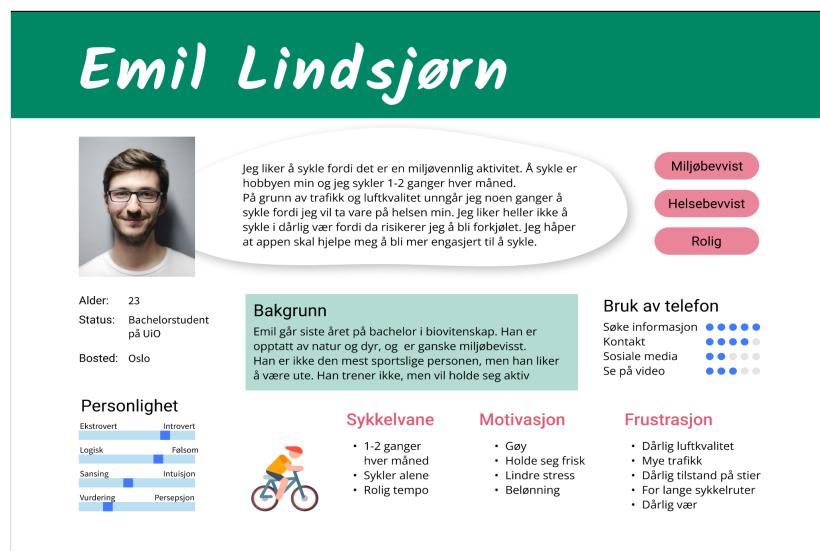
2.2 Målgruppe

Målgruppen vi bestemte oss for er universitetsstudenter som er miljøbevisste og sykler som hobby. Dette er en yngre gruppe mennesker i aldersgruppen 16 til 35 år. Vi velger å spesifisere målgruppen mot mennesker som har interesse for sykling som sykler rekreasjonelt minst 2-3 ganger i måneden. Videre innsnevrer vi målgruppen ytterligere ved å ta sikte på mennesker som har fokus på helsen sin og er miljøbevisste. Vi har valgt å fokusere på en slik målgruppe på grunn av deres interesse for sykling, helse og miljø. Dette er naturlig siden applikasjonen viser sykkelstier, og værdata som kan tolkes som helsedata og miljødata. På denne måten kan vi ta sikte på å dekke behovene til målgruppen vi realistisk sett mener har størst interesse og behov for en slik applikasjon.

Geografi:	Osloborger
Demografi:	16-35 år
Atferd:	<ul style="list-style-type: none"> • Sykler minst 2-3 ganger i måneden for rekreasjon • Over gjennomsnittet bevisst over miljøvalg • Bevisst over helsen sin

Tabell 2.1: Målgruppe

I sammenheng med valg av målgruppe har vi laget en persona med et brukerscenario. Persona er en beskrivelse av en fiktiv person som skal representere en vanlig bruker i det valgte segmentet. Dette mener vi var til stor nytte da det gav oss et ansikt på målgruppen, i tillegg til en felles forståelse for hvilken målgruppe vi ønsket å tilfredsstille behovene til.

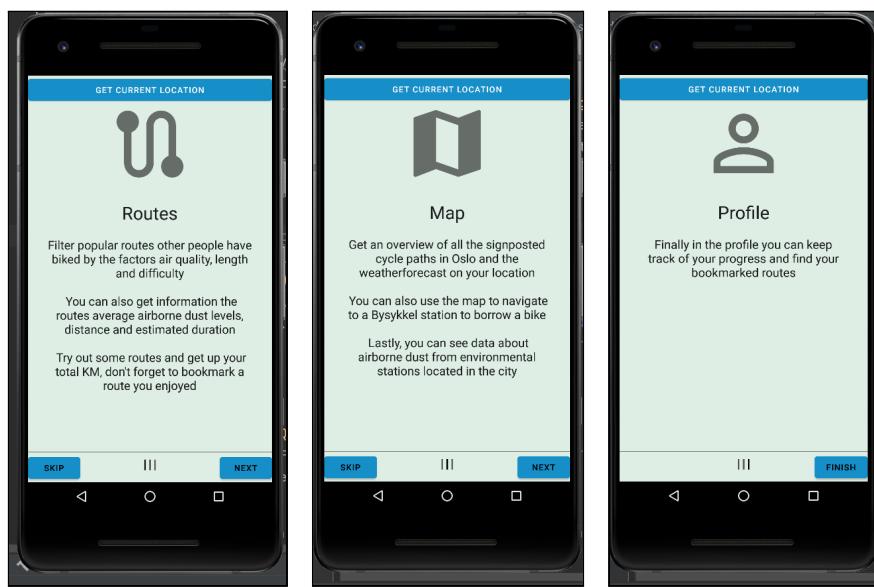


Figur 2.1: Persona

2.3 Struktur og funksjonalitet

Oppgaveteksten gir klare føringer for hva appen til case 1 skal inneholde. Det er et krav om minst ett værvarsel (Locationforecast eller Nowcast), minst en kilde til luftkvalitet fra MET eller NILU og sist men ikke minst anbefalte sykkelruter (feks fra Oslo Kommune). Til sammen vil disse kravene utgjøre vårt MVP (minimum viable product). Appen har som hovedfunksjonalitet å vise anbefalte sykkelruter, i tillegg til svevestøv data samt værvarsle. Utover MVP har vi implementert ekstra funksjonalitet som baserer seg på besvarelsene fra brukerundersøkelsen.

Applikasjonens er delt opp i tre vinduer som kan aksesseres ved hjelp av en menylinje i bunnen av appen. Disse tre vinduene er “Map” som er i midten av menylinjen, “Routes” relativt venstre og “My Profile” til høyre. Ved åpning av applikasjonen for første gang vil systemet vise en introskjem (splash-skjerm) som gir brukeren en introduksjon i de ulike vinduene i menylinjen. Hvis brukeren ønsker kan man aktivere stedstjenester ved å trykke på en knapp. Ved bruk av stedstjenester får brukeren været på nåværende lokasjon. Hvis brukeren ikke godtar å dele lokasjon bruker vi Ole Johan Dahls hus som utgangspunkt for værvarsle. Disse funksjonene kommer vi nærmere inn på i avsnittene under, der vi redegjør for alle de tre vinduene.



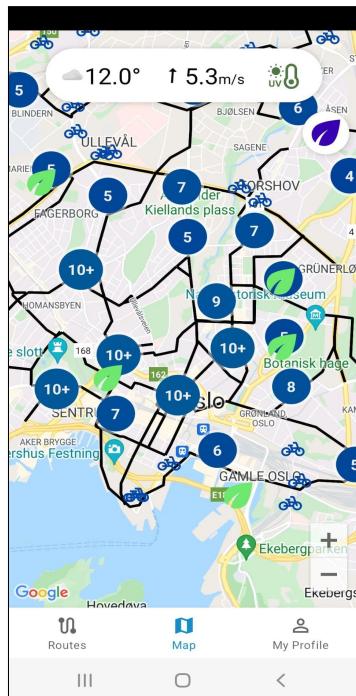
Figur 2.2: Introskjem

2.3.1 “Map”

“Map” er applikasjonens hovedvindu, det vil si at hver gang appen åpnes vil dette være det første brukeren møter på. Dette vinduet viser et kart zoomet inn på en blå markør som

representerer brukerens posisjonen. Selve kartet er hentet fra Google Maps (Android Developers, 2022). Som standard viser kartet viktige lokasjoner, og du kan zoome inn og ut. Kartet visualiserer i tillegg data om sykkelnettverket i Oslo (Oslo Kommune API), bisykkelstasjoner (Bisykkel API) og luftkvalitetsstasjonene (NILU API). Sykkelnettverket over skiltede sykkelruter i Oslo er vist med svarte linjer på kartet, der disse i tillegg er klikkbare (viser rute ID). Videre vises også bisykkelstasjonene og deres kapasitet gjennom klikkbare blå sykler på kartet. Presentasjonen av sykkelstasjonene er gjort på en dynamisk måte, da stasjonene blir “clustret” til sirkler avhengig av zooming. I toppen av høyre side har vi lagt inn en knapp som slår av og på visning av luftkvalitetsstasjonene i Oslo. Luftkvalitetstasjonene er vist ved blader, der fargen på bladet viser til mengden av svevestøv. Ikonene er også klikkbare. Bladene endrer farge utifra luftkvaliteten på stasjonen. Det går fra grønn, oransje, rød og til slutt lilla.

Øverst på kartet er værdata for den aktuelle timen presentert. Til venstre finner vi temperatur, samt ikon for værbeskrivelsen. I midten ligger vindstyrke og vindretning representert med pil. Til høyre har vi et symbol for UV-stråling, der fargen på logoen representerer nivået av UV-stråling. Fargene har samme ordning som for luftkvalitetsstasjoner.



Figur 2.3: “Map”

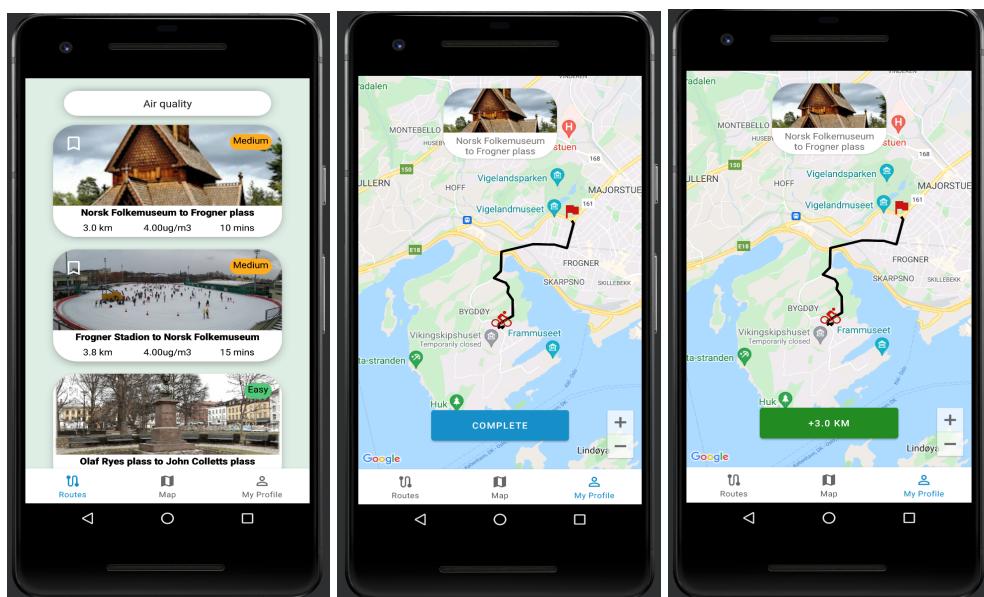
2.3.2 “Routes”

Ved å trykke på “Routes” i menylinjen blir du tatt til et vindu som har som hovedfunksjonalitet å vise en liste over ti anbefalte sykkelruter i Oslo (Bsykkel API). Disse rutene blir presentert i kort, der man finner et bilde assosiert med ruten (Google/ app logo dersom det ikke er noe bilde), samt informasjon om start- og sluttposisjon. Kortet viser rutens distanse, gjennomsnittlig svevestøvsnivå, predikert tid og vanskelighetsgrad.

Vanskelighetsgrad er markert med både skriftlig beskrivelse (easy, medium, hard) kombinert med farge (grønn, oransje og rød). Det siste elementet på kortene er en favoritt knapp i venstre øvre hjørnet som ved et trykk går fra hvit til gul, og lagrer denne i for brukeren i “My Profile” vinduet. Dette forutsetter at brukeren er innlogget, ellers har ikke denne knappen noe funksjonalitet.

På toppen av vinduet ligger en knapp som fører til en drop-down meny med de tre alternativene; “Length”, “Difficulty” og “Air quality”, og sorterer listen med hensyn på det alternativet bruker velger (max til min).

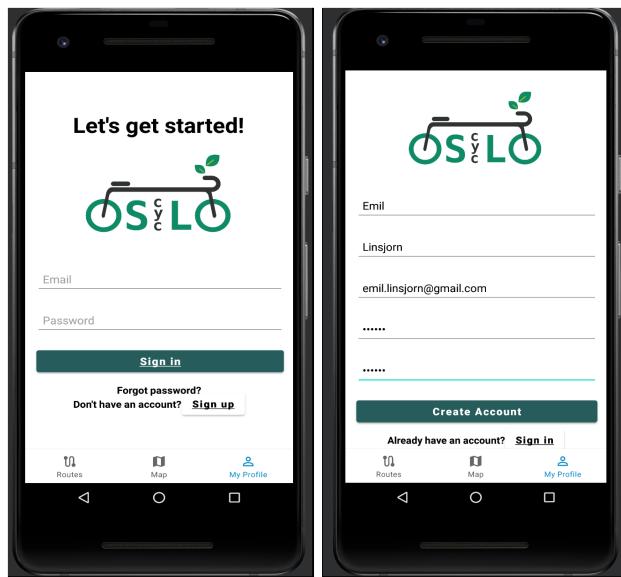
Disse kortene for ruter er klikkbare, der bruker ved et klick blir sendt til et vindu som viser et minikart med den valgte ruten tegnet opp på kartet. I øvre del av vinduet vil navn og bilde av ruten vises (samme som på kortene) med start- og sluttposisjon. I bunnen er det en knapp der brukeren kan registrere fullført rute, og videre trykke igjen (knapp med antall km for ruten) for å bli sendt tilbake til “Routes” vinduet. Denne har lignende oppførsel, da den krever at du er innlogget for registrering i “My Profile”.



Figur 2.4: “Routes”

2.3.3 “My Profile”

Ved å trykke på “My Profile” knappen blir brukeren bedt om å opprette en profil før brukeren kan få tilgang til profilsiden. Brukeren kan få tilgang til de fleste funksjonene uavhengig av om de er innlogget med en profil. Imidlertid vil brukeren gå glipp av personlig informasjon i form av total tilbakelagt kilometer og lagring av favorittruter. Hvis brukeren allerede har en konto kan brukeren logge inn ved å fylle ut e-post og passord. Hvis brukeren ikke har en konto fra før kan hen opprette en konto ved å trykke på “Sign up” knappen.



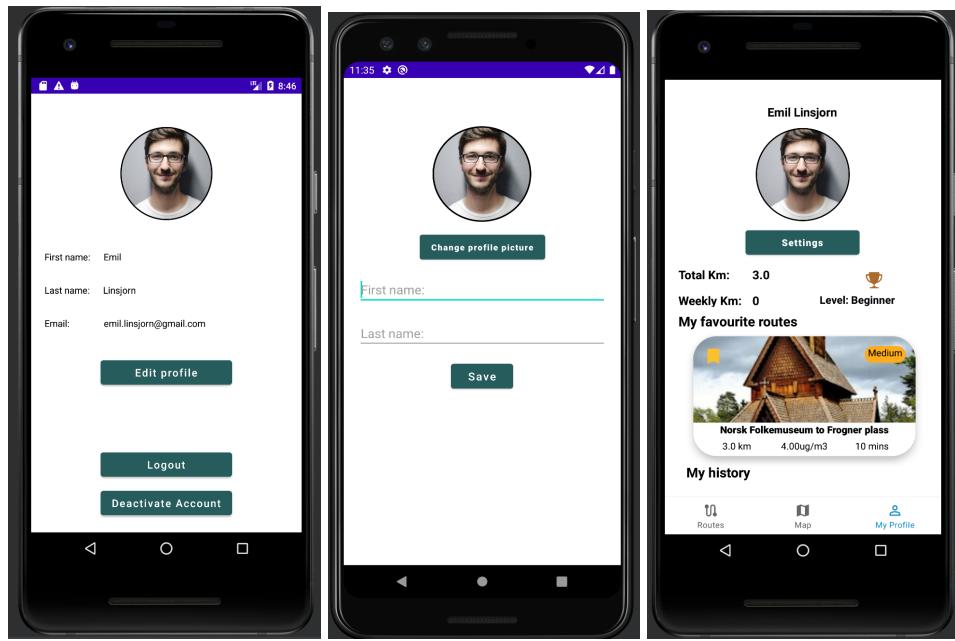
Figur 2.5 : Sign in og Sign up fragments

Etter at brukeren har fylt ut alle feltene kan hen trykke på “Create account”. Hvis alle feltene er riktig fylt ut vil brukeren motta en popup-melding om at kontoen er opprettet og vil bli sendt til hovedsiden til appen som er “Maps”. Etter opprettelse av kontoen kan brukeren trykke på “My Profile” og brukeren vil ha en oversikt over sin personlig informasjon. Informasjon innebærer navn, distanse dvs. antall kilometer syklet og ukentlig kilometer syklet, brukerens nivå (nybegynner, mellomliggende og avansert) basert på antall kilometer syklet og favoritrutene sine.

Ettersom brukeren registerer fullførte ruter i Routes oppdateres antall kilometer syklet. Pokalen vil bytte farge til enten bronse for nybegynner, sølv for mellomliggende eller gull for avansert.

Under “settings” kan brukeren se registrert informasjon samt logge ut av appen og deaktivere kontoen sin. Ved å trykke på logg ut blir brukeren henvist tilbake til “Maps”. Ved å trykke på “Edit profile” får brukeren mulighet til å redigere informasjonen sin samt bytte profilbildet

sitt. Ved å trykke "Save" får brukeren en bekrefstelse om at endringene er oppdatert og hen blir sendt tilbake til "Maps".

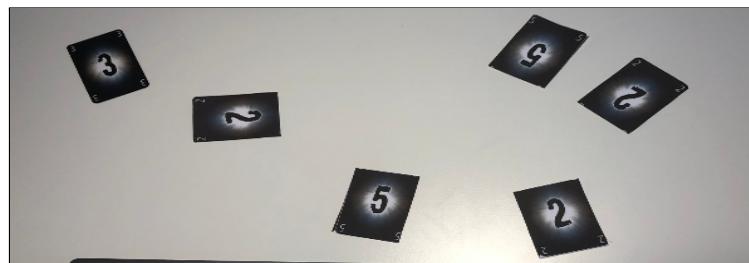


Figur 2.6: "Account Settings, Edit Account Settings og My Profile"

3.0 Kravspesifikasjon og modellering

3.1 Brukerhistorier

I tabellen under kan du finne brukerhistorier som vi skrev ut i fra hva brukerne ønsket og hva vi så for oss kunne vært nyttig for målgruppen vår. Brukerhistoriene med høy prioritering ble gitt mest oppmerksomhet og er det vi mener er hovedfunksjonaliteten til applikasjonen. Vi brukte poker planlegging (også kalt Scrum poker) til å estimere hvor lang tid vi trodde hver brukerhistorie skulle ta. Dette gikk ut på at hvert gruppemedlem valgte et kort med tallene 1, 2, 3, 5, 8, 13, 20, 40 og 100 for å representerer meningen deres. Til slutt ble vi enige om en felles prioritering.



Figur 3.1: Poker planlegging

Brukerhistorie	Prioritet	Estimert tid
Som en astmatiker som er opptatt av miljø og helse, ønsker jeg å ha oversikt over sykkelruter med god luftkvalitet	Høy	13
Som en aktiv syklist vil jeg se hvor jeg kan hente og droppe av en Bysykkel nærmere der jeg skal	Høy	2
Som en syklist som sykler for fornøyelse ønsker jeg å få forslag til nye ruter jeg kan sykle.	Høy	40
Som en syklist ønsker jeg å kunne se informasjon om været for å planlegge sykkelruten min	Høy	1
Som en syklist ønsker jeg å kunne se hvor i Oslo det er skiltede sykkelveier for å trygt kunne kjøre i byen	Høy	5
Som en konkurranseinnstilt syklist ønsker jeg å kunne lagre fremgangen min for å sammenligne med venner	Middels	100
Som en bruker ønsker jeg en introduksjon til appen, slik at jeg vet hvordan jeg bruker den riktig	Middels	5
Som en bruker som ikke sykler ofte, ønsker jeg å motta poeng og oppnå milepåler for å motivere meg til å sykle videre	Lav	40
Som en bruker ønsker jeg å kunne lage egne ruter og dele de med andre	Lav	100
Som en bruker ønsker jeg å kunne se hvilke ruter som har minst trafikk for å unngå å sykle sammen med biler	Lav	8

Tabell 3.1: Brukerhistorier

3.2 Krav

3.2.1 Systemkrav

Kravene vi har stilt til maskinvaren applikasjonen

- Android som operativsystem
- Minimum API-nivå 23

Ytterligere har vi lagt til følgende krav som må oppfylles:

- Android enheten må ha stedstjenester påslått for fullstendig funksjonalitet
- Enheten må ha tilgang til internett

3.2.2 Funksjonelle krav

Basert på brukerhistoriene ovenfor (3.3 Brukerhistorier) har vi utarbeidet følgende funksjonelle krav i tabellen nedenfor. Tabellen inneholder en oversikt over de kravene og deres prioritet. Vi baserer oss på skalaen for prioritet som ble presentert på forelesningen til Bergersen og Sjøberg. De deler inn i kan, bør til må krav (Bergersen & Sjøberg, 2022).

Funksjonelle krav	Prioritet
I det brukeren åpner applikasjonen vil de få en kort introduksjon til applikasjonens funksjonalitet	Bør
Vise brukerposisjon på kartet	Bør
Applikasjonen skal vise kart over alle de skiltede sykkelrutene i Oslo	Må
Applikasjonen skal ha en funksjon på kartet som viser tilstanden til luftkvaliteten i Oslo	Må
Applikasjonen skal vise sanntidstemperatur der hvor brukeren befinner seg	Bør
Applikasjonen skal vise vindstyrke og vindretning i sanntid der hvor brukeren befinner seg	Kan
Vise nivåer for UV-stråling i sanntid fra brukerens posisjon	Kan
Applikasjonen skal gi brukeren forslag på stier med hensyn til faktorer som rutelengde, luftkvalitet og vanskelighetsgrad	Må
Gi brukeren mulighet til å legge til en sti under favoritter	Bør
Applikasjonen skal gi brukeren mulighet til å se hvilke veier som er trafikkerte	Kan
Vise gjennomsnittlig stigning for hele sykkelruten	Kan
Brukeren skal kunne lage egne ruter og lagre dem	Kan
Brukeren skal kunne markere ruter som de har kjørt.	Bør
Applikasjonen skal fungere for horisontal og vertikal	Bør

skjerm.	
Applikasjonen skal fungere i dark mode	Kan

Tabell 3.2: Funksjonelle krav

3.2.3 Ikke-funksjonelle krav

Nedenfor har vi utformet en tabell med de ikke-funksjonelle kravene ved applikasjonen.

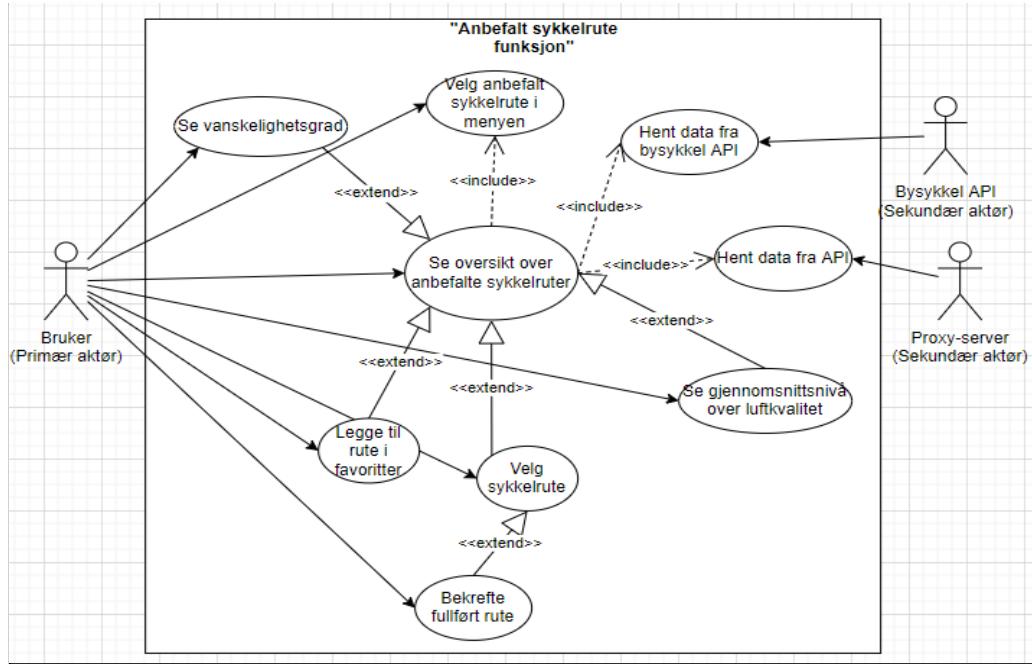
Dette er krav som støtter opp om og beskriver hvordan de funksjonelle kravene skal implementeres av systemet. Sommerville deler inn de ikke funksjonelle kravene i tre kategorier: organisatoriske krav, produktkrav og eksterne krav (Sommerville, 2015, s.108).

Ikke-funksjonelle krav	Krav beskrivelse
Koden skal være modularisert, oversiktlig og selvforklarende	Organisatorisk krav
Systemets kodehåndtering skal gjøres i GitHub	Organisatorisk krav
Kotlin skal brukes som programmeringsspråk	Organisatoriske krav
Appens filer skal være logisk inndelt i en mappestruktur	Organisatoriske krav
Det skal være skrevet enhetstester	Organisatoriske krav
Det skal være skrevet integrasjonstest for navigasjon	Organisatoriske krav
Appen skal støtte Activity Lifecycle	Organisatoriske krav
Kall mot API skal gjøres pålitelig	Organisatoriske krav
Appen skal implementere design-pattern MVVM	Organisatoriske krav
Antall "Warnings" og "Errors" fra IDE-en bør være så lavt som mulig	Produktkrav
Appen skal være universelt utformet (WCAG 2.1)	Ekstern krav/ produktkrav
Appens ulike funksjoner (generelt) skal lastes inn under 2 sekunder	Produktkrav
Appen skal ikke krasje når den brukes	Produktkrav
Appens layout skal være dynamisk og responsiv	Produktkrav
Applikasjonen krever minimum 45 MB lagringsplass	Produktkrav

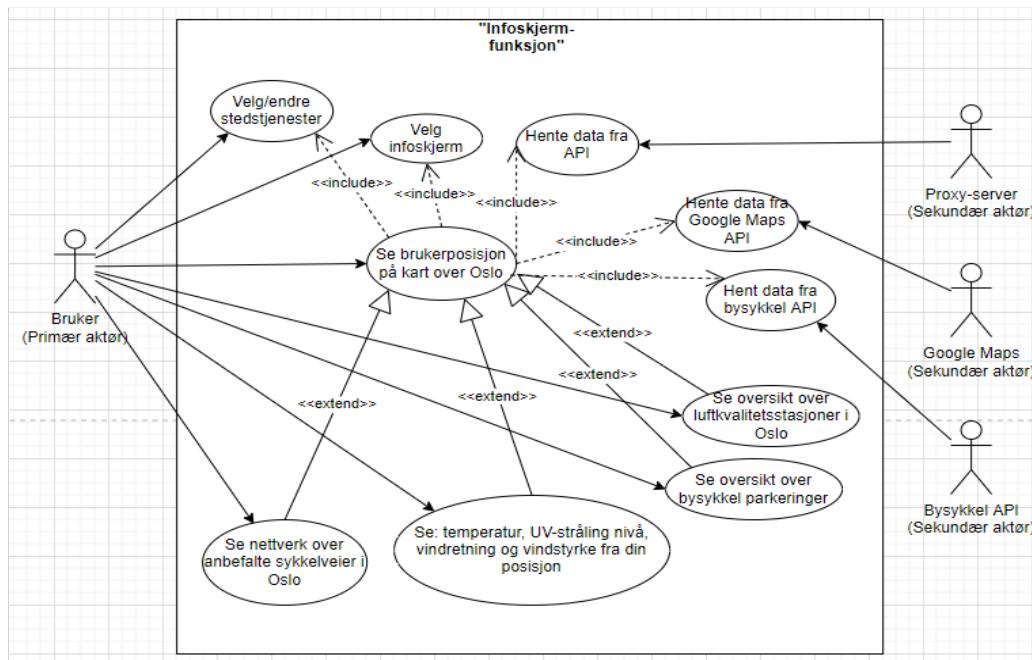
Tabell 3.3: Ikke-funksjonelle krav

3.3 Use-case

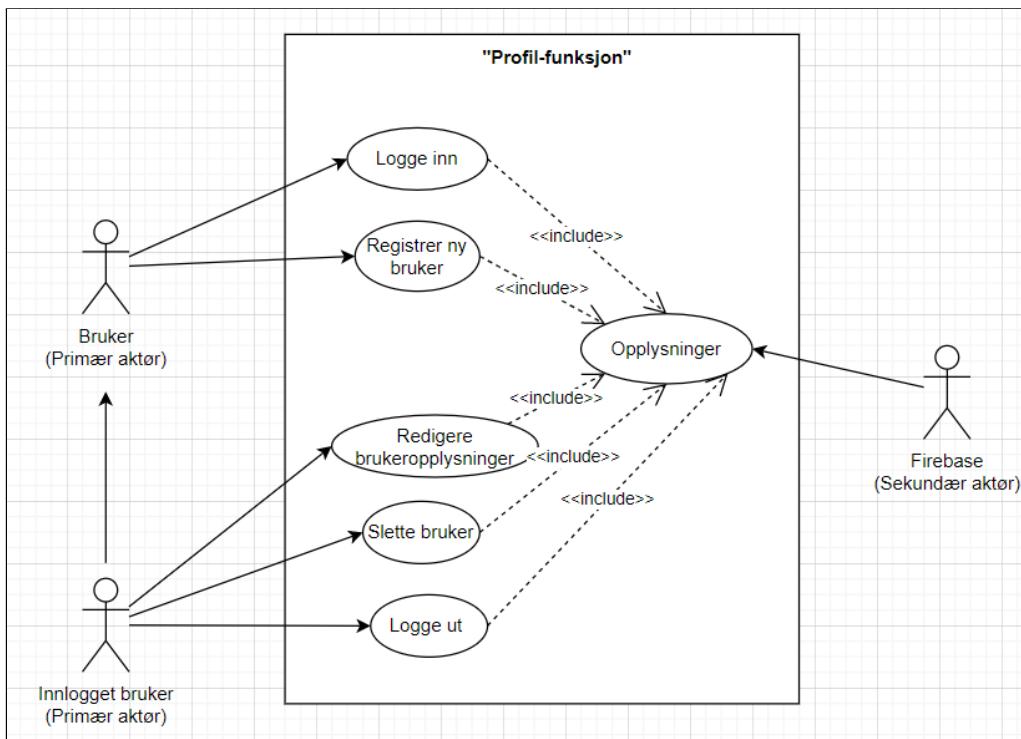
Use case viser interaksjon mellom et system og omgivelsene. Den tar utgangspunktet i primæraktørs mål og hvordan sekundæraktører assisterer dette målet gjennom systemet (Lindsjørn, 2022).



Figur 3.2: "Anbefalt sykkelrute funksjon"



Figur 3.3: "Infoskerm-funksjon"



Figur 3.4: "Profil-funksjon"

3.3.1 Tekstlig beskrivelse av use-case

Vi vil nå beskrive interaksjonene mellom bruker og system for de ulike use-case diagrammene illustrert ovenfor. Her vil vi se på flyten i noen av use-casene, samt eventuelle avvik som kan forekomme i flyten (alternativ flyt). I noen av beskrivelsene vil vi kombinere flere funksjoner på grunn av liknende flyt, samtidig som vi kun vil fokusere på de viktigste use-casene. For eksempel vil det være lik alternativ flyt som det første use-caset i tabell 3.4 når det gjelder API-er som ikke svarer.

Use-case navn:	Se værdata, sykkelnettverk, bysykkelstasjoner og luftkvalitetsstasjoner
Aktører:	Bruker, Google Maps utviklere, Bisykkel utviklere, Proxy-server
Pre-betingelse	<ul style="list-style-type: none"> • Installert Oscyclo-app • Internett tilgang
Post-betingelse	<ul style="list-style-type: none"> • Systemet skal vise brukerens posisjon på kartet over Oslo, samt værdata fra denne posisjonen. I tillegg skal sykkelnettverket, bysykkelstasjoner og luftkvalitetsstasjoner vises visuelt på et kart over Oslo. Alt skal visualiseres på samme kart.

Hovedflyt	<ol style="list-style-type: none"> 1. Brukeren åpner applikasjonen 2. Systemet spør om brukerposisjon i introskjerm 3. Brukeren klikker seg gjennom introen (“next”)eller klikker “skip”, samt trykker “Allow” på bruk av stedstjenester 4. Systemet viser nå et kart over Oslo som visualiserer punkter og linjer som representerer: luftkvalitetsstasjoner, bisykkelstasjoner, sykkelnettverk, brukerposisjon. I tillegg viser systemet værdata fra brukerens posisjon i et view. 5. Bruker trykker på trykker på luftkvalitetsstasjon/bisykkelstasjon/en del av sykkelnettverket. 6. Systemet viser nivå for pm10 (svevestøv) / kapasitet over sykkelparkering / navnet på ruten.
Alternativ flyt	<p>Alternativ flyt 1: (hovedflyt: 3) Brukeren tillater ikke deling av posisjon</p> <ol style="list-style-type: none"> 3.1 Brukeren trykker “ikke tillatt” på brukerposisjon 3.2 Systemet gir beskjed nedsatt brukeropplevelse, da data hentet gjennom brukerposisjon ikke er tilgjengelig 3.3 Systemet viser trinn 5 i hovedflyt, der brukerposisjon er satt til OJD hus. <p>Alternativ flyt 2: (hovedflyt: 4) Proxyserver returnerer NULL</p> <ol style="list-style-type: none"> 4.1 Et av API-ene via proxyserver returnere NULL 4.2 Systemet gir en feilmelding i form av en “toast”, gir beskjed om spesifikt hvilket API som returnerer null. 4.3 Bruker trykker “OK” på melding 4.4 Systemet vil fortsatt fungere, men viser ikke data fra de gjeldende API-ene <p>Alternativ flyt 3: (hovedflyt 4) Google Maps API-et svarer ikke</p> <ol style="list-style-type: none"> (A3) 4.1 Google Maps sitt API svarer ikke (A3) 4.2 Systemet gir en feilmelding i form av en “popup”. 4.3 Bruker trykker “OK” på melding 4.4 Systemet vil returnere til trinn 1 i hovedflyten (denne loopen krever ikke trinn 2 og 3 i hovedflyten)

Tabell 3.4: Tekstlig beskrivelse av use-casene; Se værdata, sykkelnettverk, bisykkelstasjoner og luftkvalitetsstasjoner.

Use-case navn:	Registrere en rute som utført
Aktører:	Bruker, Google Maps utviklere, Bisykkel utviklere, Firebase (database)
Pre-betingelse	<ul style="list-style-type: none"> • En innlogget Oscyclo-app på hovedvinduet • Internett tilgang

Post-betingelse	<ul style="list-style-type: none"> Appen har lagret den valgte ruten under fullførte ruter (i “My Profile”)
Hovedflyt	<ol style="list-style-type: none"> Brukeren trykker på “Routes” i menylinjen. Systemet sender brukeren til “Routes”. Bruk er scrollar i listen over ruter og klikker på en av dem. Systemet tar brukeren til et nytt vindu med informasjon om selve ruten. Brukeren klikker “Complete”. Systemet lagrer dette i “My Profile”

Tabell 3.5: Tekstlig beskrivelse av use-case; Registrere en rute som utført.

Use-case navn:	Legge til sykkelrute som favoritt
Aktører:	Bruk, Google Maps utviklere , Bisykkel utviklere, Firebase (database)
Pre-betingelse	<ul style="list-style-type: none"> En innlogget Oscyclo-app på “Routes” vinduet Internett tilgang
Post-betingelse	<ul style="list-style-type: none"> Appen har lagret den valgte ruten under favoritter i “My Profile”
Hovedflyt	<ol style="list-style-type: none"> Brukeren trykker på knappen som representerer favoritter Systemet lagrer ruten i favoritter og markerer logoen gul (favoritt)

Tabell 3.6: Tekstlig beskrivelse av use-case; Legge til sykkelrute som favoritt

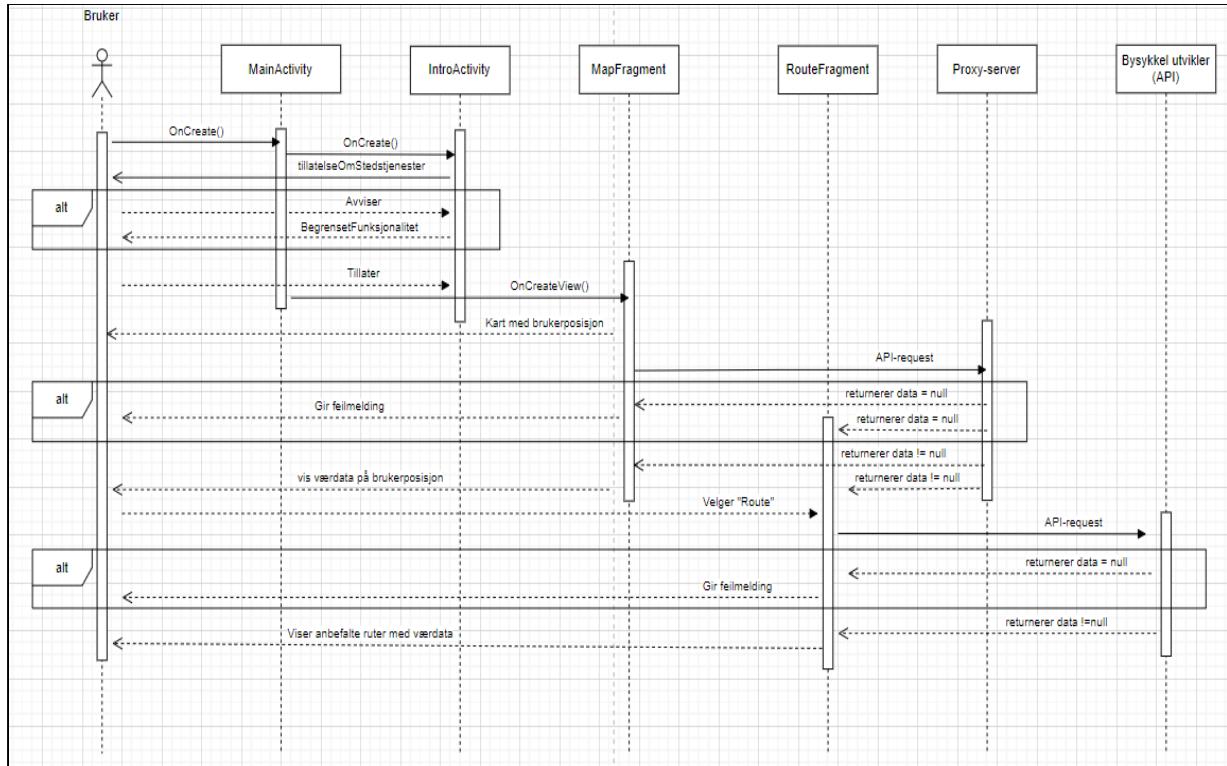
Use-case navn:	Sortere de anbefalte rutene
Aktører:	Bruk, Google Maps utviklere , Bisykkel utviklere
Pre-betingelse	<ul style="list-style-type: none"> Oscyclo-app på “Routes” vinduet Internett tilgang
Post-betingelse	<ul style="list-style-type: none"> Appen viser rutene i en sortert rekkefølge (min til max/ lett til vanskelig) basert på valgt parameter
Hovedflyt	<ol style="list-style-type: none"> Brukeren trykker på spinneren. Systemet viser en liste over alternativer; “Length”, “Air quality”, “Difficulty”. Brukeren velger ønsket parameter å sortere listen med hensyn på. Systemet viser en listen over ruter basert på valgt parameter (min til max/ lett til vanskelig)

Tabell 3.7: Tekstlig beskrivelse av use-case; Sortere de anbefalte rutene

3.4 Sekvensdiagram

Nedenfor har vi laget et sekvensdiagram som trinnvis viser interaksjonen i systemet for et par av use-casene.

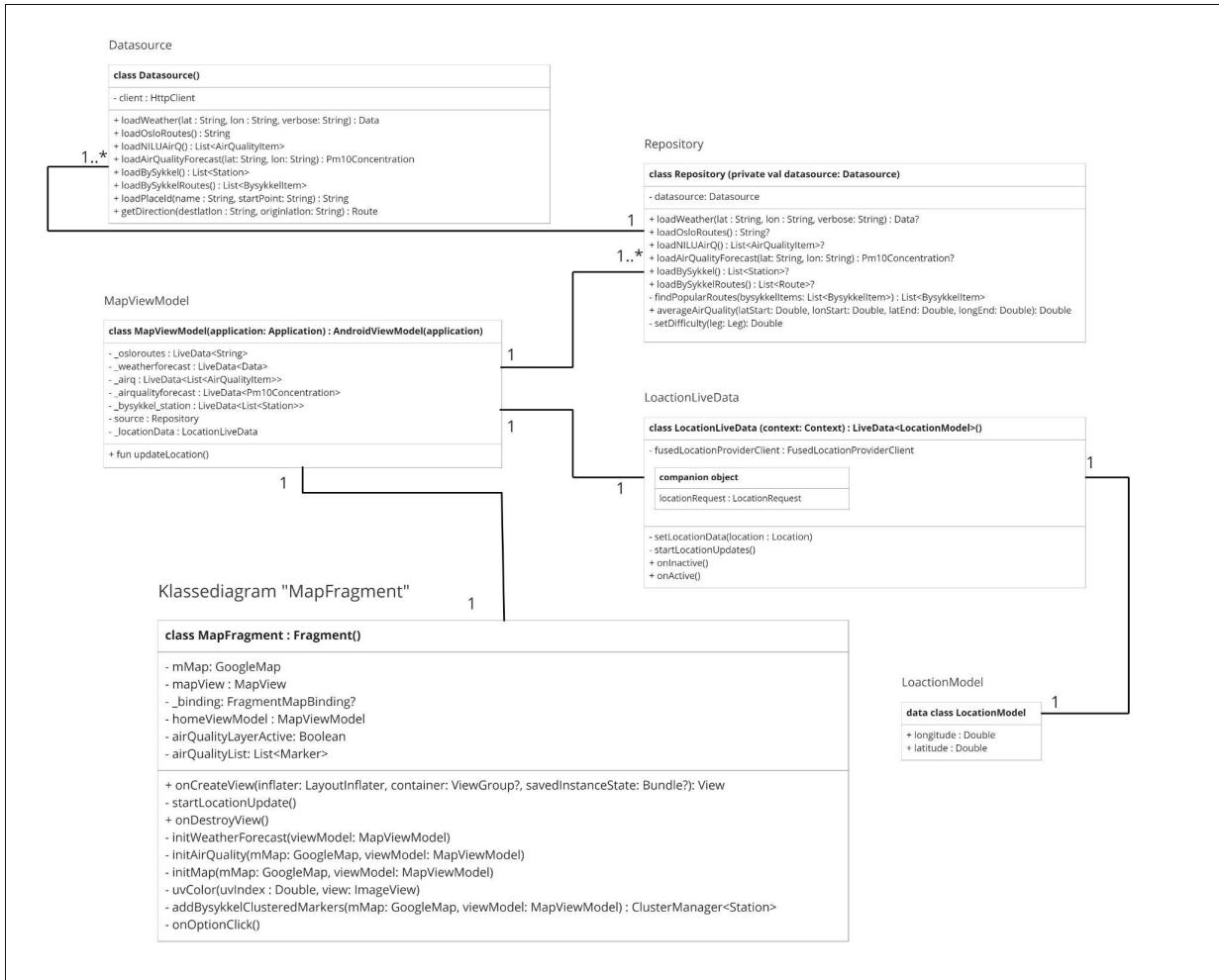
- Se posisjonen sin på et kart
- Sortere anbefalte ruter med hensyn på værdata
- Se værdata på din posisjon



Figur 3.5: Sekvensdiagram

3.5 Klassediagram

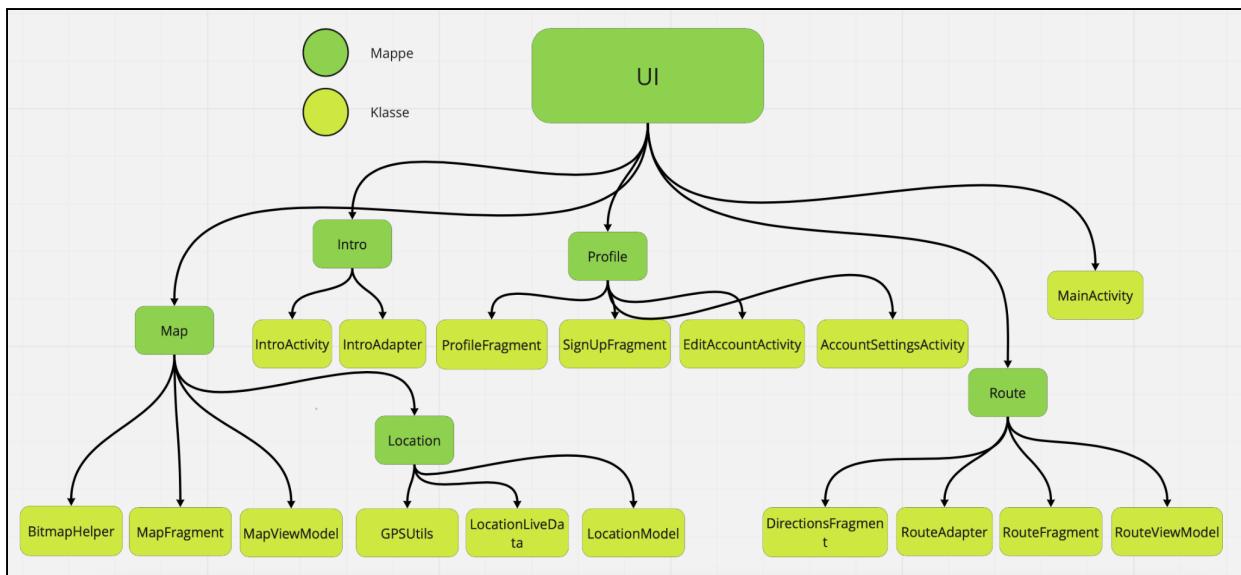
Klassediagram kan ha ulik detaljnivå (Sommerville, 2015, s.150). Vi har valgt å modellere kodenært, og relasjonene viser hvordan det ser ut i vårt program. Her har vi valgt å vise "MapFragment", mens resten av klassediagrammene ligger i Vedlegg 2.



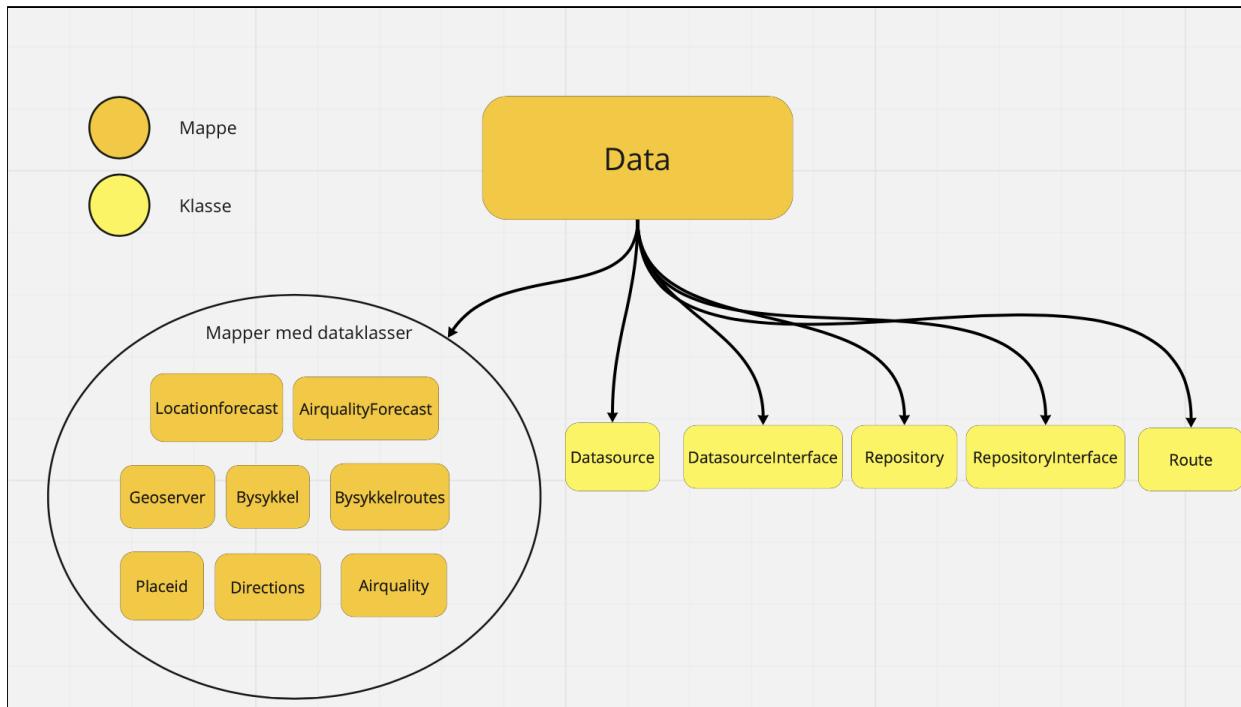
Figur 3.6: Klassediagram "MapFragment"

4.0 Produktdokumentasjon

Prosjektet er tydelig strukturert i mapper (packages) med lik funksjonalitet. Under er en oversikt over mappestrukturen til henholdsvis UI og data. Alle klasser er programmert i Kotlin, mens vi bruker XML til design.



Figur 4.1: Mappestruktur for UI



Figur 4.2: Mappestruktur for data

Valg av API

Fra Met benytter vi AirqualityForecast og LocationForecast. Fra NILU benytter vi oss av Airquality som gir oss oversikt over luftkvalitetstasjonene i Oslo.

Fra bymiljøetaten benytter vi oss av “Skiltede sykkelruter”. Dette kaller vi for geoserver i mappestrukturen, ettersom den er på GeoJson-format. Dette er API-et som viser sykkelnettverket tegnet på kartet.

Fra Oslo Bysykkel benytter vi oss av API-ene sanntidsdata og historiske data. Disse kaller vi for henholdsvis Bysykkel og bisykkelroutes. Fra bisykkelroutes får vi start og sluttkoordinater samt tittel på start og sluttspunkt. I tillegg deler API-et informasjon om Oslo Bysykkel stasjoner, der brukere plukker opp og dropper av syklene sine.

Fra Google bruker vi API-ene placeid og directions. Placeid benytter vi for å kunne finne frem bilder til rutene. Placeid returnerer en id utifra navnet på ruten. Denne id-en benytter vi til å hente bilde for ruten. Altså er ikke bildet på ruten hentet fra bisykkel-API-et, men fra Google. Directions gir oss en anbefalt sti fra startpunkt til sluttspunkt på tidspunktet brukeren åpner appen. Altså tar den anbefalte ruten henrys til trafikk på det aktuelle tidspunktet. Directions gir oss også avstand (km) og varighet (minutter) på ruten.

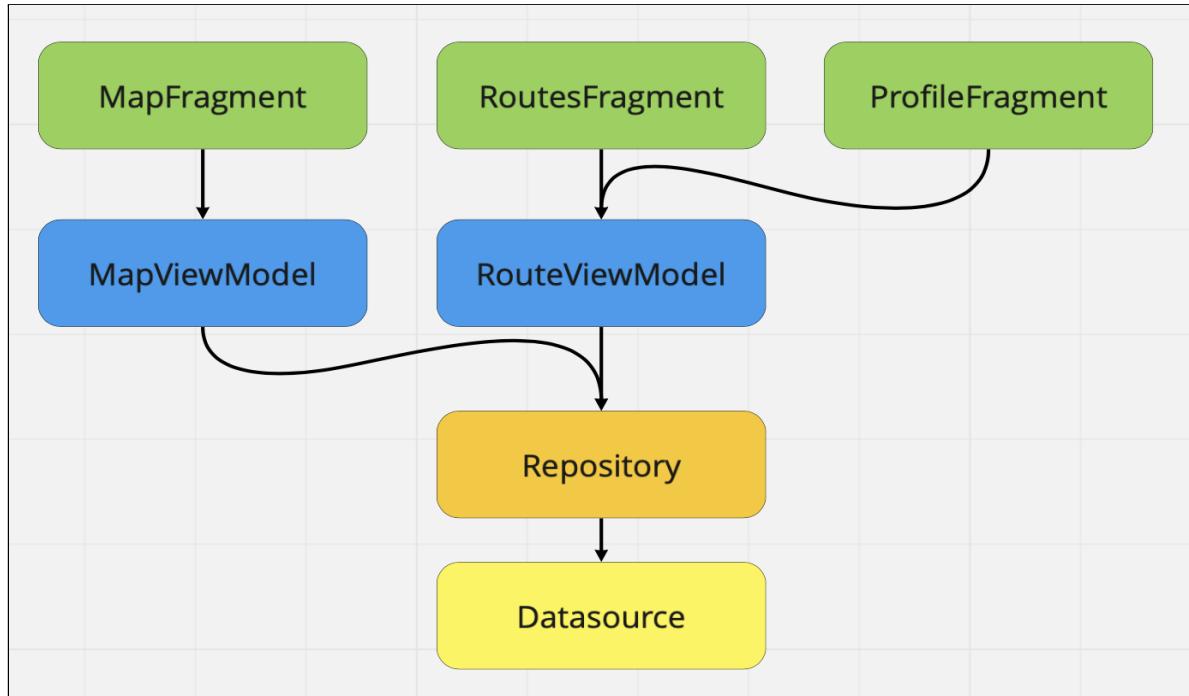
4.1 Apparkitektur

4.1.1 Design patterns

Design pattern er i læreboka definert som “A general reusable solution to a commonly occurring problem within a given context in software design” (Sommerville, 2019). Fra andre obligatoriske oppgave hadde vi lært om MVVM-arkitekturen i Android og ønsket å ta med oss dette videre inn i prosjektarbeidet. Andre design-patterns vi bruker er Factory-pattern for ViewModels og Dependency injection for testing.

To viktige prinsipper med tanke på arkitektur er kohesjon og kobling. Det er etterstrebet å ha høy kohesjon i applikasjoner (Lindsjørn, 2022). Høy kohesjon innebærer at elementer inne i en modul (kan være package eller klasse) hører sammen. Dette er ivaretatt ved at vi har delt inn i flere ulike fragments som har et begrenset ansvarsområde. Ved bruk av MVVM-arkitekturen får vi høy kohesjon blant annet gjennom å skille UI-laget fra datalaget (Android Developer , 2022). Dette gjør det videre enklere å endre koden, og kan dermed forhindre teknisk gjeld. Videre etterstreber man lav kobling, et objekt skal ikke være koblet til mange andre objekter. Dette er også ivaretatt gjennom MVVM-arkitekturen.

Under er en skisse av arkitekturen av hovedfunksjonaliteten i appen vår. Til tross for at det å sette opp en arkitektur kan være mye jobb i starten av prosjektet hindrer det at vi får stor teknisk gjeld (Martini, 2022). Vi har virkelig også sett nytten av å bruke tid på å ha en god arkitektur før vi koder, da det var trivielt for oss å utvide med flere datakilder (API-er) etterhvert som vi utviklet mer funksjonalitet i appen.



Figur 4.3: Skisse av App-arkitektur

4.1.2 UI-lag

UI-laget har ansvar for å presentere applikasjonsdata på skjermen. Android developers deler det videre inn i UI-elementer og Stateholders (Android Developer, 2022). UI-elementene er fragmentene MapFragment, RoutesFragment og ProfileFragment. Stateholders er MapViewModel og RouteViewModel. RouteViewModel er en delt ViewModel (Android Developers, 2021) da både ProfileFragment og MapFragment benytter seg av rutene som ViewModel holder på. Dataen er persistent selv etter endringer i activity lifecycle. I android er det slik at en rotasjon av appen fører til at aktiviteten lages på nytt, vi håndterer dette ved MVVM arkitekturen (Android Developers, 2021).

4.1.3 Datalag

Datalaget inneholder forretningslogikk (Android Developer, 2022). Datalaget består av et Repository og en Datasource. Datasource er delt inn i ulike metoder, der hver metode henter data fra et API. I starten av utviklingsprosessen hadde vi ikke et Repository-lag, da

kompleksiteten var liten og vi ikke skulle kombinere flere Datasources (API-er). Dette ble imidlertid nødvendig da vi skulle kombinere flere datakilder: luftkvalitet, bilder fra google, directions og rutene fra Oslo bisykkel. Dette for å for å opprettholde høy kohesjon og fasilitere for enhetstesting - single responsibility (Lindsjørn, 2022).

4.2 Kvalitetsegenskaper

ISO 25010 er en standard som tar for seg aspekter ved evaluering av IT-systemer (ISO, 2022.). På grunn av prosjektets omfang har vi prioritert enkelte krav vi anser som viktigst. Vi har lagt vekt på brukbarhet (usability) og vedlikeholdbarhet (maintainability). Disse egenskapene har vært med på å forme hvordan vi har jobbet. I innledningen brukte vi tid på å skissere appen, og har gjennom hele prosjektet vært i kontinuerlig kontakt med brukere gjennom undersøkelser. Ved å følge Android sine retningslinjer om god app-arkitektur (blant annet ved bruk av MVVM) har vi skrevet god kode med lite teknisk gjeld (lett å vedlikeholde). Vi kunne muligens ha implementert flere funksjoner raskt ved å ikke bry oss om arkitektur i like stor grad, men vedlikeholdbarhet har vært svært viktig for oss.

4.3 Dokumentasjon av kode

4.3.1 UI

Intro

Intro skjermen tar i bruk ViewPager for å kunne slide gjennom de ulike layout xml-filene: screen_one (Routes), screen_two (Map) og screen_three (My Profile). Det er laget en PrefManager som skal holde på SharedPreference som skal sjekke om applikasjonen er åpnet for første gang. IntroAdapter ekstender PagerAdapter og gir oss riktig rekkefølge for de ulike layout-ene som resulterer i sekvensen vist i applikasjonen.

Til slutt ble en aktivitet kalt IntroActivity laget. Denne aktiviteten holder på de ulike layout-ene og setter IntroAdapter til ViewPager for å få til slide funksjonen. For å få opp intro skjermen først, satte vi denne som “Launcher” i AndroidManifest.xml og kaller på MainActivity gjennom aktiviteten IntroActivity. Som tidligere nevnt vil PrefManager holde på tilstanden til applikasjonen om den er kjørt for første gang eller ikke. Hvis applikasjonen er kjørt for første gang, vises intro skjermen, ellers vil den hoppe rett til MainActivity.

```
private fun setupViewPager() {  
    introAdapter = IntroAdapter(context: this)  
    viewPager?.adapter = introAdapter  
    viewPager?.addOnPageChangeListener(pageChangeListener)  
}
```

Figur 4.4 Metode som setter opp IntroAdapter og setter denne til ViewPager

Map

I mappen Map ligger funksjonalitet for kartet som viser sykkelruter, luftkvalitet og værvarsel. MapFragment holder på hovedfunksjonaliteten, og observerer data fra ViewModel som legges inn på UI-elementene. Vi må sjekke for null fordi data fra API kommer som “platform type”- altså kan de være null (Kotlin, 2022).

Kartet tar tid å laste inn, og vi bruker derfor en callback-funksjon (getMapAsync) som gir oss et kart når det er klart til bruk. Videre kaller vi funksjoner som legger UI-en på kartet.

```
mapView.getMapAsync { map ->  
    mMap = map  
    initMap(map, homeViewModel)  
    initAirQuality(map, homeViewModel)  
    val bySykkelC = addBysykkelClusteredMarkers(map, homeViewModel)  
    map.setOnCameraIdleListener {  
        bySykkelC.onCameraIdle()  
    }  
}
```

Figur 4.5: Kode i onCreateView som setter opp alt det visuelle på kartet

For å vise sykkelrutenettverket fra Oslo kommune lager vi et GeoJsonLayer som vi legger på kartet. Dette kan vi gjøre fordi Oslo kommune sitt API har GeoJson format.

```

private fun initMap(mMap: GoogleMap, viewModel: MapViewModel) {
    // Add a marker in Oslo and move the camera
    val ojd = LatLng(latitude: 59.94410, longitude: 10.7185)
    mMap.animateCamera(CameraUpdateFactory.newLatLngZoom(ojd, zoom: 15f))
    var layer : GeoJsonLayer
    viewModel.osloroutes.observe(viewLifecycleOwner) { geo ->
        if (geo != null) {
            layer = GeoJsonLayer(mMap, JSONObject(geo))
            val layerStyle = layer.defaultLineStringStyle
            layerStyle.isClickable = true
            layer.setOnFeatureClickListener { it: Feature! -
                Toast.makeText(context, it.getProperty("rute"), Toast.LENGTH_SHORT).show()
            }
            layer.addLayerToMap()
        }
    }
}

```

Figur 4.6: initMap visualiserer sykkelrutenettverket over Oslo

Noe annet vi har vektlagt er clustering av bysykkelstasjonene. Til dette bruker vi ClusterManager som også er et objekt fra Google Maps. Dataklassen som skal clustres må implementere, altså implementere interface (grensesnittet) ClusterItem.

```

private fun addBisykkelClusteredMarkers(mMap: GoogleMap, viewModel: MapViewModel) : ClusterManager<Station> {
    // Create the ClusterManager class and set the custom renderer.
    val clusterManager = ClusterManager<Station>(context, mMap)
    clusterManager.renderer =
        StationRenderer(
            context,
            mMap,
            clusterManager
        )
    // Add the places to the ClusterManager.
    viewModel.bisykkelStation.observe(viewLifecycleOwner) { it: List<Station>! -
        if (it != null) {
            clusterManager.addItem(it)
            clusterManager.cluster()
        }
    }
    return clusterManager
}

```

Figur 4.7: Funksjon som legger til bysykkelstasjoner clusteret

For å følge Androids anbefalte arkitektur (MVVM) implementerte vi location som et LiveData-objekt. På den måten slipper vi kontinuerlige API-kall, og kan selv styre hvor ofte vi skal oppdatere stedstjenestene i appen.

```

private fun startLocationUpdate() {
    homeViewModel.locationData.observe(viewLifecycleOwner) { it: LocationModel!
        if (GPSEnabled.isGPSEnabled) {
            mMap.isMyLocationEnabled = true
            mMap.uiSettings.isMyLocationButtonEnabled = true
        } else {
            mMap.isMyLocationEnabled = false
            mMap.uiSettings.isMyLocationButtonEnabled = false
        }
        homeViewModel.updateLocation()
    }
}

```

Figur 4.8: Observerer locationdata og setter UI for location på kartet. Oppdaterer værmelding ved updateLocation()

I klassen LocationLiveData kan vi endre på parameterene til LocationRequest utifra hvor viktig lokasjon er for appen vår. For oss er det kun været som vil oppdatere seg, så vi er derfor ikke interessert i veldig høy nøyaktighet, og ønsker heller å spare batteritid på telefonen. Derfor har vi satt interval til 60 sekunder.

```

companion object {
    val locationRequest : LocationRequest = create().apply { this: LocationRequest
        interval = 60000
        fastestInterval = 50000
        priority = PRIORITY_HIGH_ACCURACY
    }
}

```

Figur 4.9: Locationrequest

Profile

Profile Fragment

ProfileFragment bruker databasekobling for validering og autentisering av brukere samt å vise eller sende brukere til riktig fragment eller aktivitet. Brukeren kan navigere fra ProfileFragment til EditAccountActivity gjennom profileSettingsButton.setOnClickListener.

```

// Navigate to Account settings
_binding.profileSettingsButton.setOnClickListener{ it: View!
    startActivity(Intent(context, AccountSettingsActivity::class.java))
}

```

Figur 4.10: Navigasjon fra ProfileFragment til EditAccountActivity i OnCreateView()

Ved pålogging kalles metoden isLoggedIn(). I denne metoden sjekkes det om det finnes en aktiv brukersesjon for å bestemme hvilken fragment som skal vises.

Logikken for Sign in er en del av ProfileFragment. Dette utføres i signInUser() metoden, under signInButtonMain.setOnClickListener{}. Vi brukte Firebase som er en Google database som bruker Firebase Authentication SDK og har metoder for å opprette og administrere brukere som bruker e-postadressene og passordene deres for å logge på (Google Developers, 2022). I denne metoden er det implementert enkle valideringsregler som sjekker at brukeren har skrevet inn riktig legitimasjon. Når brukeren har fylt inn de nødvendige feltene og trykker på “Sign in” vil det ble vist en progressbar i påvente av at brukeren skal bli pålogget. Autentiseringen er implementert gjennom FirebaseAuth sin signInWithEmailAndPassword() metoden (Google Developers, 2022).

```
else -> {
    binding.signInProgressBar.visibility = View.VISIBLE

    FirebaseAuth.getInstance().signInWithEmailAndPassword(email, password)
        .addOnCompleteListener { task ->
            if(task.isSuccessful){
                startActivity(Intent(context, MainActivity::class.java)
                    .addFlags( flags: Intent.FLAG_ACTIVITY_CLEAR_TASK or Intent.FLAG_ACTIVITY_NEW_TASK))
                activity?.finish()
                Log.d(ContentValues.TAG, msg: "signInWithEmailAndPassword:success")
                Toast.makeText(this.requireActivity(), text: "Signed in successfully",
                    Toast.LENGTH_LONG).show()
            } else {
                // If sign in fails, display a message to the user.
                Log.w(ContentValues.TAG, msg: "signInWithEmailAndPassword:failure", task.exception)
                Toast.makeText(this.requireActivity(), text: "Failed to sign in. Please try again!",
                    Toast.LENGTH_LONG).show()
                FirebaseAuth.getInstance().signOut()
                //Progress bar disappears
                binding.signInProgressBar.visibility = View.GONE
            }
        }
}
```

Figur 4.11: Del av signInUser() metoden med signInWithEmailAndPassword() metoden

Brukerdatala hentes i metodene setUserDistance() og setUserInformation(). setUserInformation() metoden bruker Firebase Realtime Database som er en skybasert database som lagre og synkroniser data med en NoSQL skydatabase. (Google Developers, 2022). Firebase-data skrives til en Firebase database-referanse og hentes ved å knytte en asynkron lytter til referansen. Lytteren utløses én gang for den opprinnelige tilstanden til dataene og igjen hver gang dataene endres (Google Developers, 2022). I

setUserInformation() blir en HashMap opprettet ut i fra dataSnapshot.value sin innhold. Deretter kan brukerens informasjon aksesseres basert på key-value prinsippet.

```
private fun setUserInformation() {
    val firebaseUser = FirebaseAuth.getInstance().currentUser!!.uid
    val databaseReference = FirebaseDatabase.getInstance().getReference(path: "Users")

    databaseReference.child(firebaseUser).addValueEventListener(object : ValueEventListener {
        override fun onDataChange(dataSnapshot: DataSnapshot) {
            if (dataSnapshot.exists()) {
                val user = dataSnapshot.value as HashMap<*, *>

                if (user.size > 0) {
                    val profilePicture = user["image"].toString()
                    val firstName = user["firstname"].toString()
                    val lastName = user["lastname"].toString()
                    val level = "Level: Beginner"

                    _binding.profileFullName.text = "${firstName} ${lastName}"
                    _binding.profileUserLevel.text = level
                    _binding.profileImageMain.let { it: CircleImageView
                        Glide.with(fragment: this@ProfileFragment)
                            .load(profilePicture)
                            .placeholder(R.drawable.profile)
                            .into(it)
                    }
                }
            }
        }

        override fun onCancelled(databaseError: DatabaseError) {
            Log.d(tag: "Could not download information for user: $firebaseUser ", databaseError.message)
        }
    })
}
```

Figur 4.12: setUserInformation() metoden

setUserDistance() metoden brukes for å hente brukerens tilbakelagt kilometer. Basert på antall kilometer syklet blir pokalen oppdatert på brukerprofilen. For testformål ble antall kilometer satt mellom 0.0 til 10.0 for nybegynnere, 10.1 til 20.0 for mellomliggende og 20.1 til 99999999.0 for avanserte.

SignUpFragment

Dersom en bruker ikke har en konto fra før kan brukeren trykker på Sign up linken på ProfileFragment for å bli sendt over til SignUpFragment gjennom signInSignUp.setOnClickListener{ }. Samtidig ved å trykke på

`signUpSignInButton.setOnClickListener{}` brukeren blir sendt fra SignUpFragment til ProfileFragment.

I `createAccount()` blir brukeren bedt om å registrere seg med fornavn, etternavn, e-postadresse, samt passord med minimum 6 tegn. Når brukeren har fylt inn de nødvendige feltene og trykker på “Create account” vil det ble vist en progressbar i påvente av at kontoen skal bli opprettet. Alle brukere får en standard profilbilde ved registrering.

Validering av brukeren er implementert gjennom `createUserWithEmailAndPassword` metoden. `saveUserInformation()` metoden blir kalt når en ny bruker er opprettet. Dersom opprettelse av brukeren mislykkes vil en feilmelding bli vist (Google Developers, 2022). `saveUserInformation()` metoden lagrer `firebaseUser ID` og en referanse til databasen (`databaseReference`). En “Users” node blir opprettet og i den blir all informasjon i forhold til brukeren lagret i databasen. Hver bruker har en unik ID som man kan identifisere brukeren med. Dataen blir lagt i en `HashMap` og `HashMap` settes i `setValue` som oppdaterer databasen. I `addOnCompleteListener{}` varsler brukeren hvis dataen er lagt til databasen eller ikke. Om den er vellykket blir brukeren sendt til “Maps” ellers får de en feilmelding.

EditAccountActivity

I `EditAccountActivity` kan brukeren kan oppdatere profilen i `updateUserInformation()`. Det nye bildet blir lagret i en mappe i Storage i Firebase og den gamle url-en blir erstattet med en ny url under brukerens unik uid. Her kan man også endre for- og etternavn.

AccountSettingsActivity

`userInformation()` metoden viser det lagrede informasjon fra databasen om brukeren som fornavn, etternavn, e-postadresse og profilbilde. `settingsLogoutButton.setOnClickListener{}` gjør at brukeren blir logget ut og returnert til `MainActivity`.

Route

Her skjer mesteparten av funksjonaliteten i adapteren. Android anbefaler at man bruker en adapter når man bruker recyclerview (Android Developer, 2022). Det meste av koden er en standard måte å opprette en adapter, men i tillegg skal `cardView` være clickable.

```

fun openMapFromCoordinates(position: Int, card : View) {
    SelectedRoute.currentPolyline = exampleList[position].directions.overview_polyline.points
    SelectedRoute.currentView = card
    findNavController(routeFragment).navigate(R.id.directionsFragment)
}

```

Figur 4.13: Funksjon som viser ruten på kartet

openMapFromCoordinates henter koordinatene vi får fra Google Maps sitt Direction-API og sender brukeren til skjermen som tegner ruten på kartet.

4.3.2 Datalag

Vi bruker Ktor med Gson for henting fra API, og har dataklasser for hvert API under package med navn Data. I tillegg har vi en dataklasse Route, som kombinerer data fra flere av API-ene. Ideen med dataklassen Route er at den skal være generell nok til å kunne erstattes med andre datakilder senere, noe som Android påpeker at er god praksis (Android Developers, 2022). Den eneste avhengigheten i Route er til dataklassen directions.Route fra Google Directions som gir directions fra startpunkt til slutt punkt.

4.4 Universell utforming

4.4.1 Definisjon på universell utforming

Universell utforming har som mål å gjøre det lettere å orientere og bevege seg i samfunnet for så mange som mulig (NTNU, 2020). Dette kan oppnås ved å minske funksjonsgapet i samfunnet. Funksjonsgapet oppstår når samfunnets krav ikke er tilpasset individets behov. Universell utforming fremstår når vi øker individets forutsetning (briller og rullestol) og senker samfunnets krav (gode tekniske løsninger) (uutilsynet, 2022).

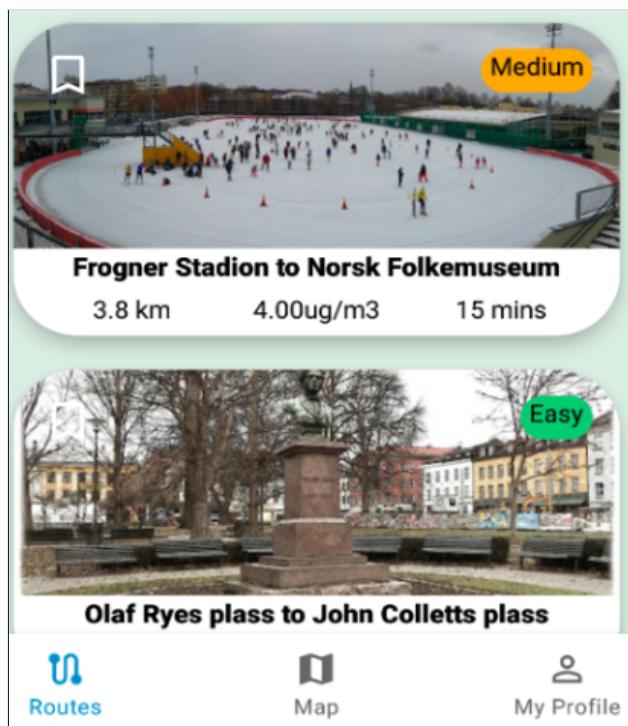
4.4.2 WCAG

“Web Content Accessibility Guidelines (WCAG) 2.0 er en internasjonal standard for universell utforming av nettsider” (difì, 2016). WCAG deles opp i 4 prinsipper: mulig å oppfatte, mulig å betjene, forståelig og robust. I tillegg inneholder hvert prinsipp underpunkter i form av suksesskriterium. Med tanke på at applikasjonen vår er avhengig av internett for å kunne ta den i bruk, må den følge kravene til universell utforming. På lik linje som nettsteder skal applikasjoner følge minst 35 av 61 suksesskriterium i WCAG 2.0

(Utilsynet, 2022a). Basert på den begrensede tiden for prosjektet har vi ikke oppnådd dette målet, men har heller valgt å fokusere på noen viktige kriterier, nevnt nedenfor.

4.4.3 Bruk av farge

Farge er en viktig komponent i applikasjoner, med tanke på at 1 av 12 menn (8 %) har en form for fargeblindhet (Norse, 2022). I WCAG-standarden legges det frem at 1.4.1 er for presentasjon av data som ikke bygger utelukkende på farger (Utilsynet, 2022b). Derfor var det viktig å ikke bare vise data i form av farge, men også i form av tekst og figurer for å senke systemkravet. Konsekvent vil dette medføre inkludering av flere potensielle brukere. Dette kravet er implementert for difficulty nivå på rutene på siden “Routes”. Valg av farger var bestemt basert på fargene fra trafikklys, slik at vi fikk et universelt fargesystem som var lett gjenkjennelig for brukeren. Fargene valgt er grønn, gul og rød, i tillegg er det implementert en korresponderende tekst: “Easy” for grønn, “Medium” for gul og “Hard” for rød. Denne teksten og fargen vises på cardviewet til ruten på “Routes” siden.

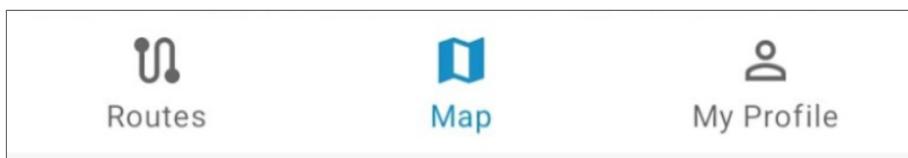


Figur 4.14 Bruk av farge på Routes siden

4.4.4 Konsekvent navigering og sidetitler

I WCAG-standarden legges det vekt på 3.2.3 konsekvent navigering, ved at det er gitt et nivå-AA krav som tilsvarer midterste nivå. Applikasjonens navigasjonsløsning er satt opp slik at den består av tre slides (Routes, Map og Profile), styrt av en navigasjonsbar nederst.

Navigasjonsbaren inneholder tre interaktive knapper med hvert sitt ikon og titteltekst. Ved å trykke på en av knappene blir brukeren sendt til den valgte siden. Brukeren vil få feedback fra systemet om nåværende posisjon i applikasjonen, ved at ikonet på navigasjonsbaren blir fargelagt med én blå farge og at de andre ikonene forblir ubelyst. Navigasjonsbaren er også lett tilgjengelig for brukeren og godt synlig på alle sidene. Denne navigasjonsløsningen vil forblie uendret selv etter at applikasjonen skrus av og på. Ikonene har i tillegg én titteltekst som representerer den tilhørende sidens emne. Dette står i WCAG-standarden som 2.4.2 Sidetitler.



Figur 4.15: Navigasjonsbar

5.0 Testdokumentasjon

En viktig del av systemutviklingsprosessen er testing av programvare. Testing har som hensikt å avdekke feil i programvaren samt demonstrere overfor kunden at appen møter kravspesifikasjonen (Sommerville, s.227, 2015). Videre skiller vi mellom enhetstesting og integrasjonstesting (Android Developer, 2022). Vi har basert oss på testpyramiden som foreslår at man har flest enhetstester, og færre integrasjonstester (Alfheim & Almås, 2022) .

5.1 Enhetstester

5.1.1 API-test

En viktig del av appen er å hente data fra API som skal presenteres til brukeren. Samtidig som vi utviklet koden satte vi opp enhetstester for å teste koden. Målet med testingen var å sikre at feil i henting fra API blir håndtert på en god måte (unntak håndteres). Data fra API parses fra JSON til dataklasser ved bruk av Ktor. Unntak kan kastes hvis vi ikke får lest inn dataen riktig, f.eks. hvis vi får en HTTP respons 4xx (feil på klientsiden) eller 5xx (feil på serversiden). I enhetstestene tester vi både valideringstester (forventet bruk) og defekttester (feil format f.eks.) (Sommerville, s.227, 2015). For enhetstesting av API bruker vi JUnit 4. Disse testene kjøres lokalt uten bruk av emulator.

```

    @Test
    fun loadAirQualityForecastWrongInput() {
        runBlocking { this: CoroutineScope
            val returnVal = source.loadAirQualityForecast(lat: "0,0", lon: "0,0")
            assert(returnVal == null)
        }
    }

    @Test
    fun loadAirQualityForecastWrongCountry() {
        // Should result to null after throwing exception
        runBlocking { this: CoroutineScope
            val returnVal = source.loadAirQualityForecast(lat: "35.652832", lon: "139.839478")
            assert(returnVal == null)
        }
    }

    @Test
    fun loadAirQualityForecastCorrectInput() {
        runBlocking { this: CoroutineScope
            val returnVal = source.loadAirQualityForecast(lat: "59.94410", lon: "10.7185")
            assert(returnVal is Pm10Concentration)
        }
    }
}

```

Figur 5.1: Enhetstester av API i klassen RepositoryTestAPI

Over er et eksempel av enhetstesting av loadAirQualityForecast. De to øverste tilfellene er defekttester, mens den nederste er valideringstest. I øverste tilfelle forventer vi en feil av type HTTP 4xx, siden det er feil format på HTTP-forespørselen. Den andre testen sjekker noe av det samme, men denne gangen er formatet korrekt, men airquality-API-et er ikke tilgjengelig i utlandet. Tester derfor med Tokyo for å sjekke nettopp dette. Den siste testen er en valideringstest der vi sjekker om riktig dataklasse opprettes.

5.1.2 Enhetstesting av ViewModel og Fragment

Siden API tar tid å laste inn, og kan være upålitelig som vist over, ønsket vi å isolere testingen. Prinsippet med enhetstesting er at det nettopp kun skal teste en funksjon (Android Developers, 2022). Sommerville foreslår å lage mock-varianter av objektene hvis man er avhengig av f.eks. en database (Sommerville, 2015, s.234). Vi har derfor laget Fake-varianter av klassene Repository og Datasource. Vi lar disse Fake-klassene implementere interface med metodene som Repository og Datasource har. Ved hjelp av design-patternet Dependency Injection kan vi teste fragment uten å bruke faktiske data fra API.

```

    @Before
    fun setUp() {
        datasource = FakeDataSource()
        repository = Repository(datasource)
    }

    @Test
    fun loadBySykkelRoutes() {
        runBlocking { this: CoroutineScope
            val returnVal = repository.loadBySykkelRoutes()
            assert(returnVal is List<Route>)
        }
    }
}

```

Figur 5.2: Bruk av `FakeDataSource` inne i en ekte repository

5.2 Integrasjonstester

Hele appen er avhengig av flere faktorer, og test av UI skjer gjennom integrasjonstesting. Test av UI kan utføres manuelt ved å bruke appen, men det er enkelt å overse aspekter samt at det eskalerer dårlig (Android Developer, 2022). Vi bruker Espresso for integrasjonstest av appen. Målet med testingen er å verifisere at navigasjonen i Profile-fragment er korrekt. Denne testingen krever en emulator.

```

@Test
fun testNavigation() {
    // Setup part: Create a TestNavController
    val navController = TestNavController(
        ApplicationProvider.getApplicationContext())

    val currentUser = FirebaseAuth.getInstance().currentUser

    // Create a graphical scenario for the ProfileFragment
    val scenario = launchFragmentInContainer<ProfileFragment>()

    scenario.onFragment { fragment ->
        // Set the graph on the TestNavController
        navController.setGraph(R.navigation.mobile_navigation)

        // Make the NavController available via the findNavController() APIs
        Navigation.setViewNavController(fragment.requireView(), navController)
    }

    // check if user is logged in or not
    if (currentUser == null) {
        // Calling:
        onView(withId(R.id.sign_in_sign_up_button)).perform(click())

        // Assertion part:
        assertEquals(navController.currentDestination?.id, R.id.signInFragment)
    }
}

```

Figur 5.3: Integrasjonstest av `ProfileFragment`

Vi følger prinsippet til Sommerville om å dele automatiserte tester opp i tre faser. En setup-fase, en kall-fase og en assertion-del (Sommerville, 2015, s. 233-234). Over tester vi om navControlleren vår åpner riktig fragment. Vi forventer at når vi trykker på sign_up_button blir vi sendt til signUpFragment.

6.0 Prosessdokumentasjon

6.1 Smidig utvikling

Vi har i denne prosjektoppgaven fokusert på smidig utvikling. Dette er en arbeidsteknikk som brukes i uforutsigbar systemutvikling, hvor de funksjonelle kravene kan forandres i løpet av prosjektet. Dette er i motsetning til tradisjonelle arbeidsteknikk, der man følger en fossefallsmodell. For å jobbe med de dynamiske omgivelsene, og ikke mot dem er metodikken i stor grad tilpasset disse kjennetegnene. Derfor kjennetegnes denne prosessen ofte av arbeid i korte og hyppige sprinter som er med på å dempe usikkerheten (Digitaliseringsdirektoratet, 2019). I disse sprintene fokuserer man på kontinuerlig læring og informasjonsinnhenting fra sluttbruker for å levere et fullverdig sluttprodukt. Viktige kjennetegn ved smidig metodikk finner vi i Agile Manifesto som viser til 12 prinsipper for smidig programvareutvikling (Sjøberg & Bergersen, 2022). Disse prinsippene har vært viktige for oss og skal reflekteres gjennom prosjektarbeidet vårt.

6.1.2 Scrumban

I begynnelsen av prosjektet ble vi enige om å arbeide med en smidig prosessmodell, kalt Scrumban. Scrumban er en hybrid mellom Scrum og Kanban (Ladas, 2021). Vi valgte å ta i bruk trekk fra scrumsprosess som arbeid i iterative sprintsykler som er tidsbegrenset. Disse sprintsyklene varte i eksakt en uke, men i enkelte tilfeller gikk vi vekk fra denne definerte avgrensningen, mer om dette i kapittel 6.3. Syklusene begynte med en planleggingsfase, der vi tok inn arbeidsoppgaver (brukerhistorier) i sprintene våre. Gruppen ansvarlig for design presenterte da disse oppgavene for oss andre, deretter evaluerte vi oppgavene etter parametere som fordeling, gjennomførbarhet og nytte. Dersom det forekom uenigheter i denne fasen tok vi en runde til med evaluering og forsøkte å løse dette gjennom diskusjon. På denne måten formet vi hver ukes sprint backlog. Etter å ha arbeidet med oppgavene møttes vi for å gå gjennom ukens sprint, og i tillegg planlegge for neste ukes sprint.

Dersom noen av oppgavene ikke ble utført ville de bli overført til neste sprint og lagt ekstra mye vekt på. På denne måten tok vi også i bruk fra kanban, der dette særlig var med på å få frem flaskehalsen i prosjektet. På grunn av vår smidige tilnærming til dette prosjektet var vi stadig åpne for forbedringer, dermed også endringer som vi viser til i kapitlene under.

På grunn av vi fulgte store deler av en scrum prosess delte vi også inn roller i henhold til dette. Vi hadde en scrum master på rullering som hadde ansvar for å koordinere arbeid samt at denne personen hadde ansvar for å lede stand-up møtene. Disse møtene ble brukt til å gå igjennom foregående ukes sprint og eventuelt hvorfor noen oppgaver ikke ble utført.

6.2 Arbeidsverktøy

I løpet av dette prosjektet har vi brukt en form for hjelpebidrifter i enhver fase av prosjektet. Særlig gjelder dette digitale plattformer som har gjort kommunikasjon og arbeid lettere, da det fjerner kravet om fysisk oppmøte. Vi vil i dette kapittelet redegjøre for hvilke hjelpebidrifter vi har brukt i de ulike fasene av prosjektet samt hvorfor disse ble tatt i bruk.

6.2.1 Arbeidsstruktur

Kommunikasjon er sentralt i ethvert teamarbeid, derfor ble det lagt stor vekt på å finne verktøy for å styrke dette. Til dette ble særlig digitale verktøy tatt i bruk, da kommunikasjon innad i gruppen foregikk over Facebook Messenger. Hvorimot kommunikasjon med veilederne våre foregikk over Microsoft Teams. Fysiske møter var vår foretrukne kommunikasjonsform, da en slik form er tydeligere og sjeldnere kan tolkes galt. Disse fysiske møtene ble også gjort digitalt noen et par ganger over Zoom, ved fravær av noen gruppemedlemmer. For å visualisere sprintene våre tok vi i begynnelsen av prosjektet i bruk Jira, men valgte å gå bort fra dette verktøyet etter et par sprints (sprint 3), da vi valgte å gå for Notion.

Tidlig i prosjektet lagde vi i en felles Google drive mappestruktur, der vi innad i denne hadde mapper for rapportskriving, møtereferater og analyse av brukerundersøkelser. Vi forsøkte å ha et strukturert oppsett så tidlig som mulig, slik at vi fikk god oversikt over dokumentasjonen vår samt at det var tidseffektivt. Denne plattformen ga oss store fordeler med hensyn på kollektiv skriving, deling av dokumenter og særlig kommentarfunksjonen i Google drive som gjorde det lettere å formidle og få oversikt over beskjeder i dokumentene.

6.2.2 Git

Git er et verktøy for versjonskontroll. Git gjør det enkelt å spore endringer i koden, og jobbe med flere versjoner samtidig (Bergersen, 2022). GitHub lar oss dele koden med hverandre slik at flere kan jobbe på samme prosjekt. Til tross for noen oppstartsproblemer og noen få problemer med merging underveis, har GitHub vært svært nyttig. Det har latt oss utvikle kode i inkrementer uten å være redd for å ødelegge noe samt mulighet for flere flere versjoner (branches).

6.2.3 Brukerundersøkelser og design

I arbeidet med den innledende brukerundersøkelser tok vi i bruk Nettskjema, mens UI-undersøkelsen ble utviklet gjennom Google Skjema. Disse verktøyene var godt tilpasset til utviklingen av undersøkelsene våre, i tillegg til de gjorde det enklere for oss med tanke på deling av undersøkelsen. Årsaken til at vi byttet fra Nettskjema til Google Skjema i UI-undersøkelse skyldes at lavere bildekvalitet ved bruk av Nettskjema, noe som vi tenkte kan være uheldig i UI-undersøkelse.

I utviklingen av prototypen vår tok vi i bruk flere hjelpeverktøy. Disse hjelpeverktøyene endret endret seg ettersom prototypen omfanget økte, da de ble mer kompliserte. I begynnelsene brukte vi Miro for vår lavtoppløselige prototype som er et verktøy for deling av konsepter og ideer over en digital tavle. Dette verktøyet fikk vi også nytte for i prosesser som; retrospektiv møter, diagrammer, UML osv. Gradvis i prototypeutviklingen gikk vi over til figma som er et bedre verktøy for utviklingen av høyoppløselige prototyper. Logoer og figurer for både prototypen og applikasjon ble funnet/laget ved hjelp av grafisk design verktøy; Miro og Adobe Illustrator.

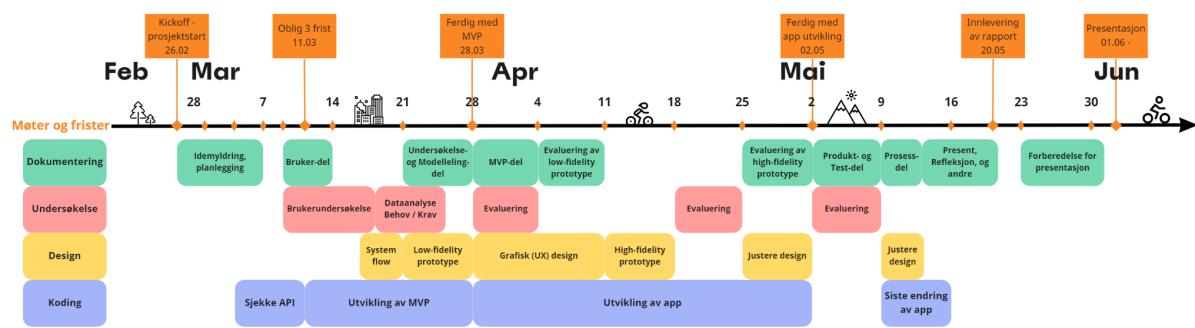
6.2 Planleggingsfase 26. februar- 14. mars

I begynnelsen av prosjektet var det viktig for gruppen å bli kjent med hverandre og utvikle god kjemi innad i gruppen. Vi delte styrker og svakheter angående de forskjellige arbeidsområdene: design, programmering, rapportskriving og samarbeid. En annen viktig aktivitet som ble utført i oppstartsfasen var utformingen av en teamavtale som satte rammene for videre arbeid i prosjektet (vedlegg 1). I tillegg var det også viktig å etablere mål og tanker rundt prosjektet og hva hver person kunne tenke seg å arbeide med. Oppstartsfasen var preget av en del diskusjoner med hensyn på valget av case. Etter tre møter og en del fram og tilbake

hvor tanker og ideer ble delt, endte vi med å velge case 1: Sykkelruter. Dette var et valg som alle på gruppen til slutt kunne enes om, selv om det var snakk om åpent case. Deretter satte vi igang brukerundersøkelser og intervjuer av målgruppen for applikasjonen.

Det ble tidlig bestemt at gruppen skulle møtes hver mandag kl 12:00, dette skulle erstatte den oppsatte forelesningstiden. I enkelte tilfeller forekom det også unntak, hvor vi spontant avtalte et ekstra møte i uken avhengig av behov. Uavhengig av antall møter i uken, holdt vi stadig kontakt gjennom messenger og fikk løst flere situasjoner via tekst. Innad i første møte ble vi enige om at hver sprint skulle være i 1 uke og skulle være fra mandag til neste mandag. Mandagsmøtene var både standout og planleggingsmøte. Møtene var delt opp i tre deler, hvor første del omhandlet hva vi hadde jobbet med i den gjennomførte sprinten, videre gikk vi gjennom agenda og til slutt diskuterte og fordele vi arbeidsoppgaver til neste sprint.

I tillegg til plan av møte, lagde vi en overordnet plan for hele prosjektet (*Figur 6.1*). Prosjektplanen inneholdt flere viktige datoer for ulike sprinter, dato for ferdigstilling av enkelte funksjoner og andre hindringer som for eksempel eksamener i andre fag). For hvert møte delte vi også ut rollene: møteleder og møtereferent. Møteleder skulle lede møte fra start til slutt og bestemme agenda for dagen. Møtereferent skulle skrive referat for møtet med vekt på diskusjoner og valg gruppen tok den dagen. Disse rollene var på rullering slik at det aldri var samme person med samme rolle i hvert møte. Vi ønsket å ha rullering for at alle skulle få et eierskap til prosjektet å hindre at teammedlemmer faller ut (Lindsjørn, 2022).



Figur 6.1 “Plan og oversikt”

Etter et par møter oppdaget vi at det var behov for retrospektive møter hvor vi fikk delt tanker om hvordan arbeidet har foregått. Vi gjorde retrospektiv på Miro hvor alle deltok og delte meninger. Deretter gikk vi gjennom hva hver av oss skrev ned og diskuterte positive og negative sider ved arbeidet rundt prosjektet.

Planleggingen av appen bestod av en del diskusjoner, men vi fikk også stort utbytte av å visualisere applikasjonen. I begynnelsen var skissene generelle og overordnede, men etterhvert ble de mer detaljerte. Ved eventuelle misforståelser ble skisser tegnet eller fremvist for å forklare enda nøyere visjonen bak en idé. Hensikten til skissene var å ha et felles “mål” å sikte etter, konsekvent førte dette til at kodingen ble lettere.

6.2.1 Arbeidsdeling

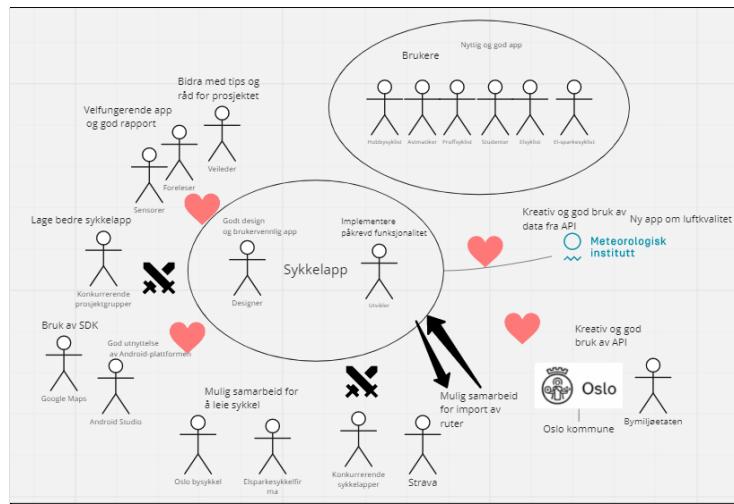
Som nevnt ovenfor jobbet vi i grupper. Vi hadde en klar arbeidsdeling, der gruppen ble delt forholdsvis 4 til 2 på koding og design. Hensikten med en slik fordeling var å fordele hovedansvar til medlemmene på en slik måte at vi kunne dra nytte av hverandres styrke. Uavhengig av hvilken gruppe man var på, var vi klare på at alle skulle ha overordnet kjennskap til andres arbeid. Dette mener vi å ha fått til gjennom stand-up møtene, der vi alle delte våre arbeid. I tillegg var vi også for at alle i gruppen hadde muligheten til å involvere seg i andres arbeidsoppgaver selv om det ikke var deres ansvarsområde.

6.3 Kravhåndteringsprosess

I dette kapittelet vil vi gjøre rede for prosjektets kravspesifikasjoner og modellering, men før det vil vi gjennomgå kravhåndteringsprosessen vi foretok i forkant. Her tok vi utgangspunkt i lærebokens standardiserte kravhåndteringsprosessen som består av fire faser; forstudiefasen, kravinnsamling og -analyse, validering av kravspec og håndtering av kravendring (Sjøberg, 2022, lysark 44). Håndtering av kravendring vil bli gjort rede for i kapittel 6.5. På grunn av prosessens omfattende form tok vi kun i bruk de trekkene vi mente ville komme mest til nytte for prosjektet vårt.

6.3.1 Forstudiefasen

I startfasen ønsket vi å få en oversikt over systemet (applikasjonen) vårt, prosjektets målsetting og hvordan vi skal nå denne målsetningen. Etter å ha definert målsettingen fra kapittel 1.3 begynte vi med planlegging av konkrete tiltak for å nå denne målsetningen. Vi begynte først med å foreta oss en analyse av systemets interesser, dette ble gjort gjennom å lage et rikt bilde, figur 6.2. Interesser som vi kartla på dette stadiet var Oslo Bysykkel, Strava og Elsparkesykkelfirmaer. Med denne informasjonen visste vi hvilke interesser vi var nødt til å rette oss mot for innhenting av informasjon knyttet til systemet.



Figur 6.2: Rikt bilde

6.3.2 Kravinnsamling og -analyse

Etter å ha fått en oversikt over systemets ulike interessenter var vi klare til å identifisere deres krav. Vi valgte her å kun fokusere på kravene til brukerne og utviklerne (våre), fordi vi mente dette var mest relevant for å oppnå målet vårt. For å identifisere disse kravene hos brukerne bestemte vi oss for å utføre en brukerundersøkelse. Denne typen informasjonsinnhenting var vår primære kilde for utforming av krav til brukere. Intervjuene og brukerundersøkelsens utforming og analyse vil vi redegjøre for i avsnittet under.

Kravene våre ble utformet med bakgrunn i kunnskap og erfaring vi har opparbeidet oss gjennom hele studiet. Her måtte vi også være forsiktige slik at brukerens krav ikke kom i konflikt med utviklerens krav. Dette omhandlet ofte krav vi trodde skulle føre til vanskeligheter under implementering. Vi som utviklere var også nødt til å analysere brukernes ønsker og behov for og deretter bestemme hvorvidt implementasjon av disse kravene skal prioriteres. For dette tok vi i bruk prioriteringskalaen “kan”, “bør” og “må” for å skille mellom kravene, dette er en videreføring av “nice to have” og “need to have” (Bergersen & Sjøberg, 2022). Der prioriteringen “må” betyr at kravet er essensielt for å løse oppgavens mål, mens “bør” er mindre viktig og “kan” er noe som er fint å ha med, men ikke avgjørende for brukeren.

I beskrivelsen av kravene har vi brukt tekstlige beskrivelser som brukerhistorier, samt diagrammer som use-case-, sekvens- og klassediagram fra UML (Unified Modeling Language). Kravene er dynamiske noe som skyldes at interessentenes brukerbehov stadig er i

endring. Dette er noe vi har forsøkt å tilpasse modelleringen vår etter, for å unngå endringer i etertid.

Brukerundersøkelse

Vi gjennomførte en spørreundersøkelse for å finne brukerens sykkelturer, hva som motiverer eller hindrer dem fra å sykle, og hvordan miljøbevissthet påvirker sykkelturer og motivasjon til å sykle. Vi lagde en spørreundersøkelse med UiO nettskjema, og la spørreundersøkelsen ut på forskjellige universitets- relaterte grupper på nettet.

I brukerundersøkelsen spurte vi deltakerne hvor mye de bryr seg om miljø på en skala fra 1 (de bryr seg lite om miljø) til 5 (de bryr seg mye om miljø). De som svarte 1 til 3 definerer vi som ikke-miljøentusiaster mens de som svarte 4 og 5 var definert som miljøentusiaster. I forhold til sykkelturer hadde vi også en skala fra 1 (sykler nesten hver dag) til 5 (sykler aldri). Vi definerte dem som svarte 1 til 3 som aktiv syklist, mens de som svarte 4 og 5 ble definert som ikke-aktiv syklist. Under dataanalysen valgte vi å dele deltakerne i fire forskjellige grupper basert på deres sykkelturer og miljøbevissthet.

	Aktiv syklist (De som svarte 1-3 på spørsmål om sykkelturen)	Ikke-aktiv syklist (De som svarte 4-5 på spørsmål om sykkelturen)
Miljøentusiaster (De som svarte 4-5 på spørsmål om miljø)	Gruppe 1	Gruppe 2
Ikke-miljøentusiaster (De som svarte 1-3 på spørsmål om miljø)	Gruppe 3	Gruppe 4

Tabell 6.1: 4 varianter av folk basert på sykkelturer og miljøhensyn

Vi sammenlignet først gruppe 1, 2 (miljøentusiaster) og gruppe 3, 4 (ikke-miljøentusiaster) for å finne miljøentusiasters tanker og behov rundt sykling. Dataene viser at det var en tydelig forskjell i demotivasjonsfaktorer mellom miljøentusiaster og ikke-miljøentusiaster. De fire hovedfaktorene som skilte miljøentusiaster og miljøentusiaster mest var **dårlig luftkvalitet** (7.1% vs 11.6%), **stor trafikk** (14.3% vs 25.6%), **dårlige ruteforhold** (14.3% vs 23.3%) og **manglende fasiliteter** (7.1% vs 21.6%). I tillegg viste dataene en annen interessant forskjell mellom miljøentusiaster og ikke-miljøentusiaster angående **gamification**. Gamification betyr å bruke spillmekanikk for å øke brukerengasjementet. 62.8% av miljøentusiaster i undersøkelsen svarte at de blir motivert av gamification, mens kun 42.9% av

ikke-miljøentusiaster er for gamification. Dette viser at miljøentusiaster har mer lyst på gamification enn ikke-miljøentusiaster.

Vi sammenlignet deretter gruppe 1 (miljøentusiaster som sykler aktivt) og gruppe 3 (ikke-miljøentusiaster som sykler aktivt) for å finne hva som kan motivere miljøentusiaster til å sykle. Dataene viser at gruppe 1 har høyere tall enn gruppe 3 i fritid (50% vs 40%), helse og trening (45.5% vs 40%), og å lindre stress (27.3% vs 20%), som deres motivasjon for å sykle, mens gruppe 3 hadde høyere tall enn gruppe 1 i alle andre faktorer, slik som å pendle, å spare penger og så videre. De tre kategoriene; fritid, helse og trening, og å lindre stress; er såkalt “**hobby-sykling**”. Derfor kan vi si at miljøentusiaster som sykler aktivt, sykler for fornøyelse oftere enn ikke-miljøentusiaster som sykler aktivt.

Til slutt sammenlignet vi gruppe 1 (miljøentusiaster som sykler aktivt) og gruppe 2 (miljøentusiaster som sykler lite) for å finne ut hva som hindrer miljøentusiaster fra å sykle. Dataene viser at det var svært stor forskjell mellom gruppe 1 og gruppe 2 i **helsebekymring** (9.1% vs 28.6%) og **stor trafikk** (13.6% vs 38.1%) som demotivasjonsfaktorer for sykling. Dette kan bety at de største grunnene som miljøentusiaster ikke velger å sykle var deres helsebekymring og trafikkbekymring. I tillegg scoret faktorer som **vær, temperatur og tid/avstand** like høyt blant gruppe 1 og gruppe 2 hvor begge gruppene var miljøentusiaster.

Ut i fra dataanalysen kom vi frem til 9 behov hos de som er miljøentusiaster og sykler aktivt.

1. Brukere vil sykle på ruter som har **bedre luftkvalitet**
2. Brukere vil sykle på ruter med **mindre trafikk**
3. Brukere vil sykle på **ruter med bedre tilstand**
4. Brukere vil vite om **sykkelfasiliteter** langs sykkelrutene
5. Brukere blir motivert til å sykle med **gamification**
6. Brukere vil heller sykle som **hobby** enn som tung trening eller reisemiddel
7. Brukere vil vite **luftkvalitet** på sykkelrutene som kan påvirke deres helse
8. Brukere vil vite **vær og temperatur** før de begynner å sykle
9. Brukere vil vite beregnet **tid eller avstand** på rutene

6.3.3 Validering av kravspesifikasjoner

Etter å ha utført spørreundersøkelsen og analyse av den var vi nødt til å sjekke om kravene de faktisk er det de ønsker. I evaluering av disse kravene så vi på om kravene oppfylte noen punkter; nødvendige (oppfyller målet til appen?), realistiske (har vi kompetansen/tid til å løse

oppgaven?) og testbare (kan man teste om kravet er oppfylt?) (Maus, 2010). Basert på disse kriteriene valgte vi ut kravene våre som er vist til i kapittelet for kravspesifikasjon og modellering.

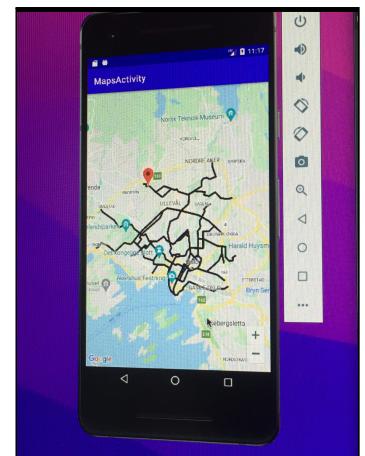
6.4 Sprintene

Sprint 1 - 14. - 21. mars

Den første sprinten jobbet vi med brukerundersøkelse og startet på utvikling av appen. Denne sprinten var gruppen svært engasjerte og ivrige og vi fikk gjort mye på en uke.

Vi bestemte oss for målgruppe i fellesskap. Deretter jobbet “design-team” videre med planlegging og utførelse (sende den ut) av spørreundersøkelse på den valgte målgruppen. Mens den “tekniske gruppen” jobbet med å vise et kart med sykkelruter over Oslo, hente værdata fra MET-API og utvikle applikasjonens MVVM-arkitektur. I tillegg satte vi opp Github og lærte oss grunnleggende GitHub ved å pushe og pulle kode.

Selv om vi fikk gjort mange oppgaver, var det flere utfordringer vi hadde denne sprinten. Vi manglet en felles forståelse for valg av målgruppe og hadde en for dårlig kommunikasjon. I tillegg hadde vi noen tekniske problemer i å få delt kode, da ikke alle medlemmene hadde brukt GitHub før, og dette måtte settes opp i Android Studio.

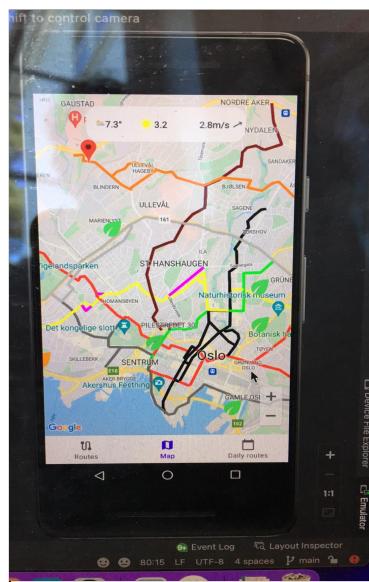


Sprint 2 - 21.-28. mars

Denne sprinten jobbet vi videre med utvikling av appen og analyserte spørreundersøkelsen. Designgruppen gikk videre med å analysere resultatene fra brukerundersøkelsen og definere brukerbehov. Ut ifra analysen ble det også laget flere brukerhistorier, og i tillegg rangert krav etter ønske og behov. I tillegg lagde vi en low fidelity prototype av applikasjonen samt utarbeidet persona. Programmeringsgruppen jobbet sammen ved parprogrammering. To og to satt sammen på hver sin oppgave. Opgavene vi jobbet med var å visualisere værdata, implementere nåværende lokasjon, og visualisere luftkvalitet på kartet. I tillegg fargela vi de ulike rutene fra Oslo kommune sitt API for å få oversikt over rutene. Når det gjaldt rapportskrivning jobbet vi videre med kapittel 2 og kapittel 3, der vi skrev om brukerundersøkelsen og krav.

Når vi så nærmere på kartet over sykkelrutene oppdaget vi at rutene ikke var det vi hadde sett for oss. Gruppen forventet at Oslo kommune sitt API skulle tilby en rekke turer med start og sluttspunkt som vi deretter kunne foreslå til brukere. Imidlertid ga API-et oss kun en oversikt over skiltede sykkelveier i byen. Vi sorterte rutene etter “ruteId” og viste dem på kartet (*Figur 6.3*). Turene gikk litt overalt og hadde ikke noe klart start- og sluttspunkt. I tillegg hadde vi noen rosa ruter med verdien “null” som ikke ga mye mening (*Figur 6.4*). Vi tok kontakt med bymiljøetaten til Oslo kommune og fikk informasjon om at API-et var et gammelt datasett som ikke inneholdt alle skiltede sykkelveier i Oslo. De sa at vi skulle se bort fra rute “null” fordi det er ruter som ikke eksisterer.

Slik vi så det stod vi mellom tre valg: bruke Strava sitt API som viser sykkelruter, lage vårt eget API, eller gjøre oppgaven om til et åpent case. På dette tidspunktet visste vi ikke om Bysykkel-API-et med ruter. Vi var veldig usikre på hva vi skulle gjøre, og flere av teammedlemmene hadde sterke meninger om de ulike alternativer. Det var steile fronter. Det ble heldigvis ingen konflikter, da vi var enige om at dette kun var saklige diskusjoner, og ikke noe som måtte bli tatt personlig. Siden dette var et viktig veivalg, og vi ikke kom til noen felles løsning valgte vi å utsette avgjørelsen midlertidig. Vi jobbet heller med andre oppgaver som måtte gjøres.



Figur 6.3: “Sortert etter id”



Figur 6.4: “Rute null”

Siden mye hadde skjedd, og det var en stund siden forrige retrospective hadde vi en ny retrospective for å evaluere oss selv. Vi kom fram til at vi hadde god arbeidsfordeling, kommet godt i gang med både rapport, brukerundersøkelser og app. Faktorer som vi synes at

holdt oss tilbake var at Jira-boardet som vi hadde brukt ble svært kaotisk og at det var vanskelig å holde oversikt. Det var heller ikke mulig å tilordne en oppgave flere personer, som derfor ikke gjenspeilet slik vi hadde jobbet. Vi hadde blant annet hatt parprogrammering. Vi hadde ikke hatt noen grense på WIP (work in progress), selv om Kanban anbefaler dette (Kniberg & Skarin, 2010, s.15). Som gruppe bestemte vi oss for å ha en begrensning på 6 oppgaver “in progress” for å unngå å miste oversikten til senere. I tillegg byttet vi verktøy fra Jira til notion for å kunne tilegne en oppgave til hver person.

Sprint 3 - 28. mars- 4. april

Den tredje sprinten implementerte vi flere funksjoner i applikasjonen, lagde høyoppløselig prototype (se vedlegg 6) og utformet intervjuguide. Design-gruppen forberedte seg til evaluering av høyoppløselig prototype. Fra spørreundersøkelsen fikk vi vite at brukerne ønsket en oversikt over sykkelparkeringer. Programmeringsgruppen jobbet derfor med å visualisere sykkelparkeringer og bisykkelstasjoner med status over kapasitet. Etter å ha fått dette inn i applikasjonen forstod vi fort at disse markørene måtte bearbeides for å øke lesbarheten og brukervennligheten. Vi bestemte oss da for å innføre “clusters” for markørene. I tillegg ønsket vi å gi brukeren mulighet for å deaktivere ikonene på kartet ved bruk av checkboxes. Fra veilederne fikk vi innspill på at vi burde lage en funksjon for å finne gjennomsnittlig luftkvalitet basert på lokasjoner på en rute. Av den grunn undersøkte vi MET sitt API: AirQualityForecast. Det var imidlertid ikke helt rett frem hvordan vi skulle implementere dette, og denne oppgaven ble utsatt til neste sprint. Til slutt jobbet vi også med å få inn brukerlokasjon, og koble den opp imot API for værdata, slik at vi fikk værdata fra brukerens posisjon. I denne sprinten oppdaget vi også Oslo bisykkel sitt API med historiske data, og vi diskuterte om dette var noe som var aktuelt å bruke i appen vår.

Sprint 4 - 4. april - 19. april

Denne sprinten hadde vi brukbarhetstesting og analyse av undersøkelsen samt enhetstesting av API-kall og koding av profil-delen av appen. Vi forlenget sprinten til to uker, på grunn av arbeid med andre emner og påskeferie. Dette betød at et av møtene utgikk, men vi hadde fortsatt et stand-up-meeting i denne perioden for å holde oss oppdatert på fremgangen. Tidlig i sprinten besluttet vi å implementere Oslo Bisykkel sitt API med historiske data. Vi hentet inn data fra API-et og opprettet layout for siden med anbefalte ruter. Her ville vi enkelt teste at vi fikk til å hente data fra API-et og vi lagde et recyclerview med tilhørende adapter som viste rutene fra API-et. På dette tidspunktet viste vi alle rutene som var syklet (over 5000). I

tillegg implementerte vi en funksjon som henter bilde fra Google basert på navn på startpunktet til ruten. Denne sprinten følte vi gikk litt tregere enn de andre på grunn av påskeferien, men vi opprettholdt fortsatt en god prosjeksjon.

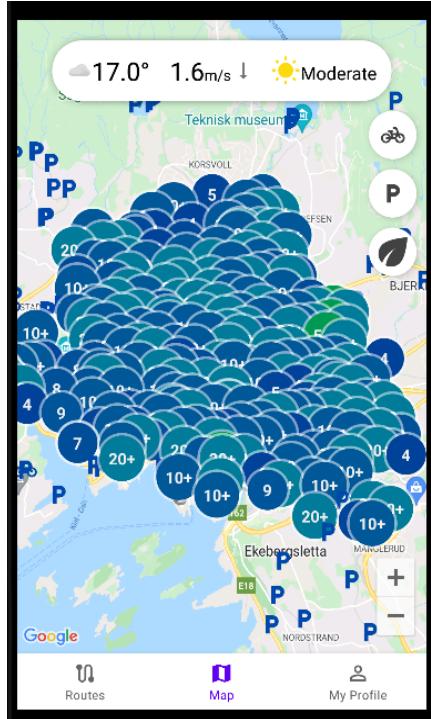
Sprint 5 - 19. april - 10. mai

Den femte sprinten gjennomførte vi UI-undersøkelse av ikoner til appen (kap 5.5.2), analyse av undersøkelsen, bestemte navn og logo til appen, undersøkte hvorvidt appen er universelt utformet, la til mer funksjonalitet i appen og fjernet bugs. Grunnet eksamener i andre emner bestemte gruppen seg for å utvide denne sprinten til en 3 ukers sprint. I denne sprinten bestemte vi oss for å gå helt over til kanban, da vi anså dette som bedre egnet for denne perioden. Dette skyldtes at vi hadde noen få oppgaver igjen som hindret flyten i arbeidsprosessen som vi mente måtte bli gjort for å bli kvitt disse flaskehalsene. En annen grunn til at vi forlenget denne sprintene var at disse oppgavene ikke hadde estimert tid, noe vi gjorde i henhold til kanban utvikling.

På kodefronten handlet det om å knytte profilen til en database, koble til profilen med resten av appen og sortere recyclerviewet etter ulike parametere (lengde, luftkvalitet og vanskelighetsgrad). I tillegg gjorde vi cardviews-ene klikkbare slik at den åpnet et kart over ruten. Vi la dessuten til funksjonalitet for å endre personlige opplysninger og bilde på profil-delen av appen.

Videre fjernet vi enkelte bugs vi hadde i appen, blant annet fikk vi ikke knappen som skulle fjerne clusters (parkering og bisykler) til å fungere korrekt. De var bare halvveis forsvunnet, og kunne dukke opp igjen hvis vi scrolltet til siden. Et annet problem var at parkering og bisyklene brukte hver sin cluster-manager. De samarbeidet derfor ikke, og det ble uoversiktig for brukeren (se figur 6.5).

Til slutt lagde vi en introskjerm som møter brukeren når de åpner appen og gir dem en kort introduksjon til hva appen kan brukes til.



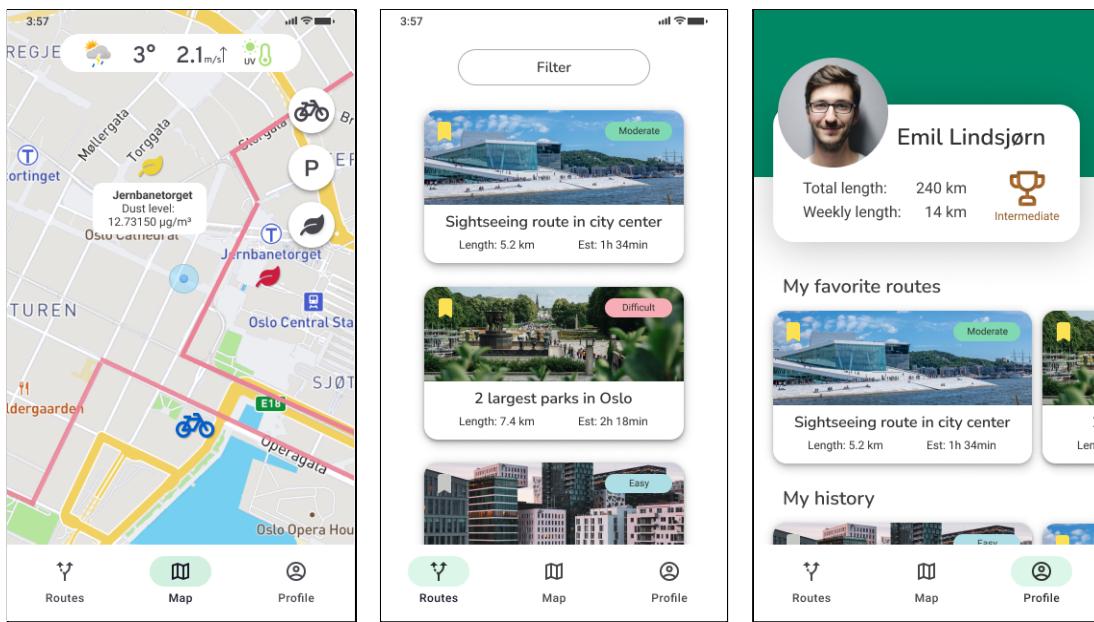
Figur 6.5: "Utkast av Parking og bysykler"

6.5 Evaluering og håndtering av kravendring

Evaluering med sluttbrukere spilte en viktig rolle i vårt prosjekt ved å la oss forbedre appen kontinuerlig. Vi gjennomførte 2 typer evaluatingsprosesser; en kombinasjon av intervju og brukbarhetstesting om funksjoner til appen, og deretter en spørreundersøkelse på design. Som nevnt er brukernes krav stadig i forandring noe som også reflekteres gjennom undersøkelsene utført ovenfor. Ved å treffe brukergruppen vår igjen fikk vi verdifull informasjon som var med på endre, fjerne eller gi innsikt om nye krav.

6.5.1 Brukbarhetstesting og intervju

Vi lagde en høyoppløselig prototype med hjelp av Figma (vedlegg 6), samtidig som vi begynte å programmere MVP. Prototypen vår var interaktiv og nesten alle knapper fungerer som en enkel versjon av en ekte app. Dette hjalp oss til å sjekke om de planlagte funksjonene oppfattes på riktig måte, hvordan vi kan forbedre appen vår, og å justere dem i forkant av å programmere den ekte appen.

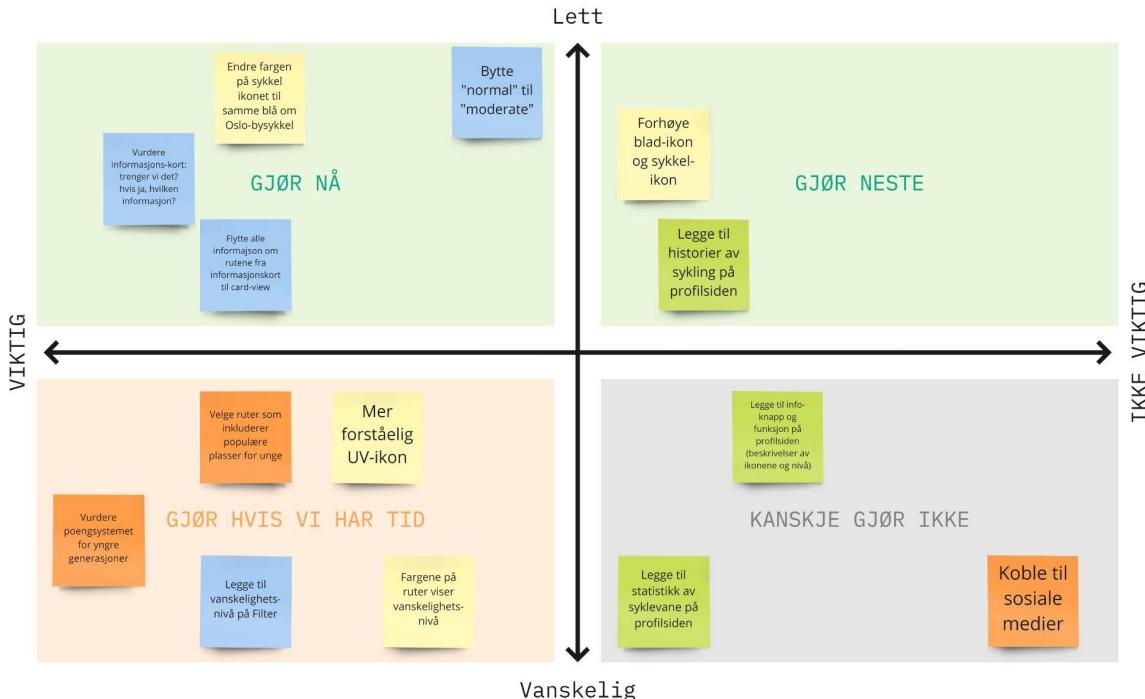


Figur 6.6 : Høyoppløselig prototype

Vi valgte å gjennomføre en kombinasjon av brukbarhetstesting og intervju om prototypen fordi vi ville teste funksjoner samtidig som vi ville få nye innspill og ideer fra sluttbrukere. Vi lagde en intervjuguide og et samtykkeskjema (vedlegg 4 og 5) fordi brukbarhetstesting og intervjuet ble tatt opp for å dele det med andre gruppemedlemmer som ikke var tilstede og for videre dataanalyse. Undersøkelsesobjektet faller innenfor vår målgruppe (universitetsstudent under 30 år, som sykler mer enn 2 ganger i måned).

I brukbarhetstestingen og intervjuet stilte vi først flere generelle spørsmål om sykkelmaner til undersøkelsesobjektet, og deretter beskrev vi generelle funksjoner av appen som vi tenkte på, ved å vise den interaktive prototypen. Vi spurte undersøkelsesobjektet om å bruke prototypen fritt, og deretter stilte vi flere spørsmål angående funksjoner i appen og intervjuobjektets opplevelser om bruk av appen.

Som dataanalyse transkriberte vi brukbarhetstesting/intervju opptak, og deretter lagde vi et sammendrag av dataene. De nylig funnet brukerbehovene og mulige endringer av appen ble kategorisert i en matrise: viktig - ikke viktig og lett å implementere - vanskelig å implementere.



Figur 6.7: kartlegging av nye brukerbehov og mulige endringer etter evaluering

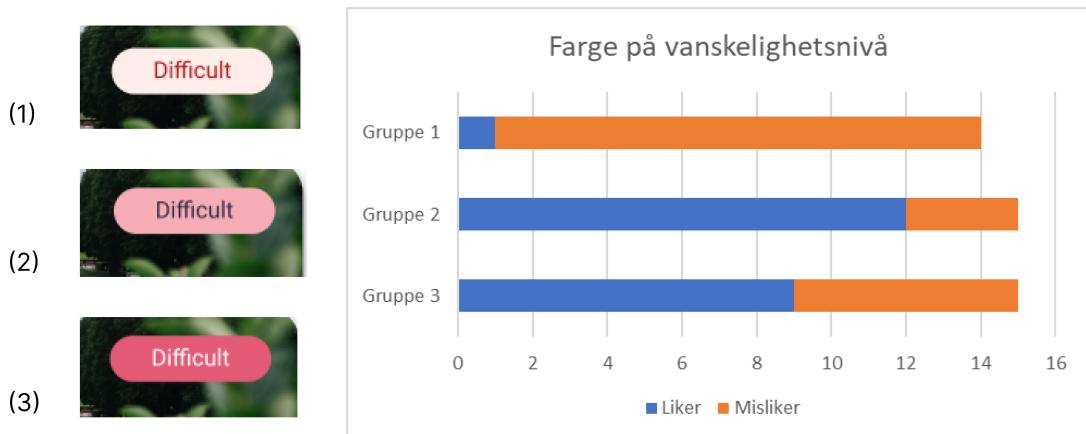
Basert på behovene som ble kartlagt, gjorde vi flere endringer fra små til store beslutninger. De viktigste endringene er følgende:

- 1. Informasjonsside med detaljer slettes, og card-view inneholder all viktig informasjon i stedet.** Ved å gjøre dette, kan brukere vite hvordan hver rute er uten å trykke på card-view og å åpne informasjonsside.
- 2. Bytte poengsystem til total lengde.** Selv om den innledende undersøkelsen viste at målgruppen vår blir motivert av gamification, fikk vi tilbakemelding at poengsystem ser ut som for eldre, mens målgruppen vår er unge. Derfor bestemte vi å implementere total lengde som en type gamification, i stedet for poengsystem.
- 3. Appen anbefaler mest populære ruter som live data.** Dette gjorde det mulig å tilpasse appen til unge generasjoner som er vår målgruppe.
- 4. Alle ruter på kart får samme farge.** Vi fikk tilbakemelding at det så ut som at forskjellige farger på rutene betydde noe, slik som vanskelighetsnivå av rutene. For å unngå forvirring, bestemte vi å sette samme farge på alle ruter på kart.

6.5.2 Evaluering av visuelt design

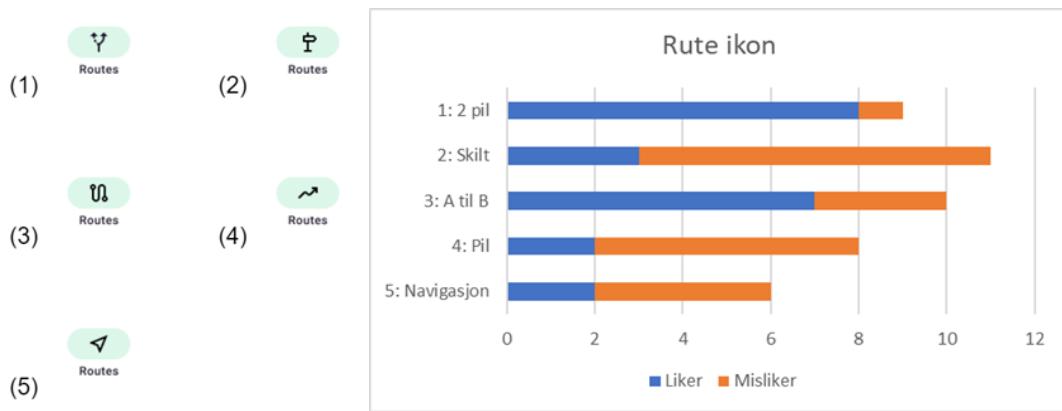
Vi gjennomførte en spørreundersøkelse om det visuelle designet, fordi en av tilbakemeldingene vi fikk under brukbarhetstesting/intervjuet var at UI (user interface) av appen ikke passer for målgruppen vår. Vi lagde et skjema med bilder av forskjellige farger og ikoner. Den opprinnelige planen var å gjennomføre spørreundersøkelsen i form av gateintervju på universitetet, men vi erfarte at det var veldig vanskelig å finne noen som vil gjerne delta på undersøkelsen. Til slutt endret vi planen og sendte skjemaet personlig til dem som falt inn under målgruppen vår, og la den ut på forskjellige universitets-relaterte grupper på nettet også. Deltakerne fikk 6 spørsmål angående farger og ikoner, og de ble spurta hvilke design de liker best og minst.

Resultatet av undersøkelsen viste at det ikke var stor forskjell i preferanser mellom designeksempler med stor, mindre og minst fargekontrast. Det var imidlertid veldig få folk som svarte at de misliket designeksempelet med stor fagekontrast, som fikk oss til å gå for stor fagekontrast i appen vår. I tillegg hadde vi også spørsmål om farger på vanskelighetsnivå på sykkelrutene. Som resultat fant vi ut at fargen bakgrunn med svart tekst var mer populært enn andre.



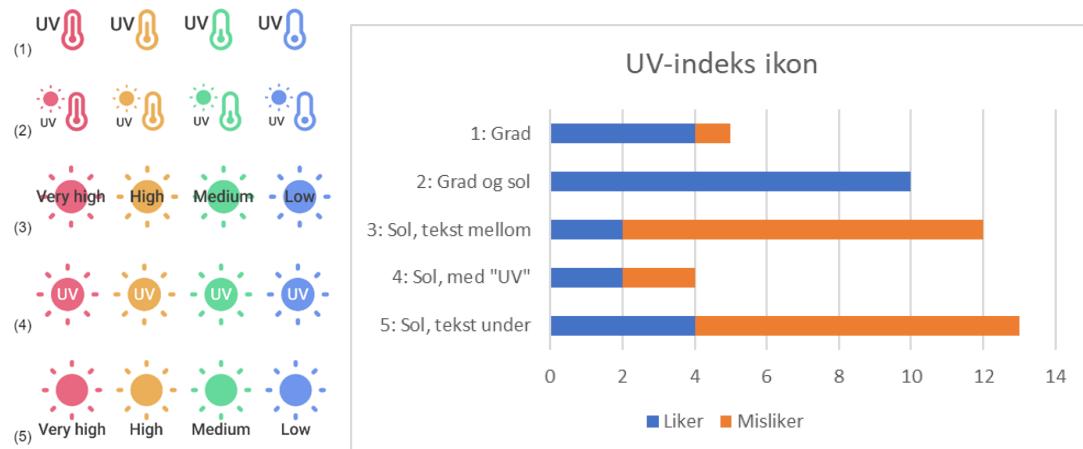
Figur 6.8: resultat av farger på vanskelighetsnivå

Angående ikoner stilte vi spørsmål om rute-, kart- luftkvalitet- og UV-ikoner. Når det gjelder ruteikon, viser resultatet at «2 piler» og «A til B» figur er mest populære blant målgruppen vår. Vi valgte «A til B» figur som rute ikon fordi det passer fint for rutene som appen viser, hvor det er A-punkt og B-punkt, ikke en rute som splitter seg.



Figur 6.9: resultat av rute ikon

Vi lagde 5 variasjoner av UV-indeks ikonet ved å kombinere solfigur, grad-figur, tekst «UV», og tekstbeskrivelse av UV-nivå. Gjennom dataanalysen fant vi ut at målgruppen vår ikke likte ikoner med tekstbeskrivelse av UV-nivået. Det mest populære ikonet var en kombinasjon av solfigur, grad-figur og teksten «UV».



Figur 6.10: resultat av UV-indeks ikon

6.5.3 Utførte krav

I tabellen under finner er det en oversikt over hvilke funksjonelle krav som har blitt implementert og hvilke som ikke har. Krav med delvis implementasjon betyr at det er satt et slags rammeverk for implementasjonen, men funksjonen er ikke ferdigstilt. Årsaker til at noen av kravene ikke er implementert har å gjøre med evalueringen vår, der kravene ikke tilfredsstilte kriteriene; nødvendighet, realistisk og testbarhet.

Funksjonelle krav - Implementasjon

Funksjonelle krav	Implementasjon
I det brukeren åpner applikasjonen vil de få en kort introduksjon til applikasjonens funksjonalitet	Implementert
Vise brukerposisjon på kartet	Implementert
Applikasjonen skal vise kart over alle de skiltede sykkelrutene i Oslo	Implementert
Applikasjonen skal ha en funksjon på kartet som viser tilstanden til luftkvaliteten i Oslo	Implementert
Applikasjonen skal vise sanntidstemperatur der hvor brukeren befinner seg	Implementert
Applikasjonen skal vise vindstyrke og vindretning i sanntid der hvor brukeren befinner seg	Implementert
Vise nivåer for UV-stråling i sanntid fra brukerens posisjon	Implementert
Applikasjonen skal gi brukeren forslag på stier med hensyn til faktorer som rutelengde, luftkvalitet og vanskelighetsgrad	Implementert
Gi brukeren mulighet til å legge til en sti under favoritter	Delvis implementert
Applikasjonen skal gi brukeren mulighet til å se hvilke veier som er trafikkerte	Ikke implementert
Vise gjennomsnittlig stigning for hele sykkelruten	Ikke implementert
Bruker skal kunne lage egne ruter og lagre dem	Ikke implementert
Brukeren skal kunne markere ruter som de har kjørt.	Ikke implementert
Applikasjonen skal fungere for horisontal og vertikal skjerm.	Ikke implementert
Applikasjonen skal fungere i dark mode	Ikke implementert

Tabell 6.2: Implementerte funksjonelle krav

Ikke-funksjonelle krav - Implementasjon

Blant de ikke-funksjonelle kravene var krav vi ikke klarte å oppfylle; applikasjonen skal være universelt utformet (WCAG 2.1), funksjoner/innhold skal lastes inn under 2 sekunder og layout skal være dynamisk og responsiv. Årsaker til at dette ikke ble oppfylt er høye krav for universell utforming (WCAG 2.1) som nevnt i kapittel 4.4, mens vi kommer mer inn på

vanskighetene ved innlasting i kapittel 8.2. Hvorimot det siste kravet ikke ble oppfylt på grunn av begrenset tid, og ble derfor ikke ble prioritert.

6.6 Avsluttende fase - 10. mai - 20. mai

Hovedsakelig lå fokuset på koding og design i sprint intervallet 1-5. Til tross for dette, hadde vi alltid rapportskriving i bakhodet og var opptatt av å skrive om gjennomførte prosesser fortløpende. De siste ti dagen før innleveringsfristen 20.mai, skiftet fokuset fra applikasjonen og koding til rapportskriving. Imidlertid greide vi ikke helt å slutte å jobbe med appen da vi fortsatt hadde ting vi ønsket å forbedre appen vår. Blant annet ønsket vi å kunne oppdatere profilbilde og vi måtte sørge for at intro-skjermen fungerte som den skulle. Når vi skrev på rapporten viste det seg at notatene våre var nyttige, med tanke på at det var en del punkter i rapporten som kunne fylles inn. Arbeidsfordelingen rundt rapportskriving var friere og det var valgfritt hvor enhver person ville skrive.

7.0 Refleksjon

Når vi ser tilbake på prosjektet burde vi fulgt “Just In Time (JIT)” enda bedre. JIT er en styringsmetode hvor oppgaver blir satt opp steg for steg for å unngå en uforsvarlig prosess og dobbeltarbeid i framtida. Det handler om å ikke lage noe før det er etterspurt (Lindsjørn, 2022). Siden vi begynte å kode før brukerundersøkelse og spesifikasjon av brukerbehov ble ferdig, ble det et dobbelt arbeid med koding og planlegging av app design/funksjon, og vi måtte tilpasse appens funksjoner for brukerbehov etter at vi begynte å programmere.

Samtidig måtte vi ferdiggjøre alle prosessen innen 3 måneder, som ikke var lenge nok til å vente frem til brukerundersøkelsen ble ferdig. Vi tror at programmeringsteamet hadde bra forståelse for viktighet av undersøkelse og brukerbehov, og var åpne for endringer hvis nødvendig. Vi kommuniserte og diskuterte godt angående hva som burde bli prioritert og bli endret.

Prosjektplanen vi lagde i begynnelsen (figur 6.1) synes vi at vi stort sett har fulgt. Imidlertid fokuserte vi for mye på apputvikling, og som resultat utsatte vi rapportskriving til slutten av prosjektet. Det som hjalp oss med rapportskrivingen til slutt var møtereferater som vi skrev etter hver eneste møte. I tillegg hadde vi hatt flere retrospekt møter, og en refleksjonssessjon sammen med alle gruppemedlemmer ved hjelp av en refleksjonstabell på Miro.

Angående pandemi var samfunnet og universitetet allerede ganske åpent, så det har ikke påvirket gruppearbeidet i så stor grad. Vi har likevel brukt verktøyene som pandemien har innført. Når gruppemedlemmer var syke kunne vi ha hybride møter - både fysisk og digitalt - ved å bruke Zoom. Vår tilvenning til hybride møter førte også til fleksibilitet i andre situasjoner slik som andre planer eller jobb utenfor gruppeprosjektet.

8.0 Videre arbeid

Gruppen hadde store ambisjoner og ville oppnå mye med prosjektet. Alle hadde ideer og ønsker om ulik funksjonalitet til applikasjonen. Derfor måtte vi prioritere hva vi ville implementere. Dette i tillegg til tidspress gjorde at enkelte funksjoner ble utelatt og andre nedprioritert, men er fullt mulig å arbeide med i videreutvikling.

8.1 LocationForecast & Stedstjenester

Vi har gjort flere endringer i forhold til hvor vi spør bruker om å tillate stedstjenester. Vår største utfordring her er at hvis vi spør brukeren om stedstjenester når kartet dukker opp vil kartet lastes inn underveis som pop-up vinduet er på skjermen. Dermed vil ikke kartet være oppdatert i forhold til stedstjenesten. Vi løste dette midlertidig med å lage en knapp i intro skjermen med navn "GET CURRENT LOCATION" som får pop up vinduet til å dukke opp. Fordelen med denne løsningen er at hvis brukeren tillater stedstjenester i et så tidlig stadie så kan vi laste inn kartet med brukerens posisjon umiddelbart. En annen fordel er at vi ikke tvinger brukeren til å bruke nåværende lokasjon, men at dette er noe brukeren selv kan initiere. Ulempen her er at knappen kan bli oversett av bruker og vi vil etter det aldri spørre brukeren igjen. Vi hadde løst dette ved å få pop-up-en til å dukke opp med en gang intro skjermen åpnes og dermed gjøre det umulig for brukeren å overse. Det er også mulig å spørre brukeren om stedstjenester akkurat i det kartet åpnes og etter at brukeren har gitt respons, laste kartet på nytt.

8.2 Bugs & loading

Etter implementeringen av "Routes" siden, støtet vi på en bug rundt knappen til "Complete-funksjonen" til den valgte ruten. Når brukeren velger et cardview med en rute på "Routes" siden, blir brukeren sendt til en midlertidig side ("DirectionsFragment") som viser den valgte ruten med et start- og sluttspunkt på kartet. På denne siden har vi en knapp med

teksten “Complete” som skal legge til antall kilometer på ruten, til den totale lengden brukeren har kjørt. Problemet oppstår når brukeren trykker på knappen også bytter fragment til “Profile” nede på navigasjonsbaren, uten å trykke knappen en gang til for å avslutte aktiviteten. Da vil kilometerne bli lagt til i profilen, men hvis brukeren går tilbake til «Routes» siden vil den foregående siden med den valgte ruten forbli. Dermed har vi laget en evig loop. Det er mulig å forlate denne loopen ved å trykke på “Complete” knappen og deretter den grønne knappen som dukker opp i ettermat. Dette er ikke ideelt og vi ønsker for eventuell videre arbeid å fikse på denne buggen. Ved å trykke “Complete” knappen ønsker vi å legge til kilometer akkurat som før, men at vi forlater denne midlertidige siden og overfører brukeren tilbake til “Routes” siden med cardview-ene. I tillegg hadde det vært hjelpsomt for brukeren om vi hadde implementert en “Cancel” knapp som kanselerer den valgte ruten.

Applikasjonen henter en betydelig mengde data fra ulike API-er noe som fører til lengre loading tid. Dette er et problem fordi det svekker brukervennligheten til applikasjonen. For brukeren gjelder det å ha tålmodighet spesielt når det kommer til “Routes” fragment. Her henter vi ruter fra Bisykkel API-et fra den nåværende måneden vi er i. Dermed har det mye å si for loading tiden om vi er i begynnelsen eller i slutten av måneden. Dette skyldes at nye sykkelruter legges inn etterhvert som Bisykkel brukere sykler, dermed vil det på slutten av måneden være mange flere ruter tilgjengelig. For brukervennligheten burde loading tiden være av stor prioritet, noe vi ville ha fokusert på i videre arbeid. Vi henter inn luftkvalitet og bilder per rute, men dette kunne vært gjort på en klokere måte. Blant annet gjør vi ikke noen sjekk på om start og sluttkoordinatet allerede har hentet bilde og luftkvalitet. Hvis vi hadde sjekket for dette kunne vi ha unngått mange API-kall (som er kostbart med tanke på kjøretid). Vi kunne også ha lagt til en progressbar eller en tekst der det står “Loading...” for å informere brukeren at innlastinga vil ta tid.

8.3 Profil

I arbeidet fremover vil det første steget være å refaktorere og videreutvikle profil-siden. Vi hadde problemer med å integrere navigasjonsgrafen til navigasjonsbaren sammen med navigasjonsgrafen til profilen. Dette har vi løst midlertidig ved å ha to views oppå hverandre, og sette dem View.VISIBLE eller View.GONE. Imidlertid har vi ved denne løsningen fått teknisk gjeld.

I tillegg ønsker vi å gjøre “Profile”-fragmentet dynamisk. Den nåværende tilstanden til fragmentet er statisk og viser ikke hele sidens innhold når enheten er i landscape-mode. Vi vil fikse dette ved å gjøre hele profilsiden til et recyclerview. Da vil det være mulig å scrolle seg rundt og se all informasjonen på siden.

Videre ville vi sett på “Bookmark”-funksjonen. Den er delvis implementert i den forstand at “Bookmark”-knappen er til stede og den har en funksjon klar til å videreutvikles. I tillegg så har de bokmerkede rutene en tilordnet plass i profilen. Det som gjenstår er å skille de bookmarkerte rutene fra resten og fremstille de til brukeren. Videre vil det også være nødvendig å koble funksjonen til Firebase slik at appen husker brukerens bookmarkeringer selv om appen blir slettet, restartet eller lastet ned på en annen enhet.

Lignende mangler finner vi hos “My history”-funksjonen, men i en litt større grad. Det eneste som er tilrettelagt på historikk-funksjonen er placeholderen i profilen. Rutene har enda ingen variabler som markerer om de er fullført eller ikke. Etter det vil funksjonen kreve en tilkobling til Firebase for å lagre brukerens fremgang i likhet med “Bookmark”-funksjonen. Til slutt må vi presentere rutene til brukeren i profilen.

For å ferdigstille vår visjon av profilen gjenstår det kun en funksjon for å vise brukeren antallet kilometer som er syklet denne uken. Denne funksjonen vil hver eneste uke restartes til null og telle brukerens kilometer denne uken. I tillegg vil vi implementere en deaktivering-knapp som vil slette brukerens profil fra databasen etter ønske fra brukeren. Dette vil gi brukeren mer frihet og kontroll over applikasjonen.

9.0 Konklusjon

Helhetlig er vi er veldig fornøyd med prosjektet og resultatet. Vi hadde flere utfordringer og uenigheter underveis, men vi kommuniserte og diskuterte godt og fant gode løsninger. Det har alltid vært viktig at hver av oss uttaler hva vi mener, og deretter finner vi den gylne middelvei.

Til tross for at vi hadde en fordeling der enkelte kodet, mens andre designet vil vi si at vi har kommunisert godt og lært av hverandre. Vi gav innspill og konstruktive tilbakemeldinger til hverandre uavhengig av fagområde, og til slutt kom vi fram til en løsning sammen. Det var utfordrende, men fortsatt spennende å jobbe med studenter fra ulik bakgrunn. Videre har vi

allerede snakket om at vi ønsker å jobbe videre på aspektene som vi nevnte i 8.0 (videre arbeid) i denne gruppen og forbedre appen.

Har vi laget en app som motiverer til sykling, opplyser mennesker om Oslos sykkelruter og gir syklister nyttige væropplysninger?

Vi vil konkludere med at vi har laget en app som opplyser om sykkelruter som kan inspirere andre, samtidig som vi gir anbefalinger basert på været. Imidlertid er det vanskelig å si om vi har laget en app som motiverer til sykling, da dette er noe som burde blitt testet gjennom ytterligere brukerundersøkelser. Hvorvidt vi har nådd det målet er ikke så viktig, vi tar likevel med oss svært nyttige erfaringer av å jobbe i et større tverrfaglig prosjekt.

10.0 Litteraturliste

Alfheim, H., & Almås, S. (2022, februar 7). *Kvalitet og test i prosjektarbeid*. UiO.

<https://www.uio.no/studier/emner/matnat/ifi/IN2000/v22/forelesninger/in2000.2022.02.07.test-og-kvalitet-slides.pdf>

Android Developers. (2022, mai 12). *Create dynamic lists with RecyclerView*.

https://developer.android.com/guide/topics/ui/layout/recyclerview?gclid=CjwKCAjw7IeUBhBbEiwADhiEMQjWGs73JC8j2a-9NHULuiAexlQ714vx4EimNbWDLE6LpJ64DyfnhhoC8OgQAvD_BwE&gclsrc=aw.ds

Android Developer. (2022, mars 17). *Fundamentals of testing Android apps*. Android Developers. <https://developer.android.com/training/testing/fundamentals>

Android Developers. (2022, mai). *Data layer*. Android Developers.

<https://developer.android.com/topic/architecture/data-layer>

Android Developers. (2022, mai). *Maps SDK for Android overview*. Google Developers. <https://developers.google.com/maps/documentation/android-sdk/overview>

Android Developers. (2021, oktober 27). *Handling Lifecycles with Lifecycle-Aware Components*. Android Developers. <https://developer.android.com/topic/libraries/architecture/lifecycle>

Android Developers. (2021, oktober 27). *ViewModel Overview*. Android Developers. <https://developer.android.com/topic/libraries/architecture/viewmodel>

Android Developers. (2022, februar 10). *Use test doubles in Android*. Android Developers. <https://developer.android.com/training/testing/fundamentals/test-doubles>

Bergersen, G., & Sjøberg, D. (2022, mars 7). *Forskningsmetoder / Evaluering av IT-systemer*. UiO.

https://www.uio.no/studier/emner/matnat/ifi/IN2000/v22/forelesninger/in2000.2022.3.7_metode.pdf

Bergersen, G. R. (2022, april 26). *DevOps and code management*. UiO. <https://www.uio.no/studier/emner/matnat/ifi/IN1030/v22/forelesningsressurser/2022-04-26-devops-and-code-management-preliminary.pdf>

Difi. (2016, februar 4). *WCAG 2.0*.

<https://wcag.difi.no/wcag-20.html>

Digitaliseringsdirektoratet. (2021, november 26). *Prosjektstyring og smidig utviklingsmetodikk*.

https://www.prosjektveiviseren.no/prosjektyper/digitaliseringsprosjekter/programvare_utvikling/prosjektstyring-og-smidig-utviklingsmetodikk

Google Developers. (2022, mai 16). *Firebase authentication* | *firebase documentation*.

Google.

<https://firebase.google.com/docs/auth?authuser=0>

Google Developers. (2022, mai 16). *Firebase realtime database* | *firebase documentation*.

Google.

<https://firebase.google.com/docs/database>

Google Developers. (2022, mai 17). *Authenticate with firebase using password-based accounts on Android* | *firebase documentation*. Google.

<https://firebase.google.com/docs/auth/android/password-auth>

Google Developers. (2022, mai 17). *Get started with firebase authentication on Android* | *firebase documentation*. Google.

https://firebase.google.com/docs/auth/android/start?authuser=0#sign_in_existing_users

Google Developers. (2022, mai 17). *Get started with firebase authentication on Android* | *firebase documentation*. Google.

https://firebase.google.com/docs/auth/android/start?authuser=0#sign_up_new_users

Google Developers. (2022, mai 17). *Read and write data on Android* | *firebase documentation*. Google Developers.

<https://firebase.google.com/docs/database/android/read-and-write>

Guide to app architecture. (2022, april 28). Developer Android.

https://developer.android.com/topic/architecture?gclid=Cj0KCQjwpv2TBhDoARIAsALBnVnmHLOHHZvzzITCdzErkrusRJ1ubo-y-O1quu9NjmzLpo4oz1GV5FZkaAqX6EALw_wcB&gclsrc=aw.ds

ISO. (2022, mai). *ISO/IEC 25010:2011 - Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*. ISO. <https://www.iso.org/standard/35733.html>

Kniberg, H., & Skarin, M. (2010). *Kanban and Scrum: Making the Most of Both*. C4Media Incorporated.

Kotlin. (2022, april 13). *Calling Java from Kotlin*. Kotlin.

<https://kotlinlang.org/docs/java-interop.html>

Ladas, C. (2021, juli 30). *What is Scrumban?* Agile Alliance.

<https://www.agilealliance.org/scrumban/>

- Lindsjørn, T. (2022, februar 28). *Utvikling av Android apper og bruk av patterns*. UiO.
<https://www.uio.no/studier/emner/matnat/ifi/IN2000/v22/forelesninger/utvikling-av-android-app-og-bruk-av-patterns.pdf>
- Lindsjørn, Y. (2022, februar 2.). *Teamarbeid og smidig metodikk. Lean og Scrum. Prosjektarbeid*. UiO.
https://www.uio.no/studier/emner/matnat/ifi/IN2000/v22/forelesninger/in2000.2022.02.02.agile_lean_teamwork.pdf
- Lindsjørn, Y. (2022, februar 16). *IN2000: Kravhåndtering, modellering, design og prinsipper*. UiO.
<https://www.uio.no/studier/emner/matnat/ifi/IN2000/v22/forelesninger/in2000.2022.02.16.modellering.pdf>
- Martini, A. (2022, februar 21). *Architecture and technical debt*. UiO.
https://www.uio.no/studier/emner/matnat/ifi/IN2000/v22/forelesninger/IN2000_2022-02-21_architecture_and_TD.pdf
- Maus, A. (2010). *Kravhåndtering*. UiO.
<https://www.uio.no/studier/emner/matnat/ifi/INF1050/v10/undervisningsmateriale/INF1050Forelesning4-Kravhåndteringv2010.pdf>
- Norse. (2022, mai 15). *1.4.1 Bruk av farge (Nivå A)*.
<https://norse.co/1-4-1-bruk-av-farge-nivaa-a>
- NTNU. (2020, januar 1). *Kunnskapsbanken: Hva er universell utforming?*.
<https://naku.no/kunnskapsbanken/universell-utforming>
- Sjøberg, D., & Bergersen, G. (2022, mars 8). *Introduksjon til systemutvikling, Prosesser og prosessmodeller*. UiO.
https://www.uio.no/studier/emner/matnat/ifi/IN1030/v22/forelesningsressurser/in1030_2022.03.08_intro_systemutvikling.pdf
- Sjøberg, D. (2022, mars 15). *Kravhåndtering, Lærebok kap. 4* [Lysarkpresentasjon].
https://www.uio.no/studier/emner/matnat/ifi/IN1030/v22/forelesningsressurser/in1030_2022.03.15_kravhandtering.pdf
- Sommerville, I. (2015). *Software Engineering*. Pearson.
- Sommerville, I. (2019). *Engineering Software Products: An Introduction to Modern Software Engineering*. Pearson.
- Steen, S. (2020, februar 18). *Oslo suser frem som sykkelhovedstad. Internasjonalt tilrettelegges det for sykkel som aldri før*. VartOslo.

<https://vartoslo.no/hele-oslo-stine-steen-sykkel/oslo-suser-frem-som-sykkelhovedstad-internasjonalt-tilrettelegges-det-for-sykkel-som-aldri-for/115757>

Utilsynet. (2022a, mai 13). *WCAG 2.0-standarden*.

<https://www.uutilsynet.no/wcag-standarden/wcag-20-standarden/86>

Utilsynet. (2022b, mai 14). *WCAG 2.0-standarden: 1.4.1 Bruk av farge (Nivå A)*.

<https://www.uutilsynet.no/wcag-standarden/141-bruk-av-farge-niva/93>

Wang, J. (2021, november 22). *Google's latest Android version distribution numbers show 11 in dead heat with 10*. Android Police.

<https://www.androidpolice.com/googles-latest-android-version-distribution-numbers-show-11-in-dead-heat-with-10/>

11.0 Vedlegg

Vedlegg 1: Teamavtale

6. Teamavtale

6.1. Tilstedeværelse

Etter avtale har vi kommet til en enighet om en hybrid løsning, hvor noen møter fysisk og andre digitalt over Zoom. Hvis man ikke har mulighet til å være tilstede under møte, er avtalen at man må sende en melding i gruppen på Messenger.

6.2. Tidsbruk

Gruppen har som utgangspunkt satt et krav på 26 timer, som dekker alt av forelesning, møter og prosjektet osv. Ved behov og variert arbeidsmengde vil dette justeres tilsvarende. Utover i prosjektet regner vi med at det blir mer enn 26 timer i uken.

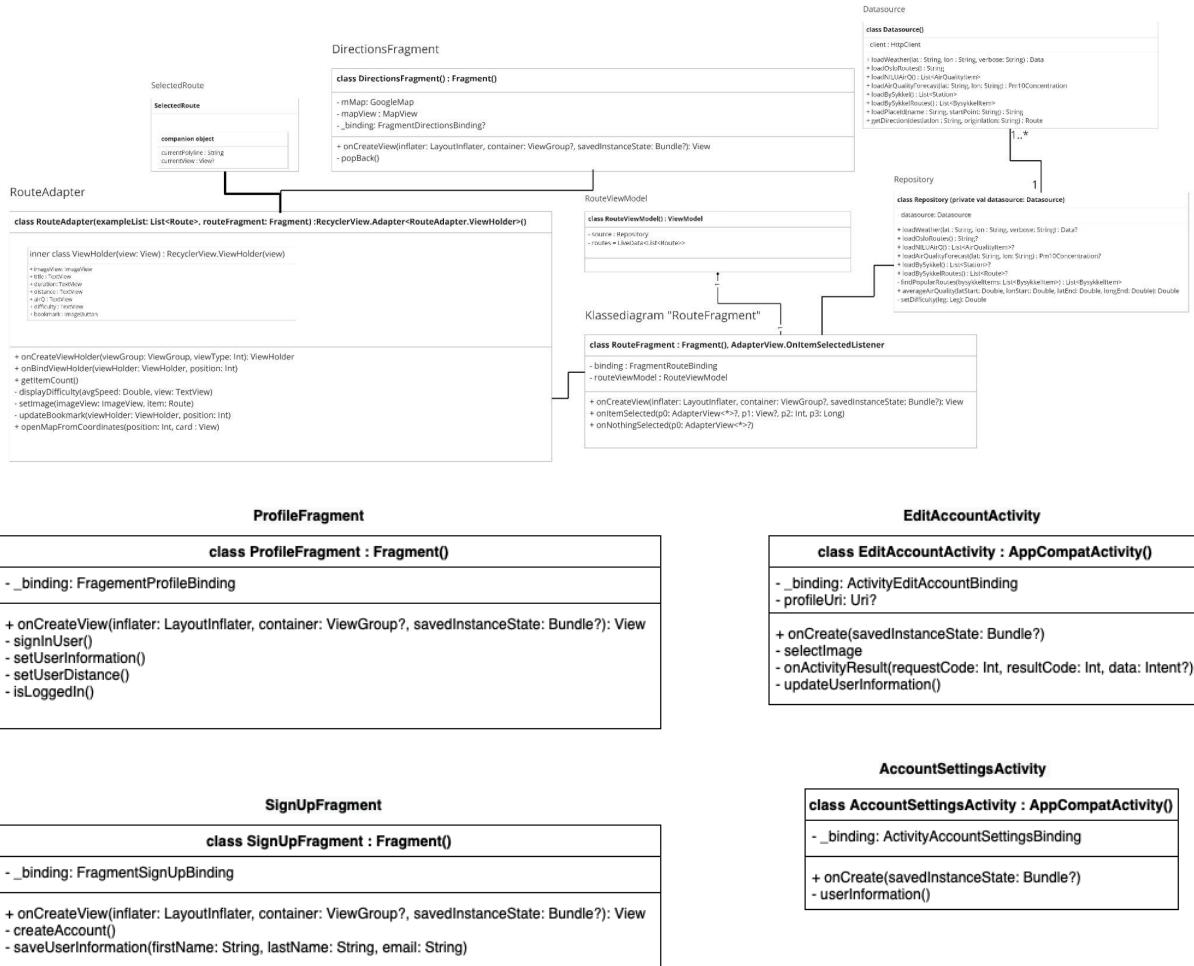
6.3. Forventninger til den enkeltes bidrag

Vi har snakket om forventningene på Kick-offen og hva hvert medlem kan bidra til. Vi ønsker at alle skal bidra på de ulike delene av prosjektet for godt læringsutbytte, men ha ekstra ansvar innenfor sitt ansvarsområde. Noe vi også setter veldig sentralt er tilgjengelighet. Med dette mener vi at gruppemedlemmene skal kommunisere med hverandre hvis de eller noen andre trenger hjelp. Utover dette forventes det overholdelse av teamavtalen.

6.4. Avvik og uenigheter

Ved avvik eller uenigheter ønsker vi som gruppe at vi kan ha en åpen samtale mellom de involverte. Istedentfor at personene som er misfornøyde med hverandre går bak ryggen til hverandre og skaper et negativt miljø.

Vedlegg 2: Klassediagram



Vedlegg 3: Innledende spørreundersøkelse

Rapport fra «Spørreundersøkelse IN2000»

Innhentede svar pr. 21. mars 2022 23:22

Leverte svar: **57**

Påbegynte svar: **0**

Antall invitasjoner sendt: **0**

Vi skal designe og utvikle en app som viser forslag over sykkelruter i Oslo i forbindelse med emnet IN2000. Svarene dine blir anonyme, og ved å svare på denne undersøkelsen samtykker du til innsamling og bruk av data til formålet med dette prosjektet.

We are going to design and develop an app that provides suggestions for cycling routes in Oslo in connection with the subject IN2000. Your answers will be anonymous, and by answering this survey, you consent to the collection and use of data for the purpose of this project.

Hva er ditt kjønn? / What is your gender? *

Svar	Antall	Prosent
Mann/Male	28	49,1 %
Kvinne/Female	29	50,9 %
Andre/Other	0	0 %

Hvor gammel er du? / How old are you? *

Svar	Antall	Prosent
16-25	36	63,2 %
26-35	21	36,8 %
36-45	0	0 %

46-55	0	0 %
-------	---	------------

56-	0	0 %
-----	---	------------

Hvor bor du? / Where do you live? *

Svar	Antall	Prosent
------	--------	---------

Oslo	51	89,5 %
------	----	---------------

Utenfor Oslo / Outside Oslo	6	10,5 %
-----------------------------	---	---------------

I snitt hvor ofte sykler du? / Approximately how often do you ride a bike? *

Svar	An tall	Prosent
------	------------	---------

Nesten hver dag / Almost everyday	13	22,8 %
-----------------------------------	----	---------------

3-4 ganger per måned / 3-4 times per month	12	21,1 %
--	----	---------------

1-2 ganger per måned / 1-2 times per month	2	3,5 %
--	---	--------------

Mindre enn 1-2 ganger per måned / Less than 1-2 times per month	13	22,8 %
---	----	---------------

Aldri / Never	17	29,8 %
---------------	----	---------------

Hvorfor/hva motiverer deg til å sykle? Why/what motivates you to ride a bicycle? *

(multiple choices)

Svar	Ant all	Prosent
Fritid, rekreasjon eller moro / Leisure, recreation or fun	13	22,8 %
Helse og trening / Health and fitness	12	21,1 %
Konkurranse / Competition	0	0 %
Å pendle / To commute	20	35,1 %
Å spare penger / To save money	14	24,6 %
Miljø bekymringer / Environmental concerns	6	10,5 %

Shopping eller dagligvare / Shopping or groceries	5	8,8 %
Å lindre stress / To relieve stress	7	12,3 %
Raskere enn andre alternativer / Faster than other options	16	28,1 %
Annet / Other	0	0 %

Spesifiser hvorfor/hva. Specify why/what.

Hvilke(n) faktorer demotiverer deg til å sykle? / What factor(s) demotivate you to bike? *

(multiple choice)

Svar	A nt al l	Prosent
Luftkvalitet / Air quality	3	5,3 %
Helsetilstand / Your health condition	3	5,3 %
Dårlig vær og veitilstand / Bad weather and road condition	2 0	35,1 %

UV-indeks / UV index (Sunburn)	2	3,5 %
Temperatur / Temperature	1	22,8 %
	3	_____
Avstand eller tid til destinasjon / Distance or time to destination	1	22,8 %
	3	_____
Frykt for kjøretøykollisjoner eller trafikk / Fear of vehicle collisions or traffic	4	7 %
Mangel på utbygde sykkelruter eller veier / Lack of developed bike routes or lanes	7	12,3 %
Mangel på sykkelfasiliteter på destinasjonen / Lack of bike facilities at destination	7	12,3 %
Ønsker ikke å ankomme svett / Don't want to arrive sweaty	1	19,3 %
	1	_____
Andre / Other	2	3,5 %

Spesifiser hva / Specify what

It's too hard to ride the bike uphill in some part of Oslo when the bike doesn't have the assisted pedaling

Ikke alltid det passer å kle seg sykkelvennlig/ha med skift/har mulighet til å skifte der man skal.

Sykler du alene? / Do you bike alone? *

Svar	Antall	Prosent
Aldri / Never	0	0 %
Noen ganger / Sometimes	3	11,1 %
Ofte / Often	9	33,3 %
Alltid / All the time	15	55,6 %

Har du brukt en sykelapp fra før? / Have you ever used a bicycle app? *

Svar	Antall	Prosent
Ja / Yes	4	14,8 %
Nei / No	23	85,2 %

Hvilke(n) app? / Which app?

Telles Strava? Viser sykkelruter hvertfall

Google maps, Strava

Strava, Komoot

Bysykkel

Bruker du fortsatt appen? / Are you still using the app? *

Svar	Antall	Prosent
Ja / Yes	4	100 %
Nei / No	0	0 %

Hva liker du ved denne appen? / What do you like about the app?

Man kan tracke turene sine. Lagre turer man liker og sykle de senere. Vanvittig nøyaktig posisjonsmåler uten å bruke alt for mye batteri. Nice ui. Tracker tid, distanse, høydemeter, og gjennomsnittlig hastighet. Meget Nice app

- Den tar tiden jeg bruker på strekningen. - Viser tiden det tar å komme seg frem til din distinasjon - Nøye beskrivelse av ruten - Kan sammenligne meg med andre som har syklet samme rute.

Kart, ruter, se hva venner gjør

Se hvor bysykkel er

Hva liker du IKKE om appen? / What do you dislike about the app?

Er egentlig ingenting jeg ikke liker. Er kanskje ikke så stor fan av all denne delingen av turer man har gjort. Men man behøver jo ikke å dele da

- Vanskelig og tidskrevende å finne alternative ruter. - Sender noen ganger til en trapp eller motorvei hvor det er vanskelig å komme seg frem med sykkel.

De beste tjenestene er bak betalingsmur

Irriterende å finne et ledig stativ

Hvorfor sluttet du å bruke appen? / Why did you stop using the app?

Var det noe som du likte ved appen? / Is there anything that you liked about the app?

Hvilke(n) faktorer demotiverer deg til å sykle? / What factors demotivate you to bike? *

(multiple choices)

Svar	A nt al l	Prosent
Luftkvalitet / Air quality	3	5,3 %
Helsetilstand / Your health condition	7	12,3 %
Dårlig vær og veitilstand / Bad weather and road condition	2 5	43,9 %
UV-indeks / UV index (Sunburn)	5	8,8 %
Temperatur / Temperature	1 6	28,1 %
Avstand eller tid til destinasjon / Distance or time to destination	1 7	29,8 %
Frykt for kjøretøykollisjoner eller trafikk / Fear of vehicle collisions or traffic	9	15,8 %
Mangel på utbygde sykkelruter eller veier / Lack of developed bike routes or lanes	5	8,8 %
Mangel på sykkelfasiliteter på destinasjonen / Lack of bike facilities at destination	3	5,3 %

Ønsker ikke å ankomme svett / Don't want to arrive sweaty	1	19,3 %
Andre / Others	7	12,3 %

Spesifiser hva / Specify what

Muscle aches and fatigue

Ikke nødvendig. Enklere å gå eller kollektivt

Det er slitsomt å frakte med rundt om jeg skal flere steder på en dag og få parkert
steder

I don't own a bike

Har ikke sykkel

I don't have a bike.

Foretrekker å gå istedenfor å sykle

Hvis vi skulle lage en app som belønner deg hvor ofte/langt du sykler, ville det motivert deg
til å sykle (oftere)? / If we were to make an app that rewards you how often/long you ride a
bicycle, would that motivate you to cycle more (often)? *

Svar	Antall	Prosent
Ja / Yes	33	57,9 %
Nei / No	24	42,1 %

Andre forslag eller ting du vil at vi skal ta i betraktnsing når vi utvikler denne appen? / Any other suggestions or things you would like us to take into consideration when developing this app?

Hadde vært nice å kunne se hvor man kan parkere sykkelen. Finnes det
sykkelstativer på Atheltica Domus, for eksempel?

Hvordan skal dere tjene penger på det her? Hvordan skal dere lage noe som er bedre enn Strava? Det finnes mye bra fra før.

Måle hvor langt du har sykklet, og kunne varsle om feks(idagnhar du syklet tilsvarende 40 ganger rundt en friidrettsbane eller denne uken har du syklet tilsvarende fra X til y)

Gamification er nyttig!

Det skal mye til for at gamification i en app motiverer meg til fysisk aktivitet.

Kanskje gjøre sånn at man ser venners måloppnåelser også, slik at man motiverer hverandre? Evt legge til et kommentarfelt for å heie på hverandre :-)

Hadde vært kjekt med varsel dersom det er fare for glatte veier ute før man drar ut på sykkeltur.

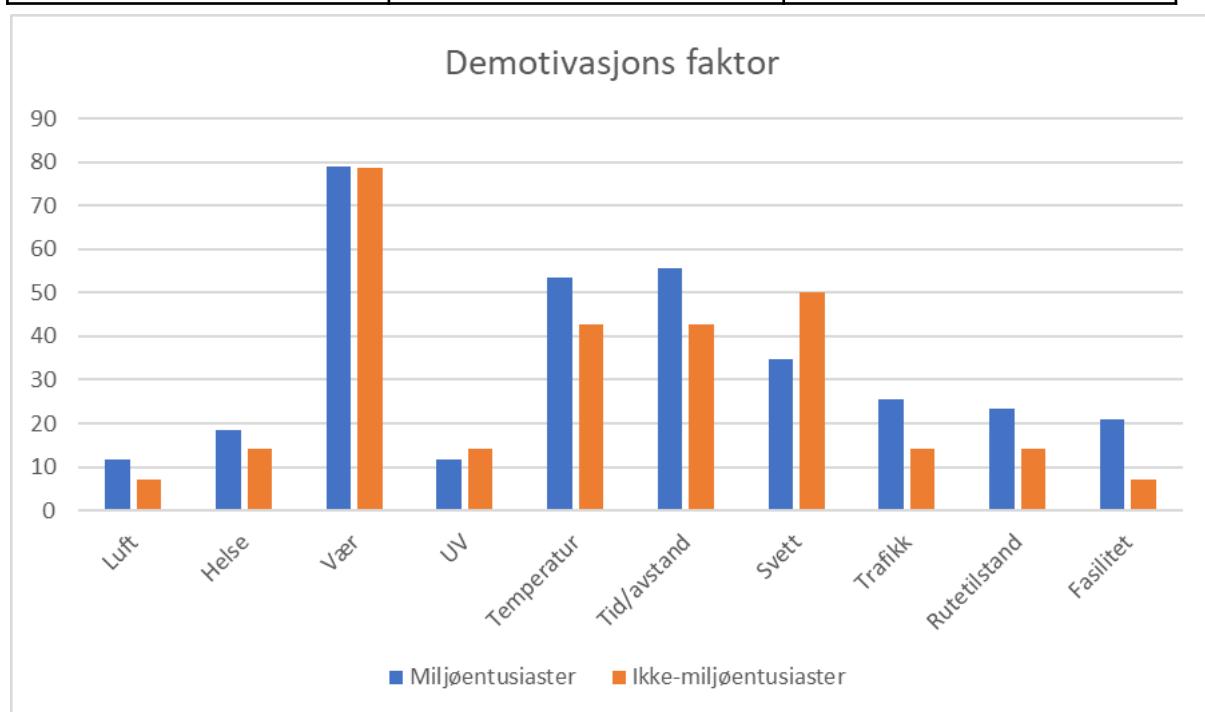
If possible, I would like to have a bicycle map that will be integrated into the app, so I do not need to open Google Map at the same time.

	Miljøentusiaster (43)	Ikke-miljøentusiaster (14)
Aktive syklister	51.2% (22)	35.7% (5)
	Miljøentusiaster (43)	Ikke-miljøentusiaster (14)

Demotivasjons faktorer (miljøentusiaster vs ikke-miljøentusiaster)

	Miljøentusiaster (43)	Ikke-miljøentusiaster (14)
Luft	11.6% (5)	7.1% (1)
Helse	18.6% (8)	14.3% (2)
Vær	79.1% (34)	78.6% (11)
UV	11.6% (5)	14.3% (2)
Temperatur	53.5% (23)	42.9% (6)
Tid/avstand	55.8% (24)	42.9% (6)

Svett	34.9% (15)	50.0 % (7)
Trafikk	25.6% (11)	14.3% (2)
Dårlig rutetilstand	23.3% (10)	14.3% (2)
Fasilitet	21.0% (9)	7.1% (1)

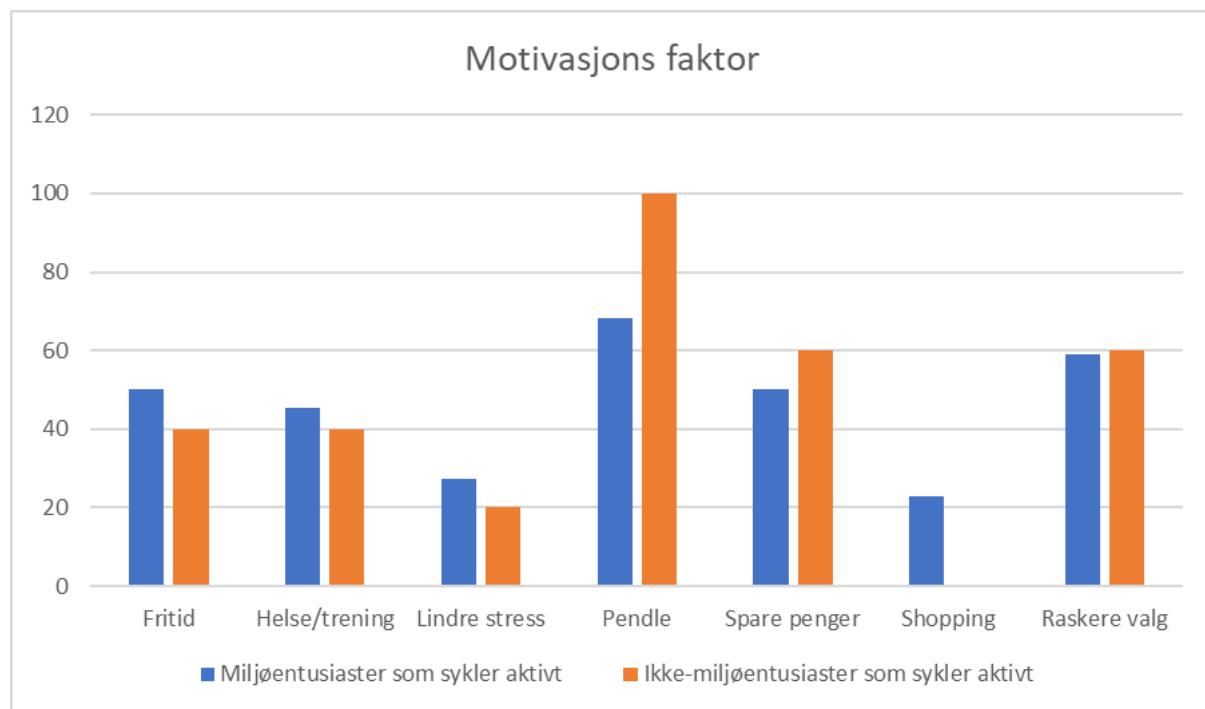


	Miljøentusiaster (43)	Ikke-miljøentusiaster (14)
For gamification	62.8% (27)	42.9% (6)

Motivasjonsfaktorer (miljøentusiaster som sykler aktivt vs ikke-miljøentusiaster som sykler aktivt)

	Miljøentusiaster som sykler aktivt (22)	Ikke-miljøentusiaster som sykler aktivt (5)
--	--	--

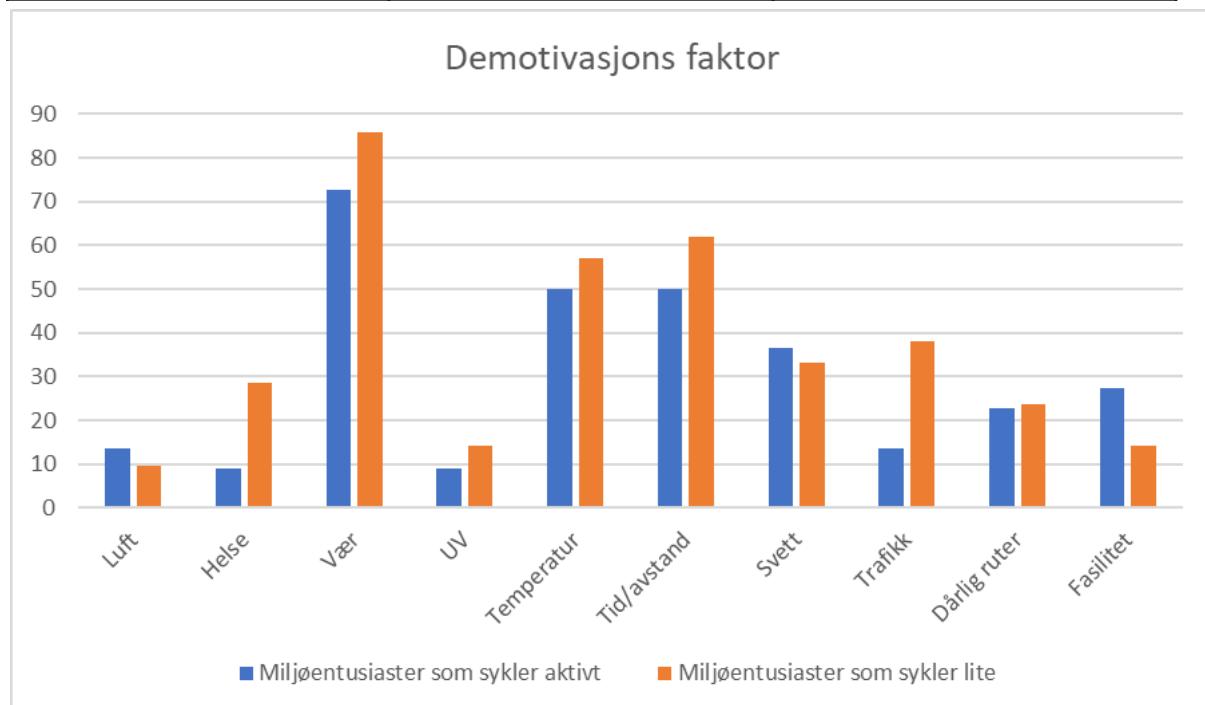
Fritid	50.0% (11)	40.0% (2)
Helse/trening	45.5% (10)	40.0% (2)
Lindre stress	27.3% (6)	20.0% (1)
Pendle	68.1% (15)	100% (5)
Spare penger	50.0% (11)	60.0% (3)
Shopping	22.7% (5)	0% (0)
Raskere valg	59.0% (13)	60.0% (3)



Demotivasjons faktorer (miljøentusiaster som sykler aktivt vs som sykler lite)

Miljøentusiaster som sykler aktivt (22)	Miljøentusiaster som sykler lite (21)
---	---------------------------------------

Luft	13.6% (3)	9.5% (2)
Helse	9.1% (2)	28.6% (6)
Vær	72.7% (16)	85.7% (18)
UV	9.1% (2)	14.3% (3)
Temperatur	50.0% (11)	57.1% (12)
Avstand/tid	50.0% (11)	61.9% (13)
Svett	36.4% (8)	33.3% (7)
Trafikk	13.6% (3)	38.1% (8)
Fine ruter	22.7% (5)	23.8% (5)
Fasilitet	27.3% (6)	14.3% (3)



Vedlegg 4: Intervjuguide

Interview Questions:

- Hvor ofte sykler du?
 - Hva er motivasjonen din til å sykle?
 - Hva hindrer du til å sykle?
- Hvor mye tenker du om miljø?
- Bruker du en sykkel app mens du sykler?
 - Hvis ja, hvilken?
 - Hvis nei, hvorfor ikke?
 - Hva er det du ikke likte?
- Hvilkens funksjon tror du er spesielt nyttig når du sykler?
- Hvilkens funksjon mener du ikke nødvendigvis er viktig?

Forklaring av prototypen

- Gjerne fokus på funksjon, men vi setter også pris på innspill på farger og ikon også.
- Map page
 - Hva synes du “P” og sykkel ikoner betyr?
- Route page
 - “Favorite” funksjon funker ikke.
 - Noen “back” knapp navigerer til den første siden
 - Kan ikke scroll up/right
 - Filter→Click card→Start to navigate→Complete
- Profile page

Brukbarhetstesting

- *la brukeren teste og leke med prototypen
- Improvisere oppgaver hvis hun ikke har brukt noe funksjoner
- Hva er ditt førsteinntrykk av prototypen?

Map page

- Hva synes du om oppsettet? (plasseringen av ikonene)
 - Hva liker du spesielt?
 - Hvilke endringer ønsker du å se?

- Er mengden av informasjonene passende? For mye eller for lite?
 - If it is too much, what do you want to remove?
 - If it is too little, what would you like to add?

Route page

- Filter
 - Hvilke kategorier ønsker du å se i den filter menyen?
 - f.eks distance, routes with best air quality, routes with less traffic, difficulty and time.
 - Noen endringer?
- Card views med forskjellige steder (Once you click on “Routes”)
 - Liker du hvordan oppsettet ser ut akkurat nå? (viser forskjellige steder med “card views”)
 - Hva ønsker du å se på de forskjellige “card views”?
 - f.eks oversikt over lengde og tid?
 - Ikonen for favoritt:
 - Foretrekker du en “lap” eller en “stjerne”?
 - Hvilken farge foretrekker du for det favorittikon?
 - Steder som er ikke lagret som favoritt, hvilken farge foretrekker du til ikonet? Skal det være gjennomsiktig eller ugjennomsiktig?
 - Noen endringer?
- Information cards (Once you click on the individual cards)
 - Passende mengde?
 - Hvilken informasjon ønsker du å se de individuelle card views?
 - f.eks beskrivelse, lengde, tid, traffic, vanskelighetsgrad og poeng (gamification)?
 - Ikonene for vanskelighetsgrad:
 - “Easy”, “Moderate”, “Hard” med forskjellige farge?
 - Ikonene for vanskelighetsgrad basert på elevation og lengde:
 - “Light”, “Heavy” traffic
 - Start knapp

Profile

- Synes du det er nyttig å ha en brukerprofil?
- Hva ønsker du å se i brukerprofilen?
 - Totalt skylt kilometer
 - Oversikt over reisene dine når du sykler, som du kan gå tilbake til senere?
(Favoritt vs historikken)
 - Se på andre venner som bruker appen?
 - Mulighet å trykke på andre venners profil og deres stats? eller mulig å holde seg skjult uten å dele informasjonen med andre?
- Er det noe informasjon som ikke er nødvendig?
- Gamification
 - Formål med appen - engasjere folk til å sykle.
 - Hvordan ønsker du å bli “rewarded”?
 - f.eks: du kan få poeng som leder til å få badges, titles
 - f.eks: the total length that will be compared to the actual geography
 - <https://www.developgoodhabits.com/fitbit-badge-list/>

Avslutningsspørsmål

- Noen sluttkommentar eller ting vi må ta hensyn til?

Vedlegg 5: Samtykkeskjema

Oslo / 07.04.22

Vil du delta i brukerundersøkelsen «Sykkelturer app i Oslo»?

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

Formål

Målet med prosjektet er å skape en applikasjon som motiverer til sykling, opplyse mennesker om Oslos ulike sykkelruter og gi syklister nyttige væropplysninger knyttet til rutene de ønsker å sykle.

Deltakelse

Du blir spurta om å delta fordi du faller innenfor min målgruppe, definert som miljøinteresserte hobbycyklister. Dersom du velger å delta ønsker vi å intervju deg for vår datainnsamling. Intervjuet vil vare i 30 minutter, og vi kommer til å gjøre lydopptak.

Frivillig deltagelse

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

Personvern: innsamling, oppbevaring, behandling og bruk av dine opplysninger
Ingen sensitive personopplysninger (jf. Personvernforordningens artikkel 9 og 10) vil bli innsamlet. Personlige opplysninger om deg vil kun benyttes til formålene beskrevet i dette informasjonsskrivet. Vi behandler opplysningene konfidensielt og i samsvar med personvernregelverket.

Personlige opplysning innsamlet i opptaket vil bli anonymisert i transkriberingen og rapporteringen senest 08.04.22; ingen andre enn jeg og gruppen min vil ha tilgang til dataen, og det som oppbevares av anonymisert rapportering fra intervjuet vil følge Universitetet i Oslo sine rutiner for sikker oppbevaring.

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

Hva skjer med innsamlet data når studentprosjektet avsluttes?

Opptaket og notater blir slettet senest 20.05.22. Dette gjelder også anonymiserte og avidentifiserte opplysninger om deg.

Rettigheter

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

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

Hvis du har spørsmål til undersøkelsen, eller ønsker å benytte deg av dine rettigheter, ta kontakt med Michiru Tamura Nygaard eller mine veileder Emil Damsgård Knutsen og Emil Eriksmoen Stensland på e-post emildkn@ui.no eller emiles@ui.no.

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

Med vennlig hilsen

Gruppe 3 i IN2000

Samtykkeerklæring

Jeg har mottatt og forstått informasjon om brukerundersøkelsen «Sykkeleruter app i Oslo», og har fått anledning til å stille spørsmål. Jeg samtykker til:

- Deltakelse i brukbarhetstesting og intervju
 Opptak i løpet av brukbarhetstesting og intervju

Jeg samtykker til at mine opplysninger behandles frem til semesteret er avsluttet.

Sted og dato

Fullt navn

Signatur

Vedlegg 6: Høyoppløselig prototype

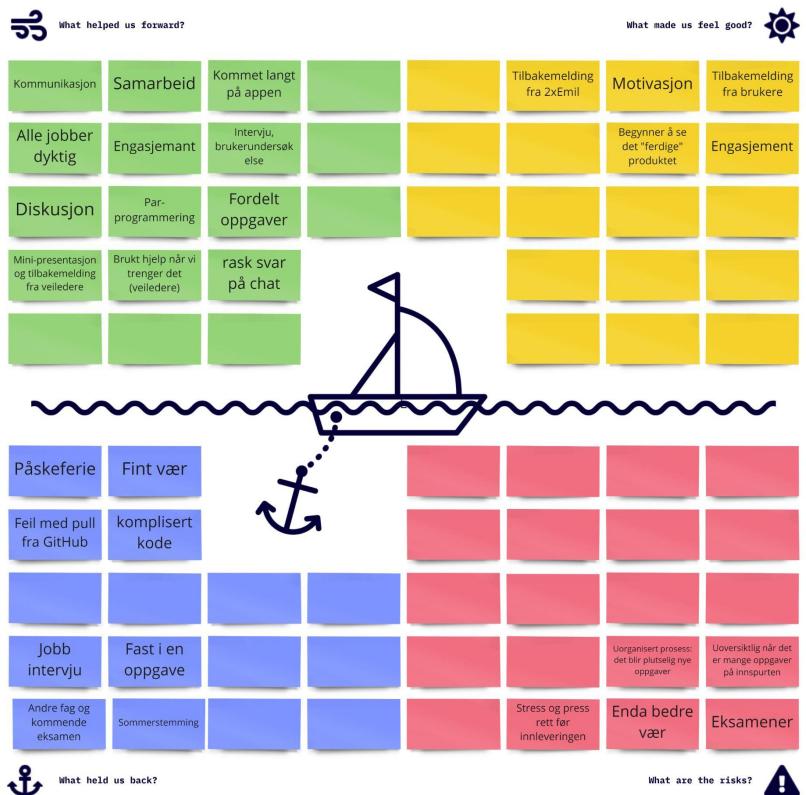
<https://www.figma.com/proto/QcAV3VSAD7CqqvnWu0BuFa/High-fidelity-prototype?page-id=0%3A1&node-id=2%3A58&viewport=282%2C250%2C0.06&scaling=min-zoom&start-in-g-point-node-id=2%3A58>

Vedlegg 7: Retrospektive møter

Retrospektiv 28.03



Retrospektiv 25.04



Retrospektiv (09.03)

