Masteroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

Kimia Dadar Abtahi
Trym Vegard Gjelseth-Borgen

Applikasjon for effektivisering av tilsyn med sau på utmarksbeite

Masteroppgave i Datateknologi
Veileder: Svein-Olaf Hvasshovd
Juni 2021

NTNU
Kunnskap for en bedre verden

Kimia Dadar Abtahi
Trym Vegard Gjelseth-Borgen

Applikasjon for effektivisering av tilsyn med sau på utmarksbeite

Masteroppgave i Datateknologi
Veileder: Svein-Olaf Hvasshovd
Juni 2021

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU
Kunnskap for en bedre verden

Sammendrag

Rundt to millioner sauer slippes på utmarksbeite hvert sommerhalvår. Dette er ikke uten risiko, da omlag 3-7% av sauene mister livet som følge av sykdom, ulykker eller rovdyr i løpet av denne perioden. For å sikre velferden til sauene kreves det fra norske myndigheter at sauebønder gjennomfører en oppsynstur minst én gang i uken. Formålet med oppsynsturen er å kartlegge hvor sauene er på beitet, hvordan de beveger seg og å oppdage eventuelle skadde eller døde sauer. Per dags dato utføres slike oppsynsturer ved at sauebonden drar ut på beitet og noterer ned den nødvendige informasjonen om sau og lokasjon ved hjelp av penn og papir. Det var derfor et ønske om å få utviklet en mobilapplikasjon som kan bistå med denne oppgaven. Mobilapplikasjonen burde kunne brukes blindt og med bare en hånd ettersom at det ofte tas i bruk kikkert under oppsynsturen. Det var også et ønske om å få på plass funksjonalitet som vil gjøre det enklere å dele informasjonen som har blitt samlet inn med andre bønder innenfor eget beitelag.

Prosjektet i sin helhet har bestått av to deler, et fordypningsprosjekt og selve masteroppgaven. I fordypningsprosjektet ble det gjort et dypdykk i eksisterende teknologi og metoder for utvikling av brukergrensesnitt som kan brukes blindt. Med bakgrunn i funnene fra litteratursøket ble deretter tre forskjellige grensenitt utviklet og brukertestet for å undersøke hva som fungerer best for å registrere sau under en oppsynstur. Resultatene fra fordypningsprosjektet ble brukt for å utvikle et grensesnitt som til slutt ble implementert i den endelige versjonen av applikasjonen.

Masteroppgaven startet med et litteratursøk for å kartlegge bondens behov og dagens utfordringer i forbindelse med informasjonsregistrering under en oppsynstur. Det ble deretter laget en kravspesifikasjon med bakgrunn i litteratursøket og i dialog med veileder professor Hvasshovd, som har erfaring med å gjennomføre oppsynsturer. En første versjon av applikasjonen ble så designet og utviklet. Den utviklede applikasjonen er kryssplattform for mobile enheter som kjører enten Android eller iOS. Applikasjonen har ingen dedikert tjenerløsning, men bruker Firebase som Backend-as-a-Service. I slutten av prosjektet ble det gjennomført brukertester på fem brukere. Resultatene fra testene ble deretter analysert og brukt til å komme med forslag til potensielle endringer som kan utføres slik at applikasjonen blir mer brukbar og brukervennlig.

Resultatene fra brukertestene viser at det er mulig å lage en applikasjon som kan brukes på oppsynsturer hvor det må registreres informasjon om sauflokker uten å ha mulighet til å se på skjermen. Informasjonen som kan registreres gjennom applikasjonen er mer enn tilstrekkelig med hensyn til framstillingen av rapporten som kreves av norske myndigheter. Den implementerte innlogging- og databaseløsningen

gjør det også mulig for bønder og andre tilsynsansvarlige innenfor samme beitelag å dele oppsynsturer med hverandre. Likevel ville introduksjon av en mer avansert web-applikasjon for dataanalyse være nødvendig for å ta full nytte av den innsamlede informasjonen. Sammenlignet med dagens løsning kan applikasjonen gi bønder og tilsynsansvarlige en enklere arbeidshverdag og bidra til at færre sau dør i løpet av året grunnet mer nøyaktig registrering av informasjon og bedre informasjonsflyt innad i beitelag.

Abstract

Approximately two million sheep are released for free range pasture during the summer in Norway every year. This is not without risk, as about 3-7% of the sheep lose their lives because of illness, accidents or predators during this period. To ensure the welfare of the sheep, Norwegian authorities require that sheep farmers carry out inspections of the herd at least once a week. The purpose of these inspections is to be able to map where the sheep are in the pasture, how they move and to find potential dead or injured sheep. Currently, farmers carry out these inspections by travelling to the pasture and write down necessary information about sheep and location using pen and paper. It was therefore a desire to develop a mobile application that would be able to assist with this task. The mobile application should be able to be used blindly and one handed as binoculars are often used during the pasture inspection. There was also a desire to implement functionality that would make it easier to share the gathered information with other farmers.

The project has in its entirety consisted of two parts, a research project and this master's thesis. The research project started with a deep dive into existing technology and methods used for developing user interfaces that can be utilized without having to look at the screen. With the knowledge gained from the literature review, three different user interfaces were created and tested on users to find out what worked the best for this specific use case. The results from the research project were then used to create a user interface that was implemented into the final version of the application.

The master's thesis started with a literature review to identify the farmers needs regarding information registration during a pasture inspection. From this, user requirements for the application were created in collaboration with supervisor Professor Hvasshovd, as he has experience with conducting pasture inspections. The first version of the application was then designed and developed. The developed application is a cross platform application for mobile devices running either Android or iOS. The application has no dedicated backend solution but uses Firebase as Backend-as-a-Service. In the final stages of the project, user tests were conducted on five different users. The results for the users tests were then analysed and used to propose suggestions to potential changes that can be made to make the application more usable.

The results from the user tests show that it is possible to create an application that can be used during pasture inspections where one might have to register information without being able to look at the screen. The information that can be registered through the application is more than sufficient with regard to creating the report required by the Norwegian authorities. The implemented login and database solution

makes it possible for farmers and other supervisors to cooperate and share information about pasture inspections with each other. To be able to take full advantage of the information sharing feature, an additional web application for more advanced data analysis would have to be implemented. Compared to today's solution, the application can give farmers and supervisors an easier day at work and contribute to less sheep casualties each year due to more accurate registered information and increased information flow between farmers.

Forord

Denne masteroppgaven er skrevet i forbindelse med studieprogrammet Datateknologi ved Norges teknisk - naturvitenskapelige universitet (NTNU). Prosjektet ble skrevet og utført våren 2021 av to studenter med hovedprofil i programvaresystemer.

Vi vil rette en stor takk til vår veileder professor Svein-Olaf Hvasshovd som har stilt sin kompetanse til rådighet og veiledet oss gjennom både fordypningsprosjektet og masteroppgaven. Vi vil også takke alle personene som tok tid ut av egen kalender til å la seg brukerteste av oss.

Heretter vil vi refereres til som utviklerne og teamet.

Kimia Dadar Abtahi og Trym Vegard Gjelseth-Borgen

Trondheim, juni 2021

Innhold

Figurer

Akronymer

CSS Cascading Style Sheets. 55, 56, 61, 79

DOM Document Object Model. 54

FHI Folkehelseinstituttet. 111

GNSS Global Navigation Satellite System. 21

GPRS General Packet Radio Service. 20

GPS Global Positioning System. 19, 23

GSM Global System for Mobile Communications. 20, 21

HTML HyperText Markup Language. 55, 56, 61, 79

LTE-M Long Term Evolution for Machines. 20, 21

NB-IoT NarrowBand-Iot. 20, 21, 24

Nibio Norsk institutt for bioøkonomi. 7, 16

NPM Node Package Manager. 57

NSG Norsk Sau og Geit. 14, 16, 17

OBB Organisert Beitebruk. 16, 18

SNO Statens naturoppsyn. 13, 14

SUS System Usability Scale. 108

TFOU Trøndelag Forskning og Utvikling. 22

UX User Experience. 107

WMS Web Map Service. 101

Del I

Introduksjon

Masteroppgavens introduksjon beskriver formålet og problemstillingene som har blitt utarbeidet og legges til grunne for utviklingen av mobilapplikasjonen *Sauron*. Den tiltenkte målgruppen og omfanget til applikasjonen blir også gjennomgått.

Kapittel 1

Problemstilling og mål

Dette kapittelet starter med å gå gjennom et brukerscenario for å gi innsikt i hverdagen til en bonde som driver med tilsyn på utmarksbeitet. Målet for oppgaven og problemstillingen med tilhørende forskningsspørsmål blir deretter presentert.

1.1 Brukerscenario

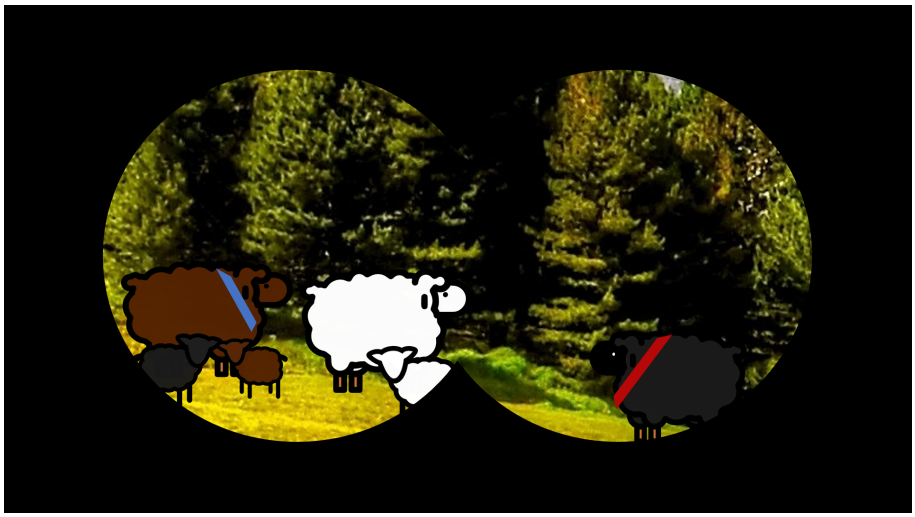
For å enklere sette seg inn i dagens situasjon og utforske problemene som oppstår under oppsynsturer, har det blitt utviklet et brukerscenario som illustrerer behovet for en digital løsning.

Geir er sauebonde og er med i et beitelag som samarbeider med å holde tilsyn på sau på utmarksbeite. Beitelaget har ansvar for til sammen 13 sauer og 17 lam. I dag er det hans tur å dra på den ukentlige oppsynsturen for å sjekke hvor sauene er og om de har det bra. For å kunne kjenne igjen og skille ulike sauflokker registreres det så mye informasjon som mulig om hver flokk. En optimal registrering inneholder informasjon om totalt antall sau i flokken, hvor mange det er av hver farge (hvit, svart, brun etc.), hvor mange søyer og lam det er, hvilken farge det er på bjelleslipsene til søyene og hvilke øremerker sauene er merket med. Likevel lar ikke dette seg alltid gjøre ettersom bøndene ofte må registrere informasjon om flokken på lang avstand. Da kan det være vanskelig å skille mellom søyer, lam og ulike bjelleslips selv om bonden tar i bruk kikkert.

Geir tar fram kartet fra forrige oppsynstur og tar utgangspunkt i lokasjonen hvor en av saueflokkene ble registrert sist. Da han kommer fram ser han at flokken har flyttet på seg. De befinner seg i en skråning over et vann et stykke unna.

Figur 1.1: Geir oppdager sau på andre siden av et vann

Geir tar fram en kikkert for å inspisere flokken nærmere. Han teller 6 sau totalt, 2 brune, 2 hvite og 2 svarte. For å unngå å huske feil velger Geir å gjøre opptellingen i to omganger.



Figur 1.2: Geir bruker kikkert for å inspisere flokken på andre siden av vannet

Han legger ned kikkerten for å skrive ned informasjonen om totalt antall sau og farger med en blyant og en notatblokk. Når han er ferdig å notere tar han fram kikkerten igjen for å telle hvor mange søyer og lam det er, samt hvilke bjelleslips søyene har rundt halsen. Avstanden er for lang for å se hvilke øremerker sauene har.

Figur 1.3: Geir tar opp kikkerten igjen men forstår raskt at noe er galt

Da Geir tar opp kikkerten skjønner han raskt at noe er galt. Nye sau har kommet til flokken og registreringene han nettopp har gjort stemmer ikke lengre. Geir forkaster notatene sine og starter på nytt igjen. Etter litt om og men klarer han til slutt å registrere korrekt informasjon om alle sauene i flokken. Geir fortsetter turen og registrerer de gjenstående sauflokkene i beiteområdet. Når turen er fullført går han tilbake til gården sin og prøver å skrive ned ruten han gikk for å finne sauene. Geir legger til notatene for dagens oppsynstur i et dokument der beitelaget har samlet dato og informasjon for sine gjennomførte oppsynsturer denne sommeren. Når beitesesongen er over, vil beitelaget lage en samlet rapport med informasjonen alle bondene har samlet fra alle oppsynsturene og sende dem til statsforvalteren i deres aktuelle fylke.

1.2 Mål for oppgaven

Dette prosjektet er delt i et fordypningsprosjekt [1] som gikk over høsten 2020 og deretter denne masteroppgaven som foregikk over våren 2021 og bygger videre på fordypningsprosjektets arbeid. Fordypningsprosjektets mål var mer tilspisset og gikk ut på å [1, s.3]:

Designe og utvikle en applikasjon som muliggjør effektiv registrering av sauer på utmarksbeite i situasjoner der brukeren kontinuerlig benytter seg av kikkert og dermed ikke kan se på skjermen.

For å nå dette målet ble det utviklet tre ulike brukergrensesnitt for registrering av sau. Alle grensesnittene ble designet for å kunne brukes uten å måtte se på skjermen. De tre ulike versjonene ble så sammenlignet mot hverandre basert på data fra brukertester på ni personer. Under masteroppgaven skal det utvikles en fullstendig applikasjon som bygger videre på resultatene fra fordypningsprosjektet. Denne har fått navnet *Sauron*. Masteroppgavens mål som er formulert som følger:

Designe, utvikle og implementere en helhetlig applikasjon som kan bistå sauebønder og beitelag med å registrere småfe på oppsynsturer.

1.3 Problemstilling

For å kunne evaluere målet for masteroppgaven er det blitt formulert forskningsspørsmål med tilhørende underspørsmål som skal besvares i løpet av gjennomføringen av masteroppgaven:

- **F1:** Hvordan utvikle et digitalt verktøy for å bistå sauebønder, beitelag og tilsynsansvarlige på oppsynstur slik at arbeidet med manuell registrering blir mer effektivt og raskere?
- **F1.1:** Kan det utvikles et system som erstatter dagens løsning med penn og papir, men fortsatt dekker alle brukerens behov?
- **F1.2:** Hvordan kan et digitalt system bistå sauebønder, beitelag og tilsynsansvarlige slik at de kan samhandle og dele informasjon om oppsynsturer med hverandre og norske myndigheter?
- **F1.3:** Hvordan utvikle et brukergrensesnitt som muliggjør registrering av sau uten å måtte se på mobilskjermen?

1.4 Oppsummering

Fra bruksscenariet kommer det fram at dagens løsning med penn og papir kan føre til vanskelige arbeidsvilkår for de tilsynsansvarlige rundt om i landet. Målet med prosjektet er å utvikle en applikasjon som kan forenkle denne prosessen og tilby utvidet funksjonalitet for samhandling innad i beitelag og med norske myndigheter.

Kapittel 2

Omfang og målgruppe

Sauebønder, personer som er med i organiserte beitelag og andre som gjennomfører oppsynsturer på utmarksbeite på vegne av sauebønder er den tiltenkte målgruppen for applikasjonen. Ifølge Statistisk Sentralbyrå er det per dags dato cirka 13 500 jordbruksbedrifter med vinterfora sauer i Norge [2]. I henhold til Nibios kilder er det 754 beitelag registrert i Organisert Beitebruk [3].

I tillegg vil norske myndigheter som er involvert i oppfølging av sau på beite, som Statsforvalteren og Mattilsynet, være interessenter ettersom applikasjonen vil kunne påvirke prosessen med innsending av oppsynsturrapporter og søknader for produksjonstilskudd.

Del II

Bakgrunn

I denne delen legges bakgrunnen for masteroppgaven fram. Først gjennomgås dagens situasjon i forbindelse med oppsynsturer og partene som er involvert i den sammenheng. Deretter presenteres tidligere arbeid innenfor samme område og inspirasjon til applikasjonen. Til slutt presenteres arbeidet som ble gjort i fordynpningsprosjektet høsten 2020.

Kapittel 3

Dagens situasjon

Dette kapittelet tar for seg dagens situasjon og problemområde. Det gås gjennom bakgrunn for tap av sau på utmarksbeite, hvilke myndigheter som er involvert i prosessen med sanking og registrering av sau, hvilke krav som blir stilt for å ha sauer på utmarksbeite og hvordan oppsynsturer gjennomføres i dag.

3.1 Tap av sau

Rundt to millioner sauer slippes på utmarksbeite i sommerhalvåret hvert år [4, s.1] [5]. Dette er ikke uten risiko, for omlag 3-7% av sauene mister livet som følge av sykdom, ulykker eller rovdyr i løpet av denne perioden [6, s.45]. Dersom bøndene kan dokumentere at skadde eller døde sauer er forårsaket av fredet rovvilt, kan de søke om erstatning fra myndighetene for å dekke tapene [7, 8]. I 2020 var det rundt 15 000 sauer og lam som ble erklært drept og førte til erstatning [9]. Resten av dødsfallene regnes som normaltap [7]. Trolig er det enda flere tap som aldri blir registrert ettersom kadaver på beite raskt råtner eller fjernes av åtseletere [10]. Ukentlige oppsynsturer er et av tiltakene for å forebygge tap og sikre dyrenes velferd samt stadfeste dødsårsak dersom et dyr har gått tapt. Sauebønder er lovpålagte å utføre disse ukentlige oppsynturene der de drar til utmarksbeitet og manuelt lokaliserer, sjekker og noterer ned dyrenes tilstand. Ved slutten av beitesesongen skal informasjonen om dyreflokken som er blitt innhentet gjennom oppsynsturene samles til en rapport som sendes til norske myndigheter.

3.1.1 Sykdom

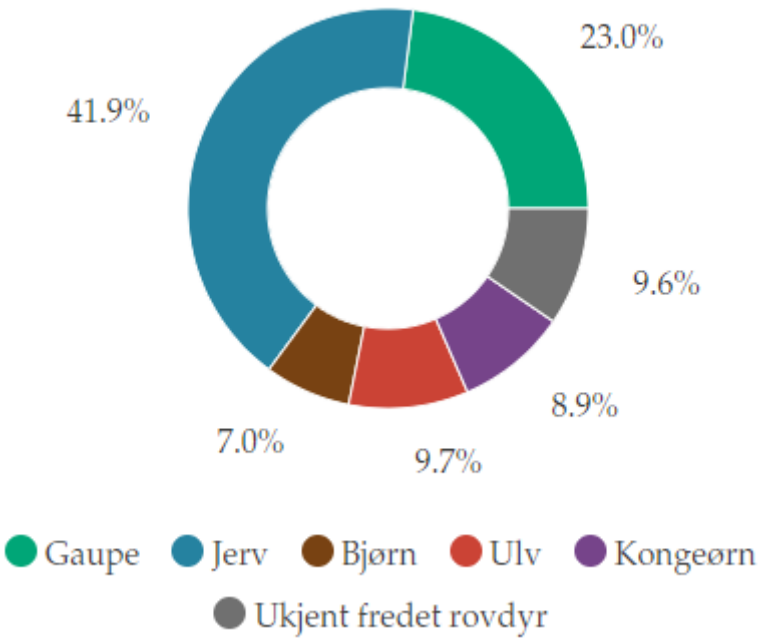
Ulike infeksjonssykdommer, feilernæring, parasittangrep og forgiftning er noen årsaker til sykdom blant småfe på utmarksbeite [11]. Ved høy dyretetthet på beite og dårlig næringstilgang vil dyrene beite tettere opp mot sin egen avføring hvor det vil være høy forekomst av parasittlarver [12]. Særlig utsatt for parasittsykdom er kopplam, som er lam oppfostret på melk fra flaske og ikke direkte fra søye [12] ettersom de går glipp av den viktige råmelka som gjør dem motstandsdyktige mot infeksjoner [13, s.44]. Langs kysten av Sørvest-Norge er sykdommen *alveld* også en fare for lam [14]. Lammene blir forgiftet og får leverskade av å konsumere planten *rome* [14]. Det finnes ingen effektiv behandling mot selve alvelden foruten å prøve å forebygge det ved å unngå beiteområder med mye romeplante [15]. Ellers er god forsyning av råmelk til nyfødte lam, vaksiner, parasittbehandling, beiterotasjon, god fôring og jevnlig tilsyn på beite tiltak som forebygger sykdom hos småfe på beite [12, 11].

3.1.2 Ulykker

Typiske ulykker som rammer småfe på beite er fallulykker, påkjørsler eller at dyrene blir sittende fast i beitelandskapet [16]. Moderne saueraser er avlet for å gi mye kjøtt og ull. Dette har resulterert i tyngre dyr. De tyngre sauene setter seg lettere fast i myrer eller elver og ikke kommer seg løs, som fører til langstrakt lidelse og død [16]. Det gjør dem også lettere bytter til rovvilt. Løshunder og rovvilt som jager dyrene kan også føre til at småfe får panikk og utsetter seg selv for fare som for eksempel å falle utenfor et stup eller bli drevet ut i vann.

3.1.3 Rovdyr

Fredet rovvilt i Norge som gir grunnlag for erstatning hvis de forårsaker tap av husdyr er bjørn, gaupe, jerv, ulv og kongeørn [17]. Under særskilte tilfeller kan det også gis erstatning for husdyr tatt av havørn, som også er et fredet rovdyr [7]. Fordelingen av rovdyrenes andel av totalt tap av sau vises på figur 3.1 under.



Figur 3.1: Fordeling av fredet rovvilts andel av total tap av sau i 2020

Bildekilde: Rovbase [18]

Parallelt som det arbeides med å sikre rovviltbestander i Norge, blir det samtidig lagt ned stor innsats for å redusere tap av sau til rovvilt. Det har vist seg å gi en positiv effekt [19]. Selv om antall innmeldte tap og påviste rovviltskader har blitt stadig færre, er det et ønske om å redusere tap av sau til rovvilt ytterligere [20]. Dette gjelder særlig i Trøndelag, som er det fylket som stod for størst tap av sau til rovvilt i 2020 [21]. Samme år ble det bevilget omlag 80 millioner kroner til tiltak som rovviltavvisende gjerder, beiteslipp og tidlig nedsanking [19]. Andre tiltak som

kan bidra til mer effektiv sanking av beitedyr samt bidra til å finne døde eller skadde dyr vil også støttes av statlige tilskudd [19]. Kostnadene for å dekke erstatning av sau og lam til rovvilt var i 2020 på nærmere 38 millioner kroner [21]. Effektivisering av oppsynsturer vil derfor ikke bare kunne forebygge lidelse blant småfe og gjøre prosessen for tilsyn enklere for bønder og beitelag, men også være økonomisk gunstig for både bønder og staten.

3.2 Involverte myndigheter

Myndighetene som er involvert i oppfølging av sau på beite i Norge er per dags dato kommunen og statsforvalterne i fylkene som er relevant for det respektive beitelaget, samt Mattilsynet og Statens Naturoppsyn. I denne seksjonen skal deres ansvarsområde og informasjonsflyten mellom dem og bøndene gjennomgås.

3.2.1 Statsforvalteren

Statsforvalteren (tidligere Fylkesmannen) har ansvar for å følge opp vedtak, mål og retningslinjer fra Stortinget og regjering, og har i tillegg en viktig rolle som bindeledd mellom stat og kommune [22]. I sammenheng med oppfølging av sau på beite, har Statsforvalteren ansvar for forvaltning av [23]:

- Produksjonstilskudd til beitelag.
- Tilskudd til spesielle miljøtiltak i jordbruket (SMIL) som fremmer natur- og kulturverdiene i jordbrukets kulturlandskap og reduserer forurensningen fra jordbruket [24].
- Erstatning for beitedyr tatt av rovvilt.
- Tilskudd til forebyggende og konfliktdempende tiltak.

3.2.2 Mattilsynet

Mattilsynet er et statlig forvaltningsorgan med ansvar for å fremme blant annet dyrehelse og etisk forsvarlig hold av dyr [25]. Dette arbeidet utføres ved å veilede om regelverk, formidle informasjon og føre risikobasert tilsyn for å forebygge dyrelidelser og utbrudd av smittsomme sykdommer [25, 26]. Mattilsynet får tilsendt beitelagets rapport over oppsynsturene fra Statsforvalteren og gjennomfører en risikovurdering av dyrevelferden på utmarksbeite [27] (flere detaljer om risikoklassene er beskrevet i kapt. 3.3.3). Dersom Mattilsynet konkluderer med at dyreeier ikke tilfredsstiller kravene for god dyrevelferd, kan Mattilsynet kreve at dyreeier iverksetter risikoreduserende tiltak som deriblant økt tilsyn [27]. Retter ikke dyreholderen seg etter Mattilsynets pålegg vil Mattilsynet trappe opp bruken av virkemidler, som tvangsmulkt, overtredelsesgebyrer eller å gjennomføre tiltak på eierens regning [27]. I de mest alvorlige tilfellene kan Mattilsynet ta dyrene i midlertidig forvaring eller omplassere dem, avvikle dyreholdet, nekte den ansvarlige å drive aktiviteter som har med dyr å gjøre eller anmelde forholdet til politiet [28, s.4].

3.2.3 Statens Naturoppsyn

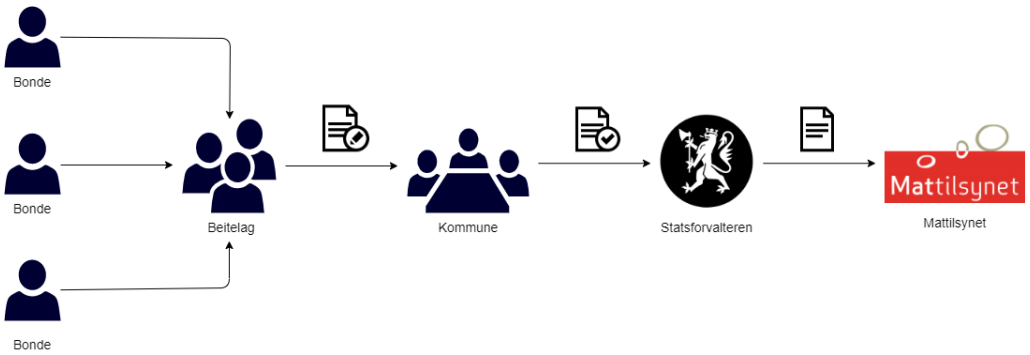
Statens naturoppsyn (SNO) er en avdeling i Miljødirektoratet og opererer som miljøforvaltningens operative feltorgan [29]. Hvis den tilsynsansvarlige finner et skadd

eller dødt dyr og mistenker at det er et fredet rovvilt som står bak, må eieren av dette dyret selv ta kontakt og vise kadaveret til SNO [30, s.18]. SNOs rovviltkontakter vil vurdere kadaveret og konkludere om skaden skyldes rovvilt eller ikke. Alle tap av husdyr erstattes fullt ut etter innsendt søknad hvis det er påvist at fredet rovvilt har forårsaket tapet [30, s.18]. Det gis også erstatning når det er sannsynlighetsovervekt for at tapet skyldes rovvilt [30, s.18]. For å komme fram til en konklusjon om tapet skyldes rovvilt, vil Statsforvalteren bruke opplysningene om beitelagets besetning og drift fra søknaden samt innhente kunnskap om rovviltbestandene og tapsforholdene i området fra kommune/landbrukskontoret og Mattilsynet [30, s.18].

Figur 3.2: Informasjonsflyt ved erstatning for sau drept av fredet rovvilt

3.2.4 Informasjonsflyt ved innsending av oppsynsturrapporter

Dersom et beitelag ønsker å søke om tilskudd til produksjon og drift av beitelaget, må de lage en samlet rapport over alle oppsynsturene som er gjennomført over perioden der dyrene er på utmarksbeite med nødvendig informasjon [23]. Med nødvendig informasjon menes opplysninger som antall døde og skadde dyr, funn eller observasjoner fra rovvilt, beiteforhold, kartinformasjon der observasjonen fant sted osv. Denne rapporten blir sendt til den aktuelle kommunen. Norsk Sau og Geit (NSG) har lagt ut et forslag til et slikt rapportskjema på deres nettsider, se vedlegg B. Etter at kommunen har godkjent rapporten, blir den sendt videre til Statsforvalteren som har ansvaret for å forvalte produksjonstilskuddet og andre relevante tilskudd. Statsforvalteren videresender også oppsynsrapportene til Mattilsynet, som vil gjennomføre en risikovurdering av dyrenes velferd på beitet. Denne flyten er visualisert i figur 3.3 under.



Figur 3.3: Informasjonsflyt mellom sauebønder og involverte myndigheter for innsending av oppsynsturrapporter

3.3 Krav fra myndighetene

Denne delen går gjennom hvilke krav som stilles til bøndene i forbindelse med drift av småfe.

3.3.1 Oppfølging

For å beskytte småfe fra unødig lidelse og død, er bønder lovpålagt å føre tilsyn på utmarksbeite minst én gang i uken [7]. I områder med kjent forekomst av rovvilt må tilsynsfrekvensen være hyppigere enn én gang i uken [7].

3.3.2 Øremerking

Småfe på beitet skal merkes med øremerker som er godkjent av Mattilsynet [31]. Sau skal merkes med både elektronisk og visuelt merke 30 dager etter fødsel [32]. Disse elektroniske øremerkene kan deretter leses av med en elektronisk håndleser for å lese av data [33, s.48]. Følgende informasjon skal være med i øremerket for at dyret skal kunne identifiseres [31]:

- Mattilsynet: MT.
- Nasjonalitetsidentifikasjon: NO.
- Dyreholdets spesielle identitetsnummer tildelt av Mattilsynet: 7 siffer.
- Individnummer: 5 siffer der første siffer er fødselsårets siste siffer og de følgende siffer er dyrets individ nummer.

Figuren under viser et eksempel på et øremerke for sau:

Figur 3.4: Øremerke for sau

Bildekilde: [34]

3.3.3 Dyrevelferd

For å sikre god helse og trivsel blant småfe samt sikre at det tas hensyn til dyrenes naturlige behov, er bønder lovpålagt å holde småfe på egnet utmarksbeite i minst 16 uker hvert år [31]. Med utmarksbeite menes beite i naturlig vill vegetasjon, skog og fjellterreng som ikke blir kultivert eller gjødslet [33, s.55]. Ansvaret for dyrenes velferd på utmarksbeite ligger hos dyreeier [27]. Mattilsynet vil sørge for at dyreholdet følges i henhold til dyrevelferdsloven; forskriften for velferd hos småfe og produksjonsdyr [27]. Dyreeier og andre som utfører tilsyn har meldeplikt dersom det oppdages smitteutbrudd, rovvilt eller andre årsaker til lidelse eller død blant dyrene på utmarksbeite. Mattilsynet opererer med tre risikoklasser for dyr på utmarksbeite [27, s.6]:

- **Lav risiko**
  Tap av under 2% for søyer, 6% for lam og 4% på besetningsnivå ligger innenfor det akseptable, men det forutsettes at det arbeides for å redusere totaltapet på sikt.
- **Middels risiko**
  Tap mellom 2-6% på søyer, 6-15% på lam og mellom 4–10% på besetningsnivå ligger innenfor et område hvor det forventes at forebyggende tiltak planlegges og gjennomføres.
- **Høy risiko**
  Tap over 6% på søyer, 15% på lam og 10% på besetningsnivå er i utgangspunktet uakseptabelt dyrevelferdsmessig sett. Forebyggende tiltak må planlegges og gjennomføres.

Mattilsynet vil vurdere risikonivået for hver enkelt beitelag ut i fra lokal kunnskap om beiteforholdene fra regionale representanter fra Organisert Beitebruk, Norsk institutt for bioøkonomi (Nibio) og søknadene for produksjonstilskudd og erstatning for tap av sau på beite [27]. Oversikt over estimert rovviltbestand og dokumenterte tap til rovvilt blir også hentet fra Rovbase [27].

3.4 Gjennomføring av tilsyn

I dag er det vanlig at flere bønder fra nærliggende områder går sammen og oppretter et beitelag der de samarbeider om tilsyn, sanking og andre aktiviteter knyttet til dyrehold på utmarksbeite [33, s.57]. I 2017 var ca. 74% av all sau som ble sluppet på utmarksbeite inkludert i et organisert beitelag [35]. Beitelag er et av tiltakene i ordningen kalt Organisert Beitebruk (OBB) som ble opprettet i 1970 som et samarbeid mellom Landbruksdepartementet og NSG [35]. Hovedmålsettingen for OBB er å legge til rette for en mer rasjonell utnytting av utmarksbeitene og redusere tapet av dyr på beite til et minimum [35]. I 2003 ble det innført et krav om at beitelag må registreres i enhetsregisteret i Brønnøysund for å være berettiget tilskudd [35].

3.4.1 Før utslipp

NSG har kommet ut med en offisiell anbefaling om at søyer som skal slippes løs på utmarksbeite bør merkes med bjelleslips [36]. Et bjelleslips er et merke som indikerer antall lam som er tilknyttet en søye med fargekoder, se figur 3.5 under. Det gjør det

mulig å for personer på oppsynsturer å få oversikt over dyretall på utmarksbeite. Det kan også lønne seg med bjelleslips dersom ens saueflokk blander seg med en annens besetning som ikke benytter seg av bjelleslips [37, s.27].

Figur 3.5: Bjelleslips for sau

Bildekilde: [36]

NSG anbefaler å bruke samme fargekoding for slips som for fostertelling, som er når det undersøkes antall fostre i en drektig søye med ultralyd [38]. Fargekodingen kan variere fra landsdel til landsdel. Masteroppgaven vil ta utgangspunkt i fargekodene som brukes i Oppdalområdet etter professor Hvasshovds erfaring med Søndre Trollheimen Beitelag som er følgende:

- Blått slips: 0 lam.
- Grønt slips: 1 lam.
- Gult slips: 2 lam.
- Rødt slips: 3 lam.

Det er ikke alle bønder som markerer at en søye ikke har lam og vil da bare unnlate å markere søyen med bjelleslips.

3.4.2 På utmarksbeite

Det er ikke en fastsatt dato for når småfe blir sluppet til utmarksbeite, men det skjer som oftest kort tid etter lammesesongen som forekommer rundt april og mai [39]. Dyrevelferdsloven krever at samtlige dyr har god helsetilstand og at lam er sammen med mordyr og i stand til å holde følge med mora på beitet [31]. Dyreeier må derfor selv vurdere hva som er best slipptidspunkt for sine dyr ut i fra deres alder, tilstand og forholdene på beiteområdet [40]. Slipptidspunktet må være sent nok til det er nok beite som gir god nok næring slik at lammene får en jevn og god tilvekst, men tidlig nok til at sauene kan sortere ut de gode beiteplantene mens de er unge og næringsrike [41, s.8]. Hvis beiteområde inneholder for eksempel høy vannføring i bekker, snøfonner o.l, må lammene være tilsvarende robuste [40].

Selve gjennomføringen av oppsynsturene vil naturlig nok variere fra bonde til bonde. Turene kan gjennomføres enten av den enkelte bonde eller gjennom organiserte beitelag. Heretter vil personer som utfører oppsynsturer bli referert til som tilsynsansvarlig. Per i dag utføres oppsynsturer ved at den tilsynsansvarlige, enten manuelt

eller ved hjelp av digitale verktøy (eksempler på slike verktøy presenteres i kapittel 4.1.1), først sporer opp dyreflokken. Ettersom dyrene beveger seg raskt over store områder og gjerne i ulendt terreng, må den tilsynsansvarlige som regel observere dyrene på avstand med kikkert og deretter notere informasjon som antall og tilstand med penn og papir. Å registrere riktig antall kan fort bli en utfordring siden sauene gjerne beveger seg mens den tilsynsansvarlige ser vekk ifra kikkerten for å notere. De store avstandene fører også til at det kan være vanskelig å få øye på øremerker eller bjelleslips, selv med kikkert. I tillegg til antall sauer på beitet, vil dato, rute for oppsynstur og opplysninger som skadde dyr, årsak, observasjon av rovvilt og beiteforhold bli registrert. Ifølge professor Hvasshovds erfaring med oppsynsturer, pleier det å ta ca. 2-3 timer, men det kan ta opptil 10 timer dersom det er vanskelig å spore opp hele dyreflokken.

3.4.3 Sanking

I følge *Forskrift om velferd for småfe* §27 [31] har dyreeier ansvar for at småfe på utmarksbeite hentes hjem i god tid før det ventes frost eller snøfall om høsten . Ifølge data fra Organisert Beitebruk pleier nedsankingen av sau å foregå i september [42, s.8]. Tidlig innsanking av dyrene i august kan forekomme som et tiltak for å forebygge tap av småfe til rovvilt, da særlig jerv. Over 80 % av skader på sau og lam utført av jerv blir påvist etter 1. august, og nesten halvparten av disse skjer etter 1. september [43]. Dersom en sauebonde velger å gjennomføre tidlig innsanking av dyrene, skal bonden bli kompensert for økningen av kostnadene for kraft- og grovfôr som følge av at dyrene må flyttes til innmarksbeite [42, s.10]. Selve nedsanking av sau kan være utfordrende ettersom det kan være vanskelig å finne hele dyreflokken på de store beiteområdene. Hvis et beitelag har sluppet dyrene samlet, må dyrene også sorteres etter nedsankingen slik at dyrene drar til riktig bonde [44].

3.5 Oppsummering

Det er lovpålagt for norske sauebønder å slippe sau på utmarksbeite i minst 16 uker årlig. Hvert år mister 3-7% av sauer sluppet på utmarksbeite livet grunnet sykdom, ulykker eller rovdyr. Selv om tallene på tap av sau på beite er i nedgående trend, arbeides det med å senke tapene ytterligere både for å minimere dyrenes lidelser samt de økonomiske tapene det forårsaker. Dette arbeides med både fra sauebøndene og beitelagenes side, og fra offisielle myndigheter som Statsforvalteren, Mattilsynet og Statens Naturoppsynet. Et av tiltakene for å redusere tap er å utføre ukentlige tilsynsturer der dyreflokkens antall og tilstand sjekkes, samt om det er rovvilt tilstede i nærområdet. Tilsynet gjennomføres ved at den personen som er tilsynsansvarlig for beitelaget sporer opp dyreflokken og registrerer relevant informasjon som videresendes til Statsforvalteren og Mattilsynet. De har ansvaret for å vurdere om det er behov for ytterligere tiltak for å sikre velferden til dyrene eller gi erstatning for tapte dyr.

Kapittel 4

Inspirasjon og tidligere arbeid

I dette kapittelet vil eksisterende digitale løsninger som brukes i sammenhenger relevant til sporing og registrering gjennomgås.

4.1 Eksisterende løsninger

I dag finnes det ulike digitale løsninger som forenkler prosessen med tilsyn på utmarksbeite. Disse løsningene gjør en av to ting; sporer dyr på utmarksbeitet med sporingsbrikker eller tilbyr et digitalt system som forenkler manuelt tilsyn. Flere slike løsninger blir presentert og undersøkt i dette kapittelet.

4.1.1 Elektronisk sporing av dyr

Utviklingen av teknologi knyttet til småfehold har utviklet seg raskt de siste ti årene for å gjøre det enklere for bonden å ha kontroll på småfe på beite [33, s.49-50]. For teknologi brukt utendørs er automatisk sporing av dyrene med såkalte radiobjeller det mest brukte blant bønder [33, s.50]. Slike radiobjeller plasseres rundt halsen på sauen når den er på beite og gir signaler om dyrenes lokasjon til bonden ved hjelp av GPS eller mobilnettet [45]. De største aktørene i det norske markedet for radiobjeller er *Telespor* [46] og *Findmy* [47]. De seneste årene har disse aktørene har fått konkurranse fra nye aktører som blant annet *Smartbjella* [48].

Figur 4.1: Sauer utstyrt med radiobjeller fra FindMy

Bildekilde: [49]

Telespor

Telespor har vært på markedet siden 2004 med elektronisk sporing av husdyr på beite med *Radiobjella* som benytter seg av GPS lokalisering [50, s.69] [51]. Tidligere generasjoner av radiobjella brukte GSM/GPRS, det vil si 2G mobilnettet, for å motta GPS posisjon fra satellitter til Telespors servere som deretter videresendte informasjonen til kundens brukerportal [52]. Dagens fjerdegenerasjons radiobjelle benytter seg av nyere NarrowBand-Iot (NB-IoT) og LTE-M, som muliggjør at sensorer og andre enheter kan kommunisere med 4G-nettet raskere og med svært lav energibruk [53, 51]. Telespor bruker Telenors IoT-dekning på 4G-nettet. Dette gjør at radiobjellene er avhengig av at det er dekning i området for å kunne kommunisere med bonden. Radiobjellene er også utstyr med bevegelsessensorer som kan utløse tre alarmer [51] dersom:

- Dyret har ikke beveget seg de siste tre timene.
- Dyret har vært på samme posisjon i en lengre periode.
- Radiobjella har ikke klart å sende sin posisjon for de siste to innrapporteringene.

Sendeintervall og alarmsensitivitet kan endres på når som helst av brukeren [51]. Selve bjellene er utstyrt med utskiftbare batteri som det anbefales å skifte etter hver sesong [51]. Telespor bruker en betalingsmodell der brukere må betale en sum for radiobjellene og et sesongbasert abonnement som dekker både mobildatatrafikken, tilgang til brukerportalen og et batteri [54, 55]. Per dags dato er prisen på en fjerdegenerasjons radiobjelle 1124 kr og sesongabonnement 124 kr [55].

FindMy

FindMy, tidligere kalt FindMySheep, ble skapt av sauebønder i 2009 som ønsket elektronisk sporingsutstyr som ikke var avhengig av mobilnettet [56]. FindMys e-bjeller er bygd rundt lavbane satellitteknologi og er dermed ikke avhengig av verken GSM eller NB-IoT for å fungere og vil i praksis fungere over hele verden [57]. Det vil si at e-bjella har dekning så lenge den har fri sikt til himmelen. E-bjella har også en funksjon der den vil prøve å sende signaler flere ganger for å få tilfredstillende resultat dersom e-bjella er i et område med utfordrende terreng og topografi [58]. Alle e-bjellene kan settes opp med egendefinerte meldingsplaner som bestemmer hvor ofte det kommer oppdateringer om dyrets posisjon i løpet av en dag [59]. I tillegg til elektronisk sporing har e-bjella Model 2 funksjoner som sporingsdata og tilvekstkart fra Sauekontrollen, søk med Bluetooth, geofencing, urovarsel som detekterer unormal adferd i flokken og sporlogg som samler posisjoner e-bjella har sendt i en valgt periode for å gi innsikt beitemønster [60]. FindMy anbefaler selv om det spores minst 25% av dyreflokken for å få god oversikt over dyrene [49]. Ca. 40 000 e-bjeller fra FindMy er i bruk per dags dato [60]. Per i dag koster én enkelt e-bjelle 1849 kr i tillegg til en årlig brukeravgift på 229 kr per bjelle [61]. Lik som Telespor har også e-bjella til FindMy utskiftbarebatteri, men disse skal vare to til tre beitesesonger og koster 99 kr per stykk [62].

Smartbjella

Smartbjella/Smartbells ble lansert i 2019 etter at ansatte ved morselskapet StalkIT [63] oppdaget at deres teknologi for å spore opp konteinere kunne være nyttig for å spore sauer på beite [64]. Smartbjella var den første aktøren som benyttet seg av IoT-teknologi innenfor dyresporing etter å ha inngått avtaler med Telenor og Telia om å bruke deres NB-IoT nettverk [65]. Det benyttes også GNSS for nøyaktig posisjonering [66]. Smartbjella nyeste versjon 2, som er ute for pre-salg i 2021, inkluderer dødsalarm, frekvensstyring av signalene fra bjellen, historisk bevegelsesmønster, temperaturmåler, Bluetooth og mulighet til å sette opp geofence [66]. Smartbjella skal være vedlikeholdfri per sesong [66]. Ved normale værforhold og med et rapporteringsintervall på 24 timer skal batteriet i Smartbjella holde opptil 15 år [66]. Prisen for Smartbjella 2 ligger på 949 kr per enhet og abonnement på en sesongbruk av smartbjella ligger på 100 kr per enhet [67]. Det er også mulig å leie radiobjeller [64]. Per dags dato er det 18000 radiobjeller fra Smartbjella i bruk i Norge [48].

Evaluering

Alle bedriftene som er nevnt i forrige delkapittel er i markedet for elektronisk sporing av dyr på utmarksbeite og arbeider med å gjøre det lettere for bønder å lokalisere dyreflokken og få tilbakemeldinger om dyrenes tilstand. Teknologiene som utnyttes for elektronisk sporing er relativt like, spesielt etter at Telespor også byttet over til å bruke NB-IoT og LTE-M, med unntak av FindMy som benytter seg av lavbane satellitter. Dette er en fordel her i Norge der dyrene ofte ferdes på øde steder uten mobildekning. Alle aktørene tilbyr også en form for alarmfunksjonalitet som gir tilbakemelding dersom et dyr har vært inaktiv over en lengre periode. Betalingsmodellene er også tilnærmet identiske, med en pris for selve radiobjellene og brukeravgift/abonnement på servicetjenester, samt tilleggskostnader for batteri som eventuelt må skiftes ut. Her skiller Smartbjella seg ut ved at batteriene har mye

lengre levetid. Smartbjella gir mer fleksibilitet da det er mulig å leie bjellene ved behov [64]. Prismessig ligger Telespor og FindMy på rundt 1500-2000 kr for en hel pakke inkludert bjelle og brukeravgift, mens Smartbjella er rimeligere med enheter og brukeravgift på rundt 1000 kr.

Store kostnader knyttet til utstyr og vedlikehold er nettopp en av grunnen til at mange bønder tidligere ikke ønsket å benytte seg av radiobjeller. De siste årene med nyere, mer stabil teknologi og konkurranse fra flere aktører på markedet som presser prisene ned, er det nå stadig flere bønder som benytter seg av radiobjeller. I rovdyrutsatte områder kan også beitelag søke om tilskudd til konfliktdempende tiltak som deriblant elektronisk overvåkningsutstyr, der Statsforvalteren kan dekke deler av kostnadene for investeringen. Dette har gjort at flere bønder fikk testet radiobjeller i praksis og senere begynt å benytte seg av dem [50, s.61-66 66] [68].

Fra 2009 til 2012 ble det utført et nasjonalt beiteprosjekt på vegne av Statens landbruksforvaltning (nå Landsbruksdirektoratet) med formål å få bedre sauehold med mindre tap av dyr på beite [69]. Utprøvingen av radiobjeller ble evaluert av Trøndelag Forskning og Utvikling (TFOU) [69, s.8]. Rapporten konkluderte med at det antakelig er en forebyggende tapsreduserende effekt med bruk av radiobjeller. Den viste også at bruken av radiobjeller kunne føre til bedre dyrevelferd ettersom bjellene ga raskere avdekking av unormal adferd blant dyrene samt mer effektiv sanking på høsten [69, s.40]. Bøndene selv rapporterte om økt produktivitet fordi radiobjellene gjorde arbeidet med å lokalisere dyrene i sammenheng med tilsyn og sanking lettere, som igjen førte til høyere tilfredshet blant bøndene og økt motivasjon til å bruke utmarksbeite [69, s.41], [50, s.60]. Tapte dyr med radiobjeller hadde økt gjenfinningsrate og bedre mulighet til å fastslå dødsårsak enn dyr uten radiobjeller [69, s.40].

Selv om overvåkningsteknologien for dyr på utmarksbeite stadig forbedres og prisen for utstyret senkes, er det fortsatt ikke vanlig å sette radiobjeller på alle dyrene, særlig ikke lam. Elektronisk sporing av dyr erstatter ikke manuelt tilsyn på utmarksbeitet, men det er et nyttig verktøy som gjør dette arbeidet lettere å gjennomføre.

4.1.2 Registrering av dyr i utmark

Per dags dato er det ingen aktive applikasjoner som legger til rette for registrering av dyr i utmark, men det finnes løsninger som har vært innom denne ideen tidligere. *Beitesnap* var en applikasjon som tok for seg registrering av tilsyn for husdyr på beite, både for bønder under beitesesongen og for privatpersoner som ønsker å melde fra om beitedyrsobservasjoner i naturen [70]. I tillegg er det gjennomført tidligere masteroppgaver med lignende problemstilling, som for eksempel applikasjonen *Lambo* som ble utviklet i sammenheng med oppgaven *Effektivisering av manuell oppfølging av sau på utmarksbeite* fra 2018 [71]. Det er også flere pågående masteroppgaver under våren 2021 som utforsker samme tema som denne masteroppgaven.

Beitesnap

Beitesnap av Fant AS er en applikasjon som ble lansert i 2017 som et verktøy for å registrere observasjoner av husdyr på beite [70]. Applikasjonen avsluttet driften fra og med 2020 grunnet dårlig økonomi som følge av få brukere [72]. Applikasjonens

målgrupper var beitebrukere som kunne legge inn sitt beiteområde og dyrenes individnummer slik at de kan få meldinger fra andre som observerer skadde eller døde dyr innenfor bondens beiteområde [70]. Alle registreringene fra beitesesongen ble samlet og utformet til en rapport som kunne sendes inn til myndighetene [70]. *Beitesnap* var også koblet opp mot Norgeskart [73], slik at det var mulig å spore oppsynsturene med GPS [72]. Kartet kunne lastes ned på forhånd for offline bruk [70]. For å få tilgang til en Beitebruker-konto på *Beitesnap*, var det en årlig avgift på 1200 kr + MVA [70]. Privatpersoner som ønsket å sende inn sine observasjonen kunne opprette en gratis bruker som gav dem mulighet til å sende inn bilder med informasjon om dyr de møtte på i utmarka [70]. Dersom observasjonen er innenfor et registrert beiteområde vil den aktuelle bonden få beskjed om det [70].

Beitesnap har mye lik funksjonalitet som *Sauron*, men hvordan informasjonen registreres er nokså ulik. I *Beitesnap* registreres et bilde av observasjonen, hvilket dyreslag som er registrert, om dyret var skadd eller dødt og eventuelt individnummeret til dyret [74]. All annen informasjon, som totalt antall dyr, antall søyer og lam, farge på dyrene, øremerker eller bjelleslips må dermed skrives i fritekst. Å observere dyrene på lang avstand for å så måtte taste inn all informasjonen kan være utfordrende, samtidig som det gjør at innrapporteringene blir mindre strukturerte.

Figur 4.2: Skjermbilde av *Beitesnaps* registreringsside

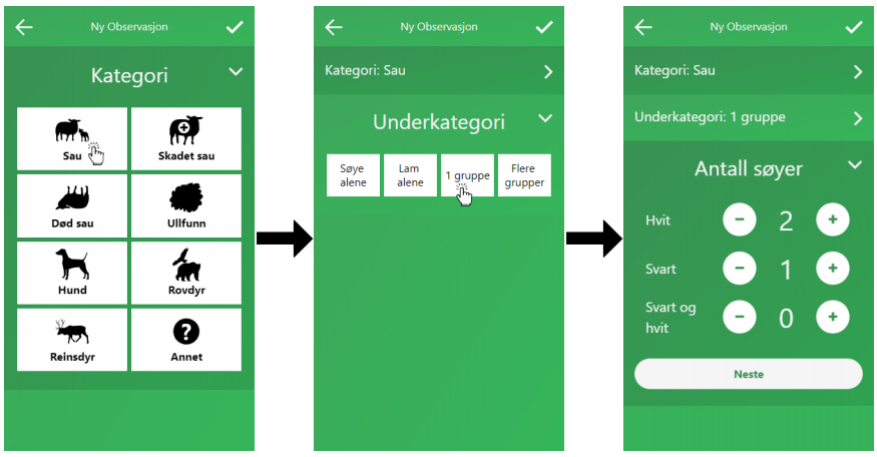
Bildekilde: [74]

Tidligere og pågående masteroppgaver

Det er flere gjennomførte og pågående masteroppgaver fra NTNU som har arbeidet med å effektivisere tilsyn av sauer på utmarksbeite. Denne masteroppgaven har tatt inspirasjon fra masteroppgaven *Effektivisering av manuell oppfølging av sau på utmarksbeite* [71] skrevet av Stian Dysthe og Andreas Kjerstad. Deres mål for prosjektet var [71, s.13]:

Utvikle et produkt som gjør det lett og effektivt å registrere detaljert, strukturert, lokasjonsbasert informasjon om sau på beite, samtidig som det møter behovet til bøndene og krav fra myndighetene.

I den sammenheng utviklet de applikasjonen *Lambo*, som har en del fellestrekk med både *Beitesnap* og *Sauron*. *Lambo* har funksjonalitet som deriblant online og offline kart med GPS-sporing av brukerens rute og registrering av observasjoner av både husdyr og rovdyr [71]. I motsetning til *Beitesnap* har *Lambo* en mer detaljert innrapportering av observasjoner, der brukeren må gå gjennom et registreringsskjema for å registrere en observasjon istedetfor å skrive det inn som fritekst. Brukeren vil først bli spurt om observasjonen gjelder sau, skadet sau, død sau, ullfunn, hund, rovdyr, reinsdyr eller annet, og deretter få spørsmål om relevante underkategorier som antall, farge osv [71, s.84-85]. Måten å registrere informasjon er det som vil skille *Sauron* fra *Lambo*, ettersom *Sauron* vil ha funksjonalitet som gjør det mulig å registrere observasjonene uten å se på mobilskjermen.



Figur 4.3: Skjermbilde fra *Lambo*s registreringsskjema

Bildekilde: [71]

4.2 Oppsummering

Det brukes stadig mer og mer digital teknologi for å assistere bønder i deres arbeid med dyrehold på utmarksbeite. Et av de mer brukte digitale verktøyene som bistår i oppsynsturer er digitale sporingsutstyr som radiobjeller som settes på dyrene og sporer deres lokasjon. Per i dag er det minst tre aktører i Norge som tilbyr radiobjeller; Telespor, FindMy og Smartbjella. Disse aktørene benytter seg av GPS-teknologi som enten NB-IoT-nettet eller lavbane satellitteknologi for sporing. Radiobjellene har

i tillegg andre funksjoner som blant annet dødsalarm, tilpasning av innrapportering og geofencing. Tidligere prosjekter som har evaluert bruken av elektronisk overvåkning av dyr på utmarkbeite har konkludert med at radiobjellene kan forebygge tap på beitet ved at farlige situasjoner raskere blir avdekket og samtidig fører til mer effektiv sanking av dyrene på høsten. Likevel vil ikke radiobjeller erstatte manuelle oppsynsturer med det første ettersom norske myndigheter krever at oppsynsturer skal bli utført manuelt hver uke. Applikasjoner som *Beitesnap* og *Lambo*, har blitt brukt som inspirasjon. En av forskjellene mellom *Sauron* og disse applikasjonene er at de mangler funksjonalitet som muliggjør registrering av sau uten å måtte se på mobilskjermen.

Kapittel 5

Fordypningsprosjekt

Denne masteroppgaven bygger videre på arbeidet gjort under fordypningsprosjektet *Tilsyn med Sau På Beite* [1] som ble gjennomført høsten 2020. Dette underkapittelet går gjennom arbeidet som ble gjort der og legger fram resultatene fra prosjektet.

5.1 Problemstilling og mål

Mens denne masteroppgaven går ut på å utvikle et helhetlig produkt for å bistå tilsynsansvarlige på oppsynstur, fokuserte fordypningsprosjektet spesifikt på selve brukergrensesnittet for registreringen av sau. Målet med fordypningsprosjektet var [1, s. 3]:

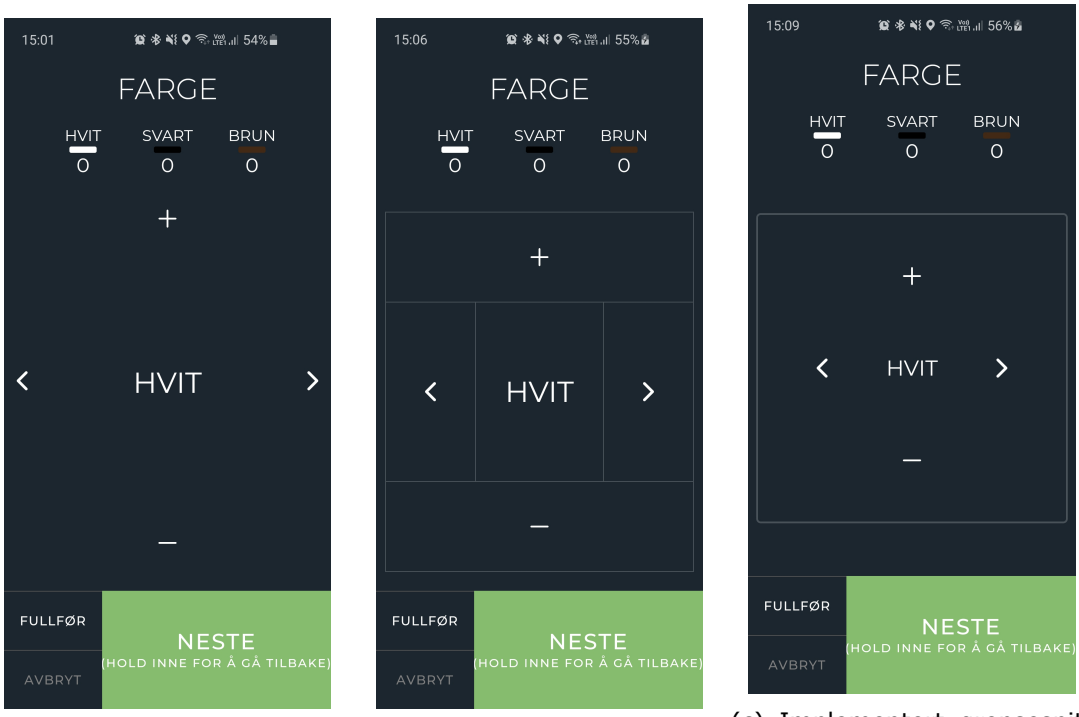
Designe og utvikle en applikasjon som muliggjør effektiv registrering av sauer på utmarksbeite i situasjoner der brukeren kontinuerlig benytter seg av kikkert og dermed ikke kan se på skjermen.

5.2 Metode

Fordypningsprosjektet startet med et litteratursøk inn i eksisterende teknologi og løsninger på brukergrensesnitt for berøringsskjermer laget for blind interaksjon. Deretter ble det designet og utviklet tre ulike grensnitt som brukte forskjellige interaksjonsmetoder for å registrere informasjon om sauflokker blindt. Brukertester på alle de tre grensesnittene ble så utført. Det ble samlet inn informasjon både underveis i brukertestene men også i etterkant i form av at spørreskjema som brukeren ble bedt om å fylle ut. Da brukertestene var gjennomførte ble det utført kvantitativ og kvalitativ analyse på resultatene.

5.3 Design og implementasjon

Figur 5.1 viser skjermbilder fra registreringsgrensesnittet for alle de tre ulike prototypene. Her har brukeren navigert seg til kategori *Farge* og holder nå på med registrering av sau av hvit farge.



(a) Implementert grensesnitt med gester

(b) Implementert grensesnitt med virtuelle knapper

(c) Implementert grensesnitt for kombinasjon av gester og virtuelle knapper

Figur 5.1: Implementerte opptellingsgrensesnitt for første prototype

Bildekilde: [1, s.46]

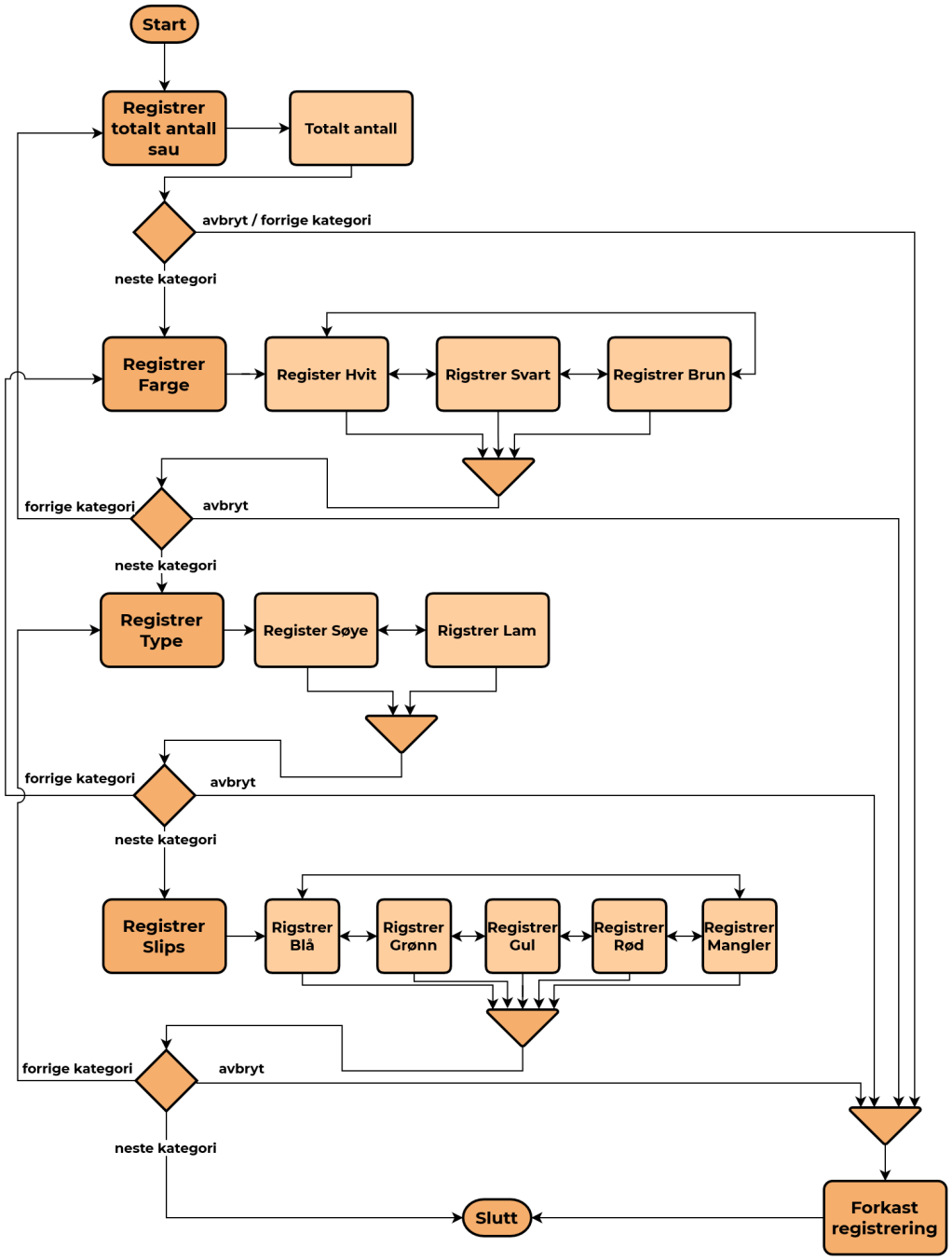
Grensesnittene i figur 5.1 er laget for å teste to ulike interaksjonsmetoder, gester og virtuelle knapper samt en kombinasjon av disse. Med gester menes interaksjoner med berøringsskjermer som sveiping, hvor man drar en finger et stykke over skjermen. Dette i motsetning til virtuelle knapper hvor man trykker eller tapper på en knapp laget i grensesnittet. Resultatene fra litteratursøket i fordypningsoppgaven viste at [1, s.9]:

Tidligere undersøkelser av bruk av gester og virtuelle knapper der brukerene har begrenset mulighet for å se på skjermen tyder på at gester som er enkle å utføre fører til mindre feil enn virtuelle knapper. Derimot kan virtuelle knapper være raskere å bruke avhenging av implementasjon.

De kommende forklaringene av de ulike protypene tar utgangspunkt i grensesnittet for registrering av farge. Prototypen i figur 5.1a viser grensesnittet laget for interaksjon med gester. Gesten som har blitt implementert er *sveiping*. Her sveiper brukeren opp eller ned for å henholdsvis øke eller senke antallet registrert for en farge. For å bytte mellom ulike farger kan brukerene sveipe sidelengs både til høyre og venstre for å bla mellom de ulike fargene. Prototypen i figur 5.1b har den samme funksjonaliteten bare at øking og senking av antall, samt veksling mellom ulike farger gjøres ved å trykke på store virtuelle knapper laget på skjermen. En kombinasjon av disse ulike interaksjonsmetodene har blitt implementert i prototypen i figur 5.1c. For å øke antall registrerte for en farge kan brukeren trykke i den øvre halvdelen av den markerte firkanten. For å senke antallet trykkes det i nedre halvdel. I hele firkanten kan

sidelengs sveiping brukes for å bla mellom ulike farger.

Likt for alle grensesnittene er det at de tar i bruk både haptisk tilbakemelding og lydtilbakemeldinger i form av tekst-til-tale for å gi brukeren henholdsvis fysisk og auditiv respons på utført interaksjon. Grensesnittet for registrering av saueflokker består av flere steg hvor forskjellig informasjon om saueflokken skal registreres. Figur 5.2 viser et flytdiagram for hvordan registrering av forskjellig informasjon for en saueflokk foregår.



Figur 5.2: Programflyt for en registrering

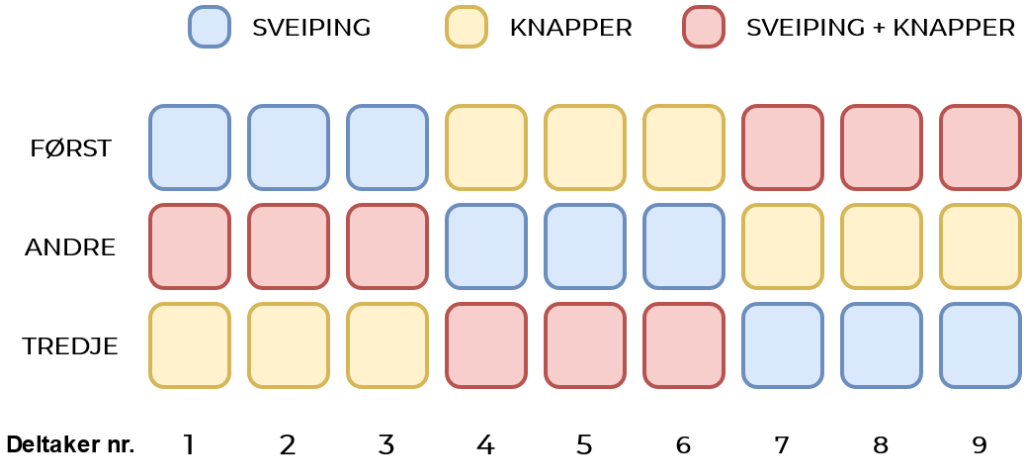
Bildekilde: [1, s.26]

Da brukeren ikke har mulighet til å se på skjermen under en registrering vil det være praktisk umulig å holde kontroll på hva som har blitt gjort og hvor brukeren befinner seg i grensesnittet uten noen form for auditiv respons. Det utviklede grensesnittet tar i bruk tekst-til-tale for å lese opp relevant informasjon til brukeren under en registrering. Et eksempel på dette kan være når brukeren beveger seg fra "Registrer Totalt antall sau" til "Registrer Farge". Da vil applikasjonen lese opp denne teksten til brukeren; "Registrer Farge, 0 Hvit sau". Dette forteller brukeren at applikasjonen nå er klar for registrering av *Farge* og at det til nå har blitt registert 0 hvite sauer. Når brukeren øker eller senker antallet for en spesifikk farge vil applikasjonen lese opp antallet i tillegg til hvilken farge det er. Det samme gjelder for når brukeren bytter mellom ulike farger.

For at brukeren skal få bekreftelse på at ulike handlinger har blitt utført i blinde har det blitt implementert haptisk tilbakemelding for alle gester og virtuelle knapper som brukes under en blind registrering. "Haptisk tilbakemelding er når elektroniske enheter gir informasjon til brukeren gjennom brukergrensesnittet ved å gjenskape berøring, som regel ved å avgi avanserte vibrasjonsmønstre eller andre bølgeformer" [1, s.8]. På den måten kan brukeren være sikker på at for eksempel et trykk på en knapp på skjermen har blitt registrert, selv om brukeren ikke har mulighet til å se på den.

5.4 Brukertestene

For å finne ut hvilke av de implementerte prototypene som fungerte best ble det utført brukertester på ni ulike brukere. Hver bruker testet hvert brukergrensesnitt i en forhåndsbestemt rekkefølge. Rekkefølgen var satt opp slik at alle de ulike grensesnittene ble testet like mange ganger i hver posisjon (se figur 5.3).



Figur 5.3: Gjennomføringsrekkefølge for testing av de forskjellige prototypene

Bildekilde: [1, s.54]

Brukertestene gikk ut på at deltakerne skulle utføre to oppgaver. I oppgavene fikk deltakeren se et bilde av en saueflokke hvor det skulle registreres ned så mye informasjon som mulig om flokken. Figur 5.4 viser et eksempel på hvordan Oppgave 1

kunne se ut. Oppgave 2 var lik Oppgave 1, bare at sauene i flokken var animert til å bevege seg rundt på skjermen for å simulere et mer reelt scenario. Før hver test av hver prototype fikk deltakeren en innføring i hvordan grensenittet fungerte. I tillegg fikk testdeltakeren 1 minutt til å teste ut grensenittet på egen hånd.

Figur 5.4: Skjermbilde av Oppgave 1

Bildekilde: [1, s.38]

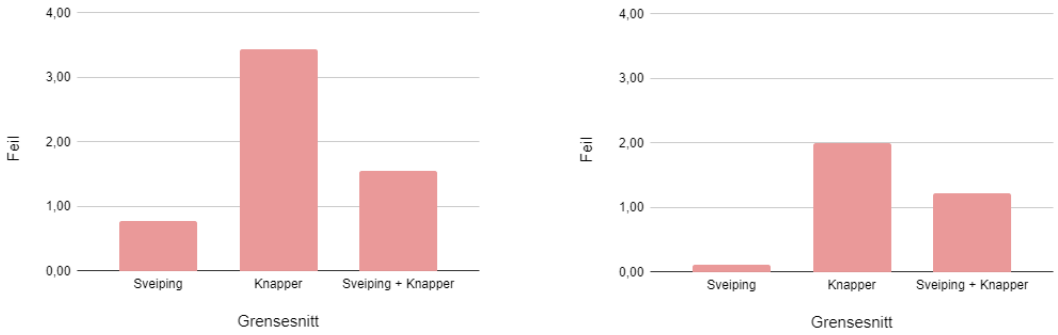
For at deltakeren ikke skulle ha mulighet til å se på skjermen under testen ble skjermen dekket til med en lue. Dette kan du se i figur 5.5. For hver brukertest ble tiden brukt på å fullføre hver oppgave registert. Deltakeren ble også bedt om å fortelle når det ble gjort feil. Eksempler på feil kunne være at deltakeren bommet på en knapp, at noe uønsket ble registrert eller at det ble hoppet over deler av oppgaven. Det ble også notert ned eventuelle tilbakemeldinger og observasjoner som ble gjort under testen. Når testen var fullført måtte deltakeren fylle ut et skjema for å gi tilbakemeldinger på hver av de tre prototypene.

Figur 5.5: Eksempel på gjennomføring av en brukertest

Bildekilde: [1, s.39]

5.5 Resultater

Figur 5.6 viser den gjennomsnittlige feilraten for gjennomføringen av Oppgave 1 og Oppgave 2. For begge oppgavene er det mulig å se at grensesnittet som tar i bruk virtuelle knapper har en tendens til å resultere i høyere feilrate enn de to andre. Den statistiske analysen gjort i fordypningsprosjektet viste at det for Oppgave 2 var en signifikant korrelasjon mellom feilraten og hvilket grensesnitt som ble brukt. Den viste også at det var en betydningsfull forskjell i feilrate mellom brukergrensesnittet som brukte sveiping og det som brukte digitale knapper [1, s.55].

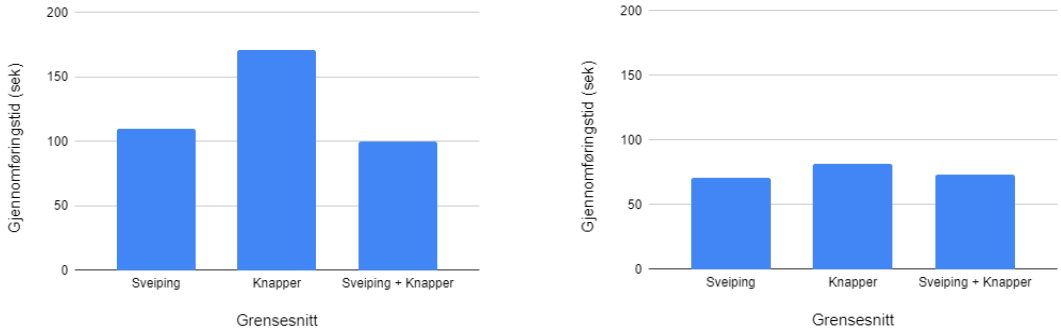


(a) Oppgave 1 (b) Oppgave 2

Figur 5.6: Gjennomsnittlig antall feil per oppgave for de forskjellige grensesnittene

Bildekilde: [1, s.55]

Resultatene for gjennomføringstidene viste at det ikke var noen signifikante forskjeller mellom de ulike prototypene. Det var en tendens til at gjennomføringstiden fikk en generell nedgang fra Oppgave 1 til Oppgave 2 [1, s.56].

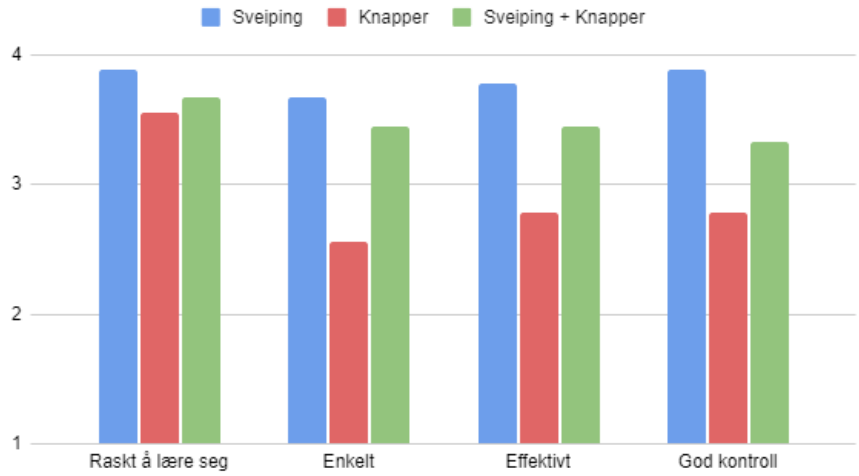


(a) Oppgave 1 (b) Oppgave 2

Figur 5.7: Gjennomsnittlig gjennomføringstid (sekunder) for de forskjellige grensesnittene

Bildekilde: [1, s.56]

Grafene i figur 5.8 viser hvordan deltakerne svarte på tilbakemeldingsskjemaet som ble utfylt etter gjennomført brukertest. Deltakerne hadde her mulighet til å gi en rating fra 1 til 4 fordelt på fire ulike kriterier hvor 4 er den beste. Resultatene viser en tendens til at deltakerne syns *Sveiping* både er enklere å bruke, gir bedre kontroll og er mer effektivt enn de to andre [1, s.56].



Figur 5.8: Gjennomsnittlige tilbakemeldinger for de forskjellige grensesnittene hvor høyere rating betyr at deltakerne likte grensesnittet bedre

Bildekilde: [1, s.57]

I tillegg til at deltakernes uttalelser under brukertesten ble notert var det et fritekstfelt som kunne fylles ut for hver av de tre prototypene i tilbakemeldingsskjemaet. Tilbakemeldingene for *Sveiping* viste at deltakerne generelt fant denne interaksjonsmetoden mest naturlig, men at det kunne bli tungt å utføre i lengden. For *Knapper* var konsensus i tilbakemeldingene at knappene kunne være vanskelig å treffe og at brukerne bommet mye [1, s.57]. Tilbakemeldingene for kombinasjonen av *Sveiping + Knapper* var veldig varierte med lite enighet mellom de forskjellige deltakerne [1, s.58]. En mer generell tilbakemelding som ble gitt var at det kunne vært ønskelig med mulighet for å lese opp informasjonen på skjermen, uten å måtte være nødt til å endre enten antall, kategori eller underkategori [1, s.57].

5.6 Konklusjon

Med bakgrunn i resultatene ble det i fordypningsprosjektet dratt en konklusjon om at grensesnittet som tok i bruk gester i form av sveiping var det som fungerte best for blind registrering av sau. For å hjelpe til med trettheten enkelte testdeltakere følte i tommelen etter en lengre periode med sveiping ble det bestemt å videreutvikle grensenittet. Muligheten for å både sveipe og trykke for å øke eller senke antall i tillegg til sveiping ble lagt til slik at brukeren kan ha mulighet til å gjøre begge deler. Dette ville gjøre det mulig å trykke raskt for å øke antallet om ønskelig, for deretter å kunne sveipe for å gjøre finjusteringer. Det ble også bestemt at funksjonalitet for å lese opp informasjon fra skjermen uten å måtte endre antall, kategori eller underkategori skulle implementeres. I den tenkte løsningen kan brukeren holde inne på skjermen i registreringsgrensesnittet for å lese opp nåværende status [1, s.64].

Del III

Eget bidrag

Visjon, prosess, implementasjon og alt annet i forbindelse med utvikling av applikasjonen *Sauron* blir i denne delen gjennomgått i detalj. Valget av metode for forskningsdelen av prosjektet blir også presentert. Bruksanvisning for å kjøre applikasjonen lokalt finner du i vedlegg A.

Kapittel 6

Visjon

Med utgangspunkt i problemstillingene beskrevet i kapt. 3 og kravspesifikasjonen utviklet i dialog med professor Hvasshovd beskrevet i kapt. 7 har utviklerne utformet en visjon for applikasjonen. Applikasjonen skal være et verktøy for å forenkle prosessen med å utføre tilsyn med sau på beite. Applikasjonen skal ta i bruk stedstjenester for å automatisk kartlegge nedlagt rute under en oppsynstur. Den nedlagte ruten, samt brukerens posisjon, skal vises fram på et kart. For å få tilgang til de mest oppdaterte kartfilene skal tjenesten Norgeskart tas i bruk. Det skal være mulig å laste ned valgte kartutsnitt. Dette vil gjøre det mulig å ta i bruk applikasjonen på steder der det ikke er tilgang til internett.

Gjennom kartgrensesnittet skal brukeren kunne legge til nye observasjoner gjort på en oppsynstur. Observasjonen som legges til plasseres på kartet der brukeren ønsker. Applikasjonen skal i første omgang kunne registrere observasjon av sauer, fredet rovdyr, skadet sau og død sau. Ved registrering av sau skal grensesnittet tilrettelegge blind bruk av mobilen. Dette skal gjøre det mulig for brukeren å kunne se i kikkert og bruke mobiltelefonen med bare én hånd. Ved observasjon av en saueflokk skal brukeren kunne legge inn informasjon om totalt antall sau, antall søyer og lam, antall dyr av hver farge (hvit, svart og brun), antall søyer med fargede slips og hvilket øremerke dyrene har. For registrering av død sau skal det være mulig å ta bilder av kadaveret. Disse bildene brukes i forbindelse med bestemmelser rundt erstatning gjort av norske myndigheter.

For at brukere av *Sauron* innenfor samme beitelag skal ha mulighet til å samhandle skal det implementeres en tjenerløsning som tilbyr ekstern lagring av registrerte oppsynsturer. Brukere av applikasjonen vil få tildelt en egen profil med epost og passord som brukes til å logge inn i applikasjonen. Når brukeren avslutter oppsynsturen vil alle registrerte observasjoner bli lagret i skyen. Alle oppsynsturene som brukeren har tilgang på skal kunne hentes fra skyen og vises på en oversiktlig måte. Brukere innenfor et beitelag vil ha tilgang til hverandres oppsynsturer.

Kapittel 7

Kravspesifikasjon

Dette kapittelet tar for seg kravene som ble utviklet sammen med fungerende produkteier professor Hvasshovd før og under utviklingen av applikasjonen. Kravene er delt i to i form av funksjonelle krav og ikke-funksjonelle krav. Både de funksjonelle og de ikke-funksjonelle kravene er nummerert og organisert inn i undergrupper. Hvert krav har også en prioritet som forteller hvor viktig det har vært å få implementert det spesifikke kravet. Krav som var absolutt nødvendige for at applikasjonen skulle regnes som ferdig ble merket med prioriteringen "Høy". Krav som ikke nødvendigvis måtte implementeres for at applikasjonen skulle regnes som ferdig, men som ville gjøre applikasjonen betydelig bedre er merket med prioriteringen "Middels". Til sist er krav som ikke er nødvendig for applikasjonen i det hele tatt, men som ville vært fint å ha med, merket med prioriteringen "Lav".

7.1 Funksjonelle krav

Tabell 7.1 viser alle de funksjonelle kravene som har blitt utviklet i løpet av prosjektet. Med funksjonelle krav menes krav stilt til funksjonaliteten brukeren skal ha tilgang til under bruk av applikasjonen.

Nr.	Beskrivelse	Prioritet
F1	Brukeren skal ha tilgang til de mest oppdaterte kartbildene via Karverkets tjenester (NorgesKart).	Høy
F1.1	Brukeren skal kunne laste ned et utsnitt av kartet til offline bruk uten internett.	Høy
F1.2	Ved bruk av kart skal brukeren kunne se sin posisjon og en linje over kartlagte bevegelser.	Høy
F1.3	Brukeren skal kunne legge til observasjoner på kartet i form av pins.	Middels
F2	Brukeren skal kunne registrere en ny oppsynstur.	Høy
F2.1	Brukeren skal kunne ha mulighet til å fylle ut nødvendig informasjon om en oppsynstur.	Høy
F2.3	Brukeren skal kunne registrere saueflokker på beitet med informasjon som gjør det mulig å kjenne igjen den spesifikke saueflokken ved et senere tidspunkt.	Høy
F2.3.1	Brukeren skal kunne registrere totalt antall sau i en flokk.	Høy
F2.3.2	Brukeren skal kunne registrere antall av hver farge på sauene i flokken.	Høy
F2.3.3	Brukeren skal kunne registrere antall søyer og lam i flokken.	Høy
F2.3.4	Brukeren skal kunne registrere antall av søyer med slips og farge på slipset.	Høy
F2.3.5	Brukeren skal kunne legge inn øremerking i registreringen.	Høy
F2.3.6	Brukeren skal kunne legge til egne øremerker med navnet på tilhørende bonde og farge på øremerket.	Høy
F2.3.7	Brukeren skal kunne legge til øremerker hvor et øremerker kan ha en eller flere egendefinerte farger.	Middels
F2.3.8	Systemet skal gi brukeren tilbakemelding dersom det forekommer avvik, mellom f.eks. antall registrerte slips og lam, i registreringen.	Middels
F2.4	Registreringer skal lagres eksternt.	Middels
F3	Systemet skal ha innlogging med brukernavn og passord.	Høy
F3.1	Brukeren skal ha egen profil og mulighet til å se og endre på den.	Lav
F4	Brukeren skal kunne registrere forskjellig informasjon om saueflokker uten å måtte være nødt til å se på selve brukergrensesnittet.	Høy
F4.1	Brukeren skal få verbale tilbakemeldinger om utførte handlinger under blind bruk slik at brukeren slipper å se på skjermen.	Høy
F4.2	Brukeren skal få haptisk tilbakemelding i form av vibrasjoner i mobiltelefonen når skjermen trykkes på under blind bruk.	Høy

Table 7.1: Tabell som viser funksjonelle krav for applikasjonen

7.2 Ikke-funksjonelle krav

De ikke-funksjonelle kravene vises i tabell 7.2. Med ikke-funksjonelle krav menes krav til applikasjonen som ikke er knyttet opp til funksjonalitet som brukeren benytter seg av direkte.

Nr.	Beskrivelse	Prioritet
IF1	Applikasjonen skal fungere uten internett.	Høy
IF2	Applikasjonen skal være kryssplattform og fungere på mobile enheter med enten Android og iOS.	Høy
IF3	Applikasjonen skal tillate bruk på inntil 10 timer uten tilgang på strøm.	Lav
IF4	Blind registrering av sau skal være så effektiv som mulig.	Høy

Table 7.2: Tabell som viser ikke-funksjonelle krav for applikasjonen

Kapittel 8

Valg av prosess

Dette kapittelet beskriver og begrunner utviklingsmetodikken og prosessen som ble valgt for utviklingen av applikasjonen *Sauron*. Flere kjente utviklingsmetodikker ble undersøkt og vurdert mot hverandre for å finne den som egnet seg best til denne masteroppgaven.

8.1 Utviklingsmetodikk

Innenfor utvikling av programvaresystemer vil utviklingsmetodikk referere til prosessen med å planlegge, utvikle, teste og distribuere et prosjekt [75] der målet er å lage fungerende programvare. Med andre ord betyr det en bestemt metodikk for å strukturere prosessen og arbeidet for å effektivisere utviklingsprosessen samt implementere et produkt av høyere kvalitet. Hvilken metodikk som har vært fremtredende innen programvareutvikling har endret seg over tid og med utviklingen av nye programmeringspråk, rammeverk, verktøy og omfanget av programvaresystemene. I dag finnes det mange ulike utviklingsmetodikker som har sine styrker og svakheter ut i fra prosjektets mål og omfang. Videre skal to kjente utviklingsmetodikker, *vannfallsmodellen* og *agil utvikling*, beskrives og vurderes som metodikker for prosjektet.

Lenge var den linære vannfallsmodellen den mest populære metodikken. Den brukes fortsatt i dag selv om andre utviklingsmetodikker har blitt mer dominerende de siste årene [76]. Lignende metodikk ble først beskrevet på 1950-tallet, men vannfallsmodellen slik den er kjent i dag ble opprinnelig introdusert av Wintson Royce i 1970 [77, s.329]. I vannfallsmodellen deles prosessen opp i sekvensielle faser som følger etter hverandre (se figur 8.1 under), der én fase må fullføres før man kan starte på neste fase [78]. Disse fasene og rekkefølgen på dem er [77, 76]:

1. Programvarekrav - Kravene for applikasjonen analyseres og skrives ned som utgangspunkt for fremtidig utvikling.
2. Analyse - Systemet analyseres for å kunne lage modeller og forretningslogikken som skal brukes i applikasjonen.
3. Programdesign - Dekker de tekniske designkravene som programmeringsspråk, datalag, tjenester osv.
4. Koding - Faktisk kildekode blir skrevet og alle modellene som ble utformet i tidligere faser implementeres.

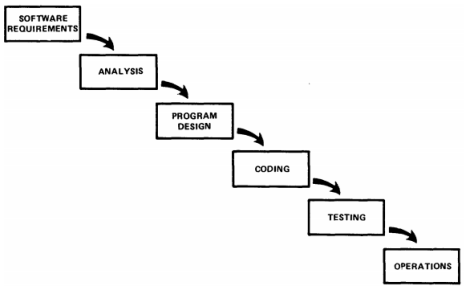
5. Testing - Testere går systematisk gjennom applikasjonen og rapporterer om feil som må løses.
6. Operasjoner - Applikasjonen blir distribuert. Denne fasen inneholder også nødvendig vedlikehold for å holde applikasjonen funksjonell.

Fordelen med vannfallsmodellen er at den er svært enkel å forstå, både for utviklere og kunder av applikasjonen. De rigide og klart definerte fasene gjør at det er lett å lage en tidslinje for leveranse og milepæler, samt at prosessen og resultatene er veldokumenterte [79]. Vannfallsmodellen passer derfor best for mindre prosjekt der kravene er veldokumenterte, klare og fikserte fra starten av og produktdefinisjonen er stabil gjennom hele prosjektet [79]. Ulempen med vannfallsmodellen er at de aller fleste systemer som utvikles i dag er store, komplekse og endres stadig, noe som gjør det vanskelig å definere alle krav før utviklingen begynner [79]. Dette har ført til at tradisjonelle og linære utviklingsmetodikker innen programvareutvikling slik som vannfallsmodellen har i senere tid blitt erstattet av agile, iterative metodikker [80].

Agil metodikk er en betegnelse for mange ulike iterative og inkrementelle utviklingsmetodikker som følger en agil filosofi, praksis og prinsipp [81, s.20]. Selv om agil metodikk og lignende lettvektsprinsipper har vært i bruk siden sent 1950-tallet [82, s.80], var det først på 1990-tallet at agil metodikk begynte å bli utbredt [82, s.87]. I 2001 gikk flere anerkjente personer innen utviklingsmetodikk sammen og utga dokumentet "Agile Manifesto" [83] som en reaksjon mot tradisjonelle linære metodikker som vannfallsmetoden og deres rigide krav om dokumentasjon [80]. Med deres erfaring i programvarebransjen beskriver de fire grunnleggende prinsipper for agil programvareutvikling [83]:

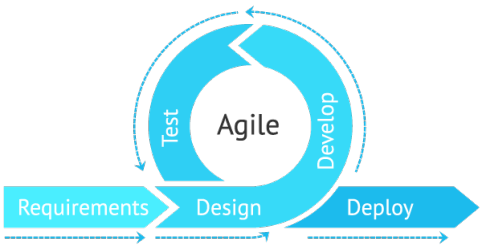
- Individer og interaksjoner over prosesser og verktøy.
- Fungerende programvare over omfattende dokumentasjon.
- Kundesamarbeid over kontraktsforhandlinger.
- Svare på endringer over å følge en plan.

Den høyste prioriteten er å gjøre kunden fornøyd, og med iterativ, agil utvikling kan kunden få leveranser i form av fungerende programvare kontinuerlig gjennom hele utviklingsprosessen [84]. Endringer i kravene ønskes velkommen, selv sent i utviklingstadiet [84]. Den mest effektive metoden for å formidle informasjon innen utviklingteamet er via direkte kommunikasjon ansikt til ansikt, ofte daglig [84]. Akkurat hvordan disse prinsippene blir fulgt i en utviklingsprosess, vil variere ut i fra hvilken agil utviklingsmetodikk som blir valgt.



Figur 8.1: Vannfallsmodellen beskrevet av Winston Royce

Bildekilde: [77]



Figur 8.2: Modell over agil utvikling

Bildekilde: [85]

Det ble bestemt i starten av utviklingsprosessen å gå for en agil utviklingsmetodikk framfor den mer tradisjonelle vannfallsmetoden. Selv om utviklingsteamet bare består av to personer og applikasjonen har et nokså snevert og spesifisert omfang, var ikke kravspesifikasjonen fastsatt fra starten av. Mye av funksjonaliteten i applikasjonen og valg av teknologi ble tatt underveis i prosjektets løp ved å ha ukentlige møter sammen med professor Hvasshovd. Det var et behov å kunne tilpasse seg gjennom prosjektet. Ettersom utviklerne i tillegg hadde mer erfaring med agil metodikk, var slik metodikk bedre egnet for hele prosjektet, både for fordypningsprosjektet og masteroppgaven.

8.1.1 Valg av agil metodikk

Eksempler på noen av de mest kjente agile metodikkene er Scrum [86], Kanban og Extreme Programming [87]. I dette prosjektet ble Scrum og Kanban vurdert.

Scrum, som for alvor tok av etter boken *Agile software development with Scrum* [88] i 2002, kjennetegnes ved at både kunder og utviklingsteam får roller med et bestemt ansvarsområde samt en rekke artefakter og hendelser som gjentas over prosjektets løp [89]. Prosjektet gjennomføres ved å deles inn i kortere iterasjoner kalt *sprinter* [89] som er en fast leveransesyklus på 1-4 uker [90]. Produkteierens ansvar er å lage en prioriert liste over produktets ønskede funksjonalitet, kalt *backlog*, som det blir tatt utgangspunkt i når man skal lage en egen *backlog* for *sprinten* som skal gjennomføres [81, s.24]. Utviklerteamet har ansvar for å implementere og potensielt levere et produkt eller funksjon for hver sprint og oppdatere *backloggen* underveis ved hjelp av en Scrum-tavle som visualiserer oppgavene [81, s.24]. Hver dag møtes utviklerne for korte *standup*-møter der de forteller hva som har blitt gjort og hva som skal gjøres for dagen slik at alle er oppdaterte [89]. Etter hver *sprint* gjennomføres et refleksjonsmøte for å se hva som kan forbedres i prosessen til neste *sprint* [89]. I tillegg får en av utviklerene en ekstra rolle som Scrum-master som skal sørge for at alle i teamet forstår og følger Scrum-praksis [81, s.24].

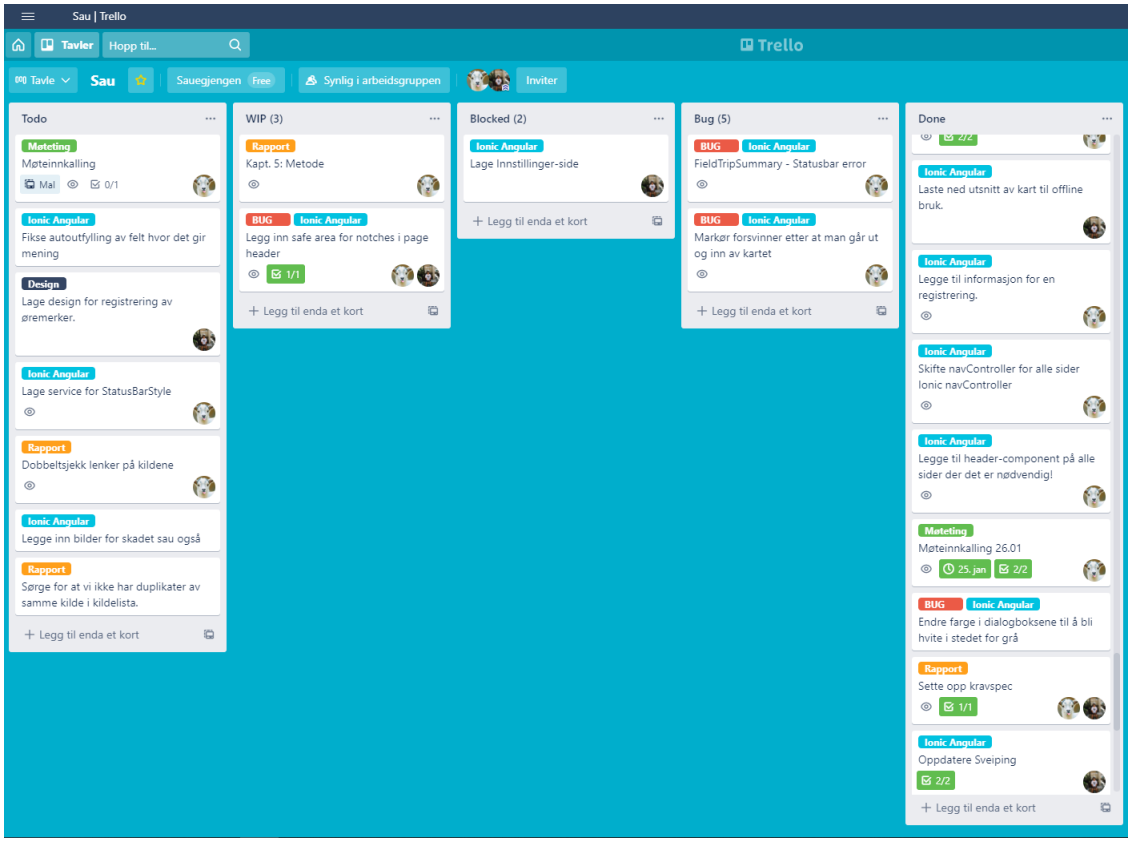
Kanban oppstod på 1940-tallet da Toyota begynte å bruke det i deres fabrikker for å optimalisere arbeidsflyten. På 2000-tallet ble prinsippene som ble utviklet hos Toyota overført til programvareutvikling [91]. Det er en agil metodikk med et par likheter til Scrum. Det brukes blant annet også en tavle for å vise *backlog*, pågående arbeidsoppgaver, utførte oppgaver og andre egendefinerte lister [91]. Kanban har et spesielt fokus på å optimalisere arbeidsflyt, maksimere effektivitet og unngå flaske-

halser [91]. Derfor benyttes det ikke tidsbestemte *sprinter* slik som i Scrum, men en kontinuerlig flyt av arbeid og distribusjon [91]. Det er heller ingen faste roller for de involverte i prosjektet. Pågående oppgaver for hver arbeidskategori blir begrenset slik at utviklerne ikke påtar seg for mange oppgaver samtidig og stagnerer fremdriften i prosjektet [91]. Om antall pågående arbeidsoppgaver er på den fastsatte maksgrensen for påbegynte oppgaver, må en utvikler hjelpe til med de påbegynte oppgavene før en ny oppgave kan startes på [91].

Selv om utviklerne hadde mest erfaring med å utvikle med Scrum både i skole- og arbeidssammenheng, ble det sett på som lite hensiktsmessig med tanke på at det bare var to utviklere i prosjektet. Det ville vært vanskelig å utfylle alle arbeidsrollene og det kunne ført til unødvendig bruk av tid på *standup-* og refleksjonsmøter. Kanban ble derfor et naturlig valg for dette prosjektet med mulighet for kontinuerlig flyt, fravær av bestemte arbeidsroller og visualisering samt begrensning av arbeidsoppgaver som gjorde det enklere å få øye på flaskehalser i utviklingsprosessen.

Trello

En essensiell del av Kanban er tavlen som visualiserer arbeidsoppgavene. Dette kan være en fysisk liste med klistrelapper eller digitale tavler. Under dette prosjektet ble samhandlingsverktøyet Trello [92] benyttet ettersom det kunne bli brukt i sanntid av begge utviklerne og la til rette for bruk av fargekoder slik at oppgavene kunne kategoriseres både for utvikling av programkode, skriving av oppgaven og andre egendefinerte kategorier. Det faktum at tavlen kunne oppdateres i sanntid gjorde også enklere for utviklerne å være oppdatert på hva den andre personen arbeidet med i de periodene utviklerne hadde hjemmekontor hver for seg. Utviklerne valgte å sette maksgrensen for påbegynte arbeidsoppgaver til tre, som det er mulig å se i figuren under i listen WIP (Work in Progress). Grensen ble satt til tre for å unngå opphoping av halvferdige oppgaver.



Figur 8.3: Skjermbilde av Trello-tavlen under prosjektet

8.2 Oppsummering

Det ble bestemt å bruke en agil utviklingsmetodikk og ikke den mer tradisjonelle vannfallsmodellen. Dette fordi kravspesifikasjonen ikke var satt før prosjektets start men ble utviklet underveis. Av mange ulike agile metodikker falt valget på Kanban. Kanbans kontinuerlige arbeidsflyt, fravær av bestemte roller for de involverte i prosjektet samt visualisering og begrensning av arbeidsoppgaver passet godt med tanke på størrelsen på utviklingsteamet og prosjektoppgaven. Samhandlingsverktøyet Trello ble benyttet som en digital Kanban-tavle.

Kapittel 9

Valg av metode

Dette kapittelet beskriver valget av metode for forskningsdelen av prosjektet. En forskningsplan har blitt utformet i henhold til rammeverket utviklet av Oates [93] i boken *Researching Information Systems and Computing*. Her beskriver hun utviklingen av en forskningsplan og deler det inn i seks deler kalt *The 6Ps of Research* (De 6 P-ene innenfor forskning): *Purpose* (Formål), *Products* (Produkter), *Process* (Prosess), *Participants* (Deltakere), *Paradigm* (Paradigme) og *Presentation* (Presentasjon). Paradigme vil ikke bli diskutert da det ikke er nødvendig ettersom at dette i hovedsak er et systemutviklingsprosjekt.

9.1 Formål

Her beskrives bakgrunnen for prosjektet, samt tidligere forskning innenfor samme felt. Det forklares hva som skiller vår forskning fra eksisterende forskning og til slutt vil forskningsspørsmålene for prosjektet bli presentert.

Bakgrunnen for oppgaven blir beskrevet i kapt. 3 og formålet blir beskrevet i kapt. 1.2. For å oppsummere er målet med oppgaven å utvikle en applikasjon som kan ta over for dagens løsning med penn og papir i forbindelse med registrering av informasjon under en oppsynstur. Applikasjonen som skal utvikles må både dekke dagens behov, tilby en løsning for blind interaksjon med en hånd mens brukeren ser gjennom en kikkert og gjøre det mulig for tilsynsansvarlige og bønder å dele informasjon innad i et beitelag. Tidligere forskning og løsninger innfor samme område inkluderer applikasjonen *Beitesnap* [70] av Fant AS fra 2017, et verktøy laget for å registrere observasjoner av husdyr på beitet. Et annet eksempel på tidligere arbeid er masteroppgaven *Effektivisering av manuell oppfølging av sau på utmarksbeite* hvor Dysthe og Kjerstad [71] utforsker en løsning for registrering av saueflokker i forbindelse med oppsynsturer. Det som skiller denne løsningen fra tidligere arbeid er at den vil gi brukerne muligheten til å registrere saueflokker blindt med en hånd mens brukeren ser gjennom en kikkert. Dette er nødvendig da sauflokkene på utmarksbeitet ofte er så langt unna at man ikke klarer å registrere nødvendig informasjon uten å ta i bruk kikkert. Løsningen vil også være kryssplattform og fungere på mobile enheter som kjører både iOS og Android, noe som ble forsøkt av Dysthe og Kjerstad [71] i 2018, men ble ikke fullført.

I løpet av prosjektet vil det utvikles en fullverdig, fungerende versjon av en applikasjon som kan kjøre på både iOS og Android. Applikasjonen skal deretter bruk-

ertestes for å finne potensielle svakheter i designet. Ved å gjøre dette er det et ønske å svare på følgende forskningsspørsmål:

- **F1:** Hvordan utvikle et digitalt verktøy for å bistå sauebønder, beitelag og tilsynsansvarlige på oppsynstur slik at arbeidet med manuell registrering blir mer effektivt og raskere?
- **F1.1:** Kan det utvikles et digitalt system som erstatter dagens løsning med penn og papir, men fortsatt dekker alle brukerens behov?
- **F1.2:** Hvordan kan et digitalt system bistå sauebønder, beitelag og tilsynsansvarlige slik at de kan samhandle og dele informasjon om oppsynsturer med hverandre og norske myndigheter?
- **F1.3:** Hvordan utvikle et brukergrensesnitt som muliggjør registrering av sau uten å måtte se på mobilskjermen?

9.2 Produkter

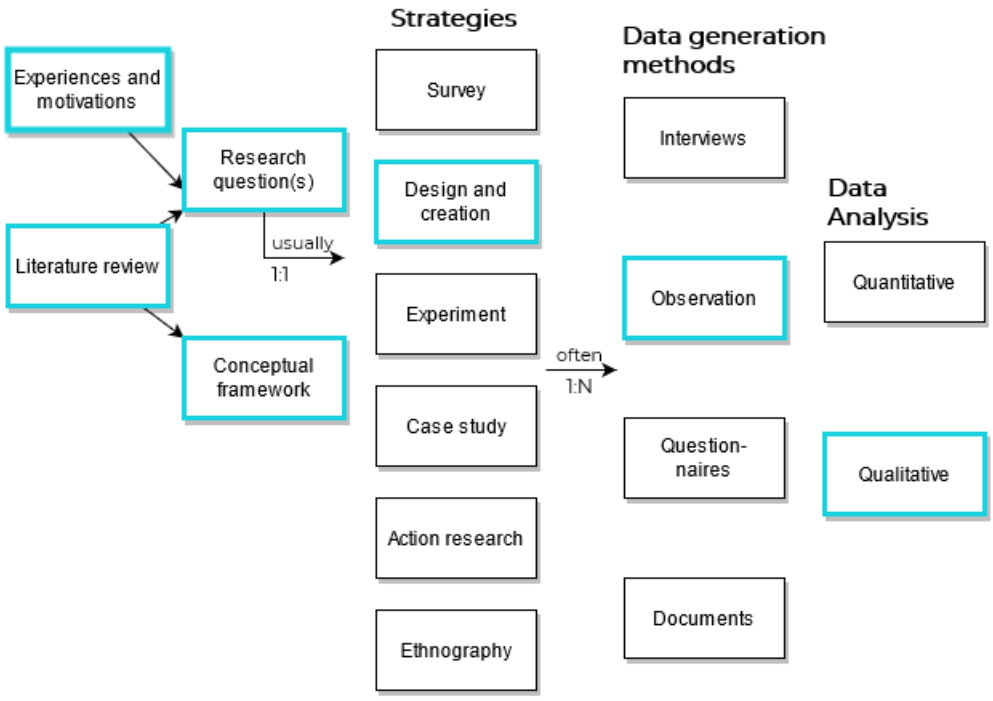
Dette underkapittelet forklarer hvilke produkter som vil komme som et resultatet forskningsprosjektet.

Hovedproduktet fra forskningen vil være en applikasjon som kan erstatte dagens løsning for registrering av informasjon om saueflokker på utmarksbeitet med penn og papir. Samtidig skal applikasjonen tilby utvidet funksjonalitet med tanke på informasjonsflyt og brukbarhet. Selv om hovedmålet med prosjektet er å utvikle en applikasjon, vil resultatene både fra fordypningsprosjktet og selve masteroppgaven være med på å bidra til økt kunnskap om både utvikling av brukergrensesnitt for blind bruk, samt utvikling av kryssplattform-applikasjoner for bruk i forbindelse med informasjonsregistrering hvor man ikke har tilgang til internett der GPS og kart spiller en hovedrolle.

9.3 Prosess

Underkapittelet om prosess beskriver planen for hvordan det er tenkt å gå fram med forskningen under prosjektet. Prosessen baserer seg på figur 9.1 hentet fra *Researching Information Systems and Computing* [93].

Arbeidet med prosjektet var motivert av ønsket om å lage en applikasjon, utviklet basert på eksisterende litteratur, løsninger og forskning som kunne være med å bidra til å redusere antall sauer som dør hvert år på utmarksbeitet. Det var også et ønske om å gjøre arbeidshverdagen til sauebønder og tilsynsansvarlige enklere. Prosjektet startet med et litteratursøk inn i eksisterende løsninger for denne typen applikasjoner. Det ble også gjort et grundig arbeid for å kartlegge brukerens behov, basert både på dagens løsning men også hva som kreves fra myndighetens side. Dette både for å hjelpe til med å forme forskningsspørsmål, men også for å kunne danne et konseptuelt rammeverk for prosjektet.



Figur 9.1: Figuren viser den valgte veien for forskningsprosess markert med fargede bokser

Bildekilde: [93, s.33]

Planen er å bruke *Design og creation*-strategien for å implementere en applikasjon basert både på resultatene fra fordypningsprosjektet og litteratursøket. Når en fullstendig første versjon av applikasjonen har blitt utviklet planlegges det å brukerteste denne på fem ulike brukere. Her vil *Observation* brukes som metode for å generere data fra brukertestene. Etter at testene er utført vil det bli gjort en kvalitativ analyse av dataene.

9.4 Deltagere

Dette kapittelet beskriver deltakerne i prosjektet og deres roller.

Hoveddeltakerne i dette prosjektet vil være de som utfører selve prosjektet, altså Kimia Dadar Abtahi og Trym Vegard Gjelseth-Borgen. Personen som kommer til å veilede oss gjennom prosjektet, gi nyttig erfaringsbasert informasjon med tanke på oppsynsturer og som vil fungere som produkteier er professor Svein-Olaf Hvasshovd. Det vil også være fem testdeltakere som deltar for å hjelpe til med brukertestene av applikasjonen. Data fra brukertestene og om testdeltakerne vil være anonymisert.

9.5 Presentasjon

Underkapittelet om presentasjon beskriver hvordan resultatene fra forskningsprosjektet vil bli presentert etter endt prosjekt.

Resultatet av forskningen vil bli presentert i masteroppgaven. Resultatene vil også

kunne være med på å utvikle en forbedret versjon av applikasjonen, hvis dette skulle være ønskelig etter prosjektets slutt.

9.6 Oppsummering

Prosjektet vil starte med et litteratursøk for å få et innblikk i dagens situasjon og eksisterende løsninger. Funnene fra litteratursøket vil bli brukt til å danne forskningsspørsmål og et konseptuelt rammeverk for prosjektet. Strategien *Design and Creation* vil benyttes i form av design og utvikling av en applikasjon. Applikasjonen vil så brukestestes hvor observasjoner vil legge grunnlaget for datainnsamlingen. Deretter vil det bli utført en kvalitativ analyse på resultatene fra brukertestene.

Kapittel 10

Valg av teknologi

Dette kapittelet gjennomgår teknologien som ble vurdert til programvareutvikling, hvilken teknologi valget falt på og årsaken til valgene som ble gjort. For at applikasjonen skulle ha mulighet til å nå kravene spesifisert i kravspesifikasjonen var det viktig å vurdere nøye hvilke teknologier som passet best til den gitte problemstillingen.

10.1 Utviklingsrammeverk

Under valget av utviklingsrammeverk var det to faktorer som var spesielt viktig å ta med inn i vurderingen. De spesifikke kravene det siktes til er det funksjonelle kravet F1 beskrevet i tabell 7.1 og det ikke-funksjonelle kravet IF2 beskrevet i tabell 7.2. F1 og IF2 handler henholdsvis om at utviklingsrammeverket må gjøre det mulig å implementere et kartgrensesnitt ved hjelp Kartverkets tjenester og at det skal tilrettelegge for utvikling av en kryssplattform-applikasjon som skal fungere på både iOS og Android.

10.1.1 Krav om en kryssplattform-applikasjon

På bakgrunn av det ikke-funksjonelle kravet IF2 var det flere ulike utviklingsrammeverk som ble vurdert. Likt for alle de vurderte rammeverkene var det at de tilbyr utvikling av en applikasjon for både iOS og Android med bare én kodebase. Ved å bare ha én kodebase vil utviklingstiden kunne reduseres betraktelig ettersom at man slipper å lage den samme applikasjonen to ganger. Framtidig vedlikehold og oppdateringer vil også være lettere og raskere å utføre. Utviklingsrammeverkene som ble vurdert til applikasjonen var *Flutter* [94] utviklet av Google, *NativeScript* [95] som er et utviklingsrammeverk for kryssplattformutvikling med åpen kildekode, *React Native* [96] laget av Facebook basert på JavaScript-rammerverket *React* [97] og *Ionic* [98] som er et kryssplattformrammeverk spesielt laget for mobile enheter. Selv om alle disse utviklingsrammeverkene tilbyr funksjonaliteten for å lage en applikasjon for både iOS og Android med bare én kodebase, vil det likevel oppstå situasjoner der det er nødvendig å skrive plattformspesifikk kode. Graden av hvor ofte slike situasjoner oppstår vil variere fra rammeverk til rammeverk. Av de nevnte rammeverkene er Ionic det rammeverket som krever minst plattformspesifikk kode, mens React Native krever mest plattformspesifikk kode. Flutter og NativeScript havner en plass midt i mellom disse to [99]. Ionic er også et av rammeverkene med flest ferdiglagde brukergrensesnitt-komponenter, noe som gjør at man under utvikling kan

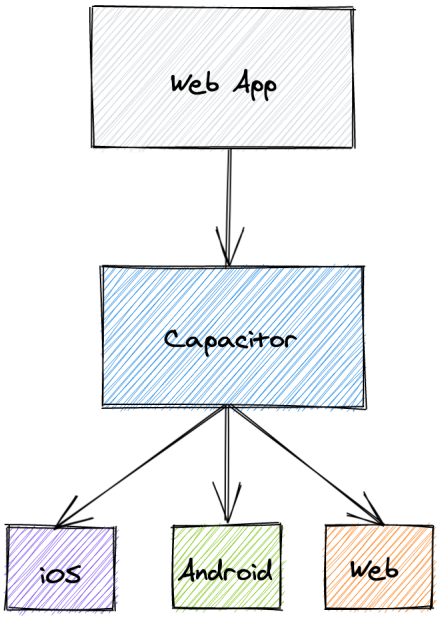
bruke mindre tid på grensesnittspesifikk implementasjon og mer tid på implementasjon av funksjonalitet. React Native har minst ferdiglagde komponenter mens Flutter stiller likt med Ionic. NativeScript havner en plass mellom Ionic og React Native [99].

10.1.2 Krav om integrasjon med NorgesKart

Det funksjonelle kravet F1 beskrevet i tabell 7.1 beskriver at løsningen krever at kartet i applikasjonen bruker Kartverkets løsninger i form av tjenesten NorgesKart. Geonorge, som er det det nasjonale nettstedet for kartdata og annen stedfestet informasjon i Norge [100], linker til JavaScript-biblioteket *Leaflet* [101] i deres brukerveiledning om bruk av Kartverkets tjenester [102]. Også tidligere forsøk på implementasjon av en lignende applikasjon har brukt JavaScript-biblioteket Leaflet for å vise fram kartdata [71, s.116]. For at implementasjonen av kartgrensesnittet i applikasjonen skulle gå så smertefritt som mulig ble det ble derfor tidlig bestemt at Leaflet skulle brukes. Derfor ble det viktig at utviklingsrammeverket støttet bruk av Leaflet. Flutter skrives i språket *Dart* [103] og støtter derfor ikke bruk av JavaScript-bibliotek Leaflet ettersom at Dart og JavaScript ikke er kompatible. Leaflet trenger direkte tilgang til et DOM-element [104] for å legge selve kartet inn i brukergrensesnittet [105]. Selv om React Native skrives i språket JavaScript, støtter ikke React Native JavaScript-biblioteker som trenger direkte tilgang til DOM-elementer [106]. Det ble derfor ikke aktuelt å bruke hverken Flutter eller React Native i løsningen. Både Ionic og Native Script støtter bruk av JavaScript-rammerverket Angular [107, 108] for å lage brukergrensesnitt i applikasjonen. Angular støtter direkte tilgang til DOM-elementer [109] i tillegg til at det støtter bruk av JavaScipt-bibliotek [110], noe som gjør at Leaflet vil fungere i både Ionic og NativeScript.

10.1.3 Ionic + Capacitor

Valget for hvilket utviklingsrammeverk som skulle brukes i applikasjonen falt til slutt på Ionic. Selv om NativeScript var en mulig kandidat, ble det bestemt å bruke Ionic ettersom at det potensielt kunne resultere i en mindre og lettere vedlikeholt kodebase og fordi Ionic har flere ferdiglagde brukergrensesnittkomponenter å velge mellom enn NativeScript [99]. Ionic er en hybrid utviklingsplattform med åpen kildekode som kom på markedet i 2013 [111]. Ionic muliggjør utvikling av kryssplattform-applikasjoner med web-teknologi og støtter de mest kjente JavaScript-rammerverkene som Angular, Vue.js og React, samt vanlig HTML, CSS og JavaScript [98]. For at kode utviklet med web-teknologi skal kjøre på mobile enheter og ha tilgang til maskinvarefunksjonalitet som mobilenes kamera eller GPS er det et behov for et mellomledd som binder sammen webapplikasjonen og maskinvaren [112]. *Capacitor* [113] og *Cordova* [114] er eksempler på slike mellomledd. Det ble besluttet å bruke Capacitor i utviklingen av denne applikasjonen. Årsaken til dette er at Capacitor er mer brukt, har 99% bakoverkompatibilitet med programtillegg fra Cordova og fordi Capacitor kom ut i 2018 [115]. Capacitor bruker også nyere moderne API-er som ikke var tilgjengelig da Cordova kom ut i 2009 [115].



Figur 10.1: Forenklet figur som viser oppbygningen av en applikasjon som bruker web-teknologi sammen med Capacitor

Bildekilde: [112]

10.2 Biblioteker

Dette delkapittelet går gjennom de viktigste programmeringsbibliotekene som blir brukt i applikasjonen *Sauron*.

10.2.1 Frontend-bibliotek

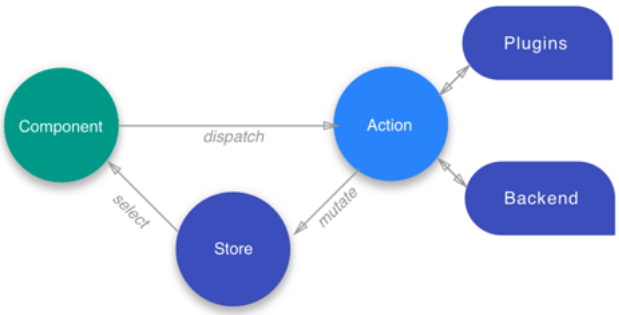
Ionic støtter bruk av flere forskjellige JavaScript-rammerverk for utvikling av webapplikasjonsdelen av mobilapplikasjonen, samt støtte for vanlig HTML, CSS og standard JavaScript [98]. JavaScript-rammeverkene som støttes per dags dato (17.03.2021) er Angular utviklet av Google [116], React utviklet av Facebook [96] og Vue.js som er et uavhengig JavaScript-rammeverk med åpen kildekode [117]. Under utviklingen av større webapplikasjoner lønner det seg å bruke et JavaScript-rammeverk framfor å bruke standard HTML, CSS og JavaScript. Etter hvert som applikasjoner vokser kreves det flere filer og mer kode per fil. Å holde kontroll på disse på egenhånd ved hjelp av mapper og ryddig kode kan fort bli vanskelig og kan føre til et spaghettilignende [118, s. 321] rot. De nevnte JavaScript-rammeverkene er laget for å holde koden i sjakk, gjøre det mulig å lage gjenbrukbare komponenter som fører til redusert kode og gir applikasjonen rom til å vokse mens den samtidig er lett å vedlikeholde [119]. En av ulempene med slike JavaScript-rammeverk er at de kan ha en relativt bratt læringskurve. Ettersom at begge utviklerne hadde erfaring med utvikling med JavaScript-rammeverk fra før av ble denne problemstillingen ikke tatt i betraktning.

Når det kom til valget av hvilket spesifikt rammeverk som skulle brukes stod det mellom React og Angular. Når applikasjonen ble påbegynt høsten 2020 var Ionic med Vue.js enda i Beta-versjon [120] og ble derfor ikke sett på som et alternativ. Valget

falt til slutt på å bruke Angular da Angular tilbyr strengere form for kodestruktur hvor HTML, CSS og JavaScript/TypeScript er delt opp i hver sin fil. Angular tilbyr også mulighet for å bruke TypeScript [121], en versjon av JavaScript som gjør det mulig å typesette koden for økt struktur og enklere feilsøking [122].

10.2.2 NGXS

Under planleggingen av applikasjonen ble det sett på som nødvendig å implementere en form for tilstandshåndtering for brukergrensesnittet. Applikasjonens flyt, spesielt under registrering av en saueflokk (se kapt. 11.2.5), består av flere ledd hvor å holde kontroll på antall registrerte søyer og lam med forskjellig farge, slips og øremerker er kritisk for at registreringen skal bli korrekt og brukeropplevelsen skal bli tilfredstillende. For å få til dette ble det valgt å bruke tilstandsbiblioteket NGXS [123]. NGXS er modulert etter det populære tilstandsbiblioteket Redux [124] laget for React, men reduserer redundant *boilerplate*-kode ved å ta i bruk moderne TypeScript-funksjonalitet som klasser og dekoratører [123]. Dataflyten ved bruk av NGXS illustreres i figur 10.2. Her referer *Components* til ulike komponenter i brukergrensesnittet. *Store* er den lagrede tilstanden til applikasjonen, mens *Actions* er handlinger utført av *Components* som *muterer* tilstanden i *Store*. Når en komponent utfører en *Action* og muterer en verdi i *Store*, vil alle komponenter som tar i bruk den muterte verdien få beskjed om dette slik at komponentene får mulighet til å hente den siste tilgjengelige tilstanden. Applikasjonen består av flere ulike komponenter som har ansvaret for forskjellige funksjonalitet som for eksempel å lese opp tekst til brukeren under blind registrering (tekst-til-tale), ta i mot input fra brukeren eller vise hva som er registrert i en oppsummering på skjermen. For at alle disse ulike komponentene skulle ha tilgang til den samme informasjonen ble bruken av NGXS essensiell.



Figur 10.2: Dataflyt ved bruk av NGXS

Bildekilde: [125]

10.2.3 Tekstopplesning

For at applikasjonen skulle nå det funksjonelle kravet F4.1: *Brukeren skal få verbale tilbakemeldinger om utførte handlinger underblind bruk slik at brukeren slipper å se på skjermen* beskrevet i tabell 7.1 var det nødvendig å implementere en form for stemme som kunne gjengi informasjon vist på skjermen. Her sto valget mellom å enten bruke en tekstoppleser ofte brukt for blinde eller svaksynte, eller å spille inn en rekke lydfiler som kunne mikses og spilles av for å lese opp informasjon fra applikasjonen. Her falt valget på å bruke en tekstoppleser ettersom å spille inn lydfiler

for hver eneste situasjon kom til å ta særdeles lang tid, samt at lydfilene ville brukt en del lagringsplass. Bruk av ferdiginnspilte lydfiler ville også gjort applikasjonen dårlig utrustet for eventuelle endringer eller utvidelser i framtiden da dette ville krevd nye lydklipp. Både iOS og Android innebygd støtte for tekstopplesning, ettersom at det gir økt tilgjengelighet for svaksynte og blinde [126, 127].

Etter et forsøk på å implementere et åpent programtillegg for Capacitor kalt *capacitor-community/text-to-speech* [128] i applikasjonen, kom det fram at det ikke hadde den funksjonaliteten som var ønsket for opplesing av tekst på iOS. For at brukeropplevelsen skulle bli så bra som mulig var det et behov for å kunne avbryte en tekstopplesning før den var ferdig og starte på opplesning av ny tekst. Dette fungerte utmerket på Android, men lot seg ikke gjøre i iOS med dette programtillegget. Det ble derfor tatt en beslutning på å lage et eget programtillegg til Capacitor for tekstopplesning, ettersom at dette er sentral funksjonalitet i applikasjonen. Gjennom Capacitor er det heldigvis enkelt å utvikle og teste egne programtillegg både på Android og iOS [129]. Programtillegget *capacitor-tts-plugin* ble laget for bruk i applikasjonen og er publisert på npmjs.com (`https://www.npmjs.com/package/capacitor-tts-plugin`), noe som gjør at man enkelt kan installere og bruke programtillegget i sitt eget prosjekt ved hjelp av NPM [130].

10.3 Backend-løsning

For at samhandling mellom brukere av applikasjonen skulle være mulig ble det behov for en form autentisering av brukere og sentral lagring av registrerte oppsynsturer. Det finnes mange ulike måter å gjøre dette på. Løsningene som ble vurdert for denne applikasjonen var enten Googles skybaserte løsning Firebase [131] eller en kombinasjon av rammeverket Node.js [132] for å lage en tjener med MongoDB [133] som databaseløsning. Valget falt til slutt på Firebase da det er svært enkelt å bruke, ikke har behov for en tjener og tar seg av autentisering og lagring eksternt i skyen. Kombinasjonen av en dedikert tjener med en ekstern dokument-database som MongoDB kunne passet utmerket til prosjektet da det gir flere muligheter og større frihet. Likevel krever en slik løsning både mer midler i form av ekstra maskinvare for å rulle ut tjenerkoden på, samt forlenget tid for implementering. Det ble derfor sett på som mer hensiktsmessig å gå for Firebase.

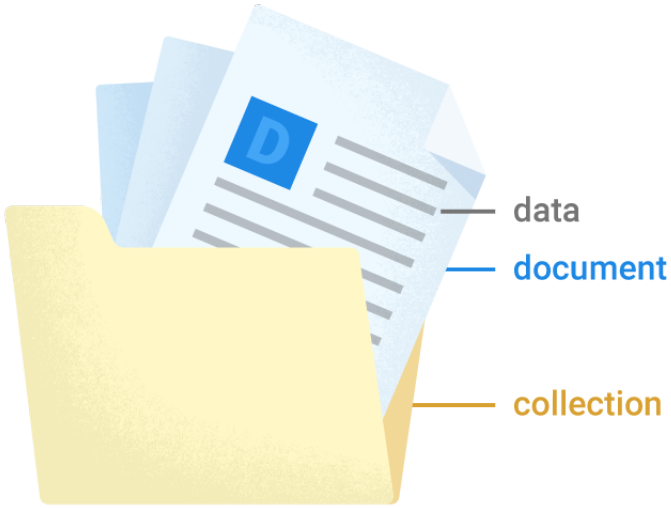
10.3.1 Firebase

Firebase ble laget av Firebase Inc i 2011 [134], før det ble kjøpt opp av Google i 2014 [135]. Firebase er en BaaS, Backend-as-a-Service, som gjør at man istedenfor å lage sin egen tjener-løsning/backend, har mulighet til å bruke tjenesten Firebase for alle tjener-relaterte oppgaver [136]. Dette gjelder alt fra autentisering med Firebase Authentication, lagring med Cloud Firestore eller maskinlæring med Firebase Machine Learning [137].

Cloud Firestore

For lagring av brukere, beitelag og registrerte oppsynsturer bruker applikasjonen Cloud Firestore. Cloud Firestore er en NoSQL-database som lagrer filer og dokumenter på JSON-format i skyen. Cloud Firestore lagrer data i en tredelt struktur. *Data*, i form av JSON, lagres i *documents*, som igjen lagres i *collections*. En *collection*

kan ha en eller flere *documents*, men et *document* har også mulighet til å peke til nye *collections* [138]. Systemet gjør det altså mulig å lagdele data som lagres i databasen for struktur og logikk. Dette var funksjonalitet som var nødvendig for å kunne dele opp tilgangen på registrerte oppsynsturer basert på brukere og beitelag.



Figur 10.3: Datastruktur i Cloud Firestore.

Bildekilde: [138]

Firebase Authentication

Firebase Authentication er løsningen for autentisering i skyen som tilbys av Firebase. Tjenesten støtter innlogging gjennom Facebook, Google, Twitter osv, men også innlogging med epost og passord. Data for den innloggede brukeren lagres i skyen og brukes til autentisering når en bruker prøver å lese eller skrive til en fil i Cloud Firestore. Den eneste funksjonaliteten som trenger å bli implementert på selve enheten er en innloggingsside som sender brukernavn og passord til Firebase Authentication [139]. Ved å velge Firebase Authentication for autentisering ble en prosess som erfaringsmessig tar lang tid, gjort på bare én dag. Dette gjorde det mulig å bruke mer tid på implementasjon av viktig funksjonalitet i andre deler av applikasjonen.

10.4 Oppsummering

For utvikling av selve grensesnittet ble utviklingsrammeverket Ionic sammen med JavaScript-rammeverket Angular valgt. Det ble her oppdaget at Ionic passet best for utvikling av denne applikasjonen basert på kravene presisert i kravspesifikasjonen. For å få tilgang til *native*-funksjonalitet hos mobiltelefonene brukes Capacitor, da det var det mest moderne alternativet med mest funksjonalitet. Tilstandshåndteringen i applikasjonen tas hånd om av biblioteket NGXS som er laget for Angular og krever mindre *boilerplate*-kode enn tilsvarende bibliotek. Det ble utviklet et eget programtillegg til Capacitor for tekstopplesning slik at applikasjonen skulle kunne håndtere tekst-til-tale likt på både iOS og Android. Programtillegget ligger nå ute i NPM-registeret, noe som gjør at det er enkelt å installere eventuelle framtidige opp-

dateringer. Backend-løsningen i applikasjonen er gjort med en Backend-as-a-Service i form av Firebase, da det tilbyr all ønsket funksjonalitet mens det samtidig ga kortest mulig implementasjonstid.

Kapittel 11

Design

Dette kapittelet tar for seg prosessen rundt designet av brukergrensesnittet til applikasjonen, resultatene som følge av designfasen og verktøyene som ble tatt i bruk under prosessen og utviklingen av grensesnittene.

11.1 Verktøy

Vektøyene som har blitt brukt i forbindelse med designfasen av applikasjonsutviklingen beskrives i dette kapittelet.

11.1.1 Coolors

Verktøyet *Coolors* fra coolors.co [140] ble benyttet for å gjøre beslutninger knyttet til fargevalg i applikasjonen enklere. Verktøyet gjør det mulig å generere ulike fargepaletter helt fra bunnen av og med utgangspunkt i enkeltfarger. Verktøyet har også funksjonalitet for å hente ut fargene som finnes på flere forskjellige formater, som blant annet hex [141] og rgb [142] som er vanlige formater for å beskrive farger i programvareutvikling.

11.1.2 Affinity Designer

Under utvikling av *Sauron* viste det seg at det ble et behov for ikoner og illustrasjoner som ikke lot seg lage i HTML eller CSS. For å unngå unødvendige kostnader og bruk av tid ble det valgt å utvikle disse elementene fra bunnen av framfor å bruke gratis ferdiglagde utgaver av lav kvalitet eller å betale for bedre utgaver. For design av disse elementene ble verktøyet *Affinity Designer* [143] brukt. Affinity Designer er et designverktøy som brukes for å lage ulike grafikker, logoer, ikoner med mer og kan sammenlignes med det mer kjente *Illustrator* fra Adobe [144]. Affinity Designer ble i hovedsagt valgt fordi utviklerne allerede hadde erfaring og tilgang til programvaren.

11.1.3 Figma

For å gjøre den programmatiske utviklingen av brukergrensesnittet mer effektivt ble det gjort et valg om at alle de ulike sidene av grensesnittet i applikasjonen først skulle tegnes i et design- og prototypeverktøy. Valget falt på verktøyet *Figma* [145]. *Adobe XD* [146] ble også vurdert. Adobe XD og Figma er ganske like i funksjonalitet, men på grunn av tidligere erfaring i Figma ble Adobe XD valgt bort. Gratisversjonen av

Figma tilbyr også mer funksjonalitet enn gratisversjonen av Adobe XD.

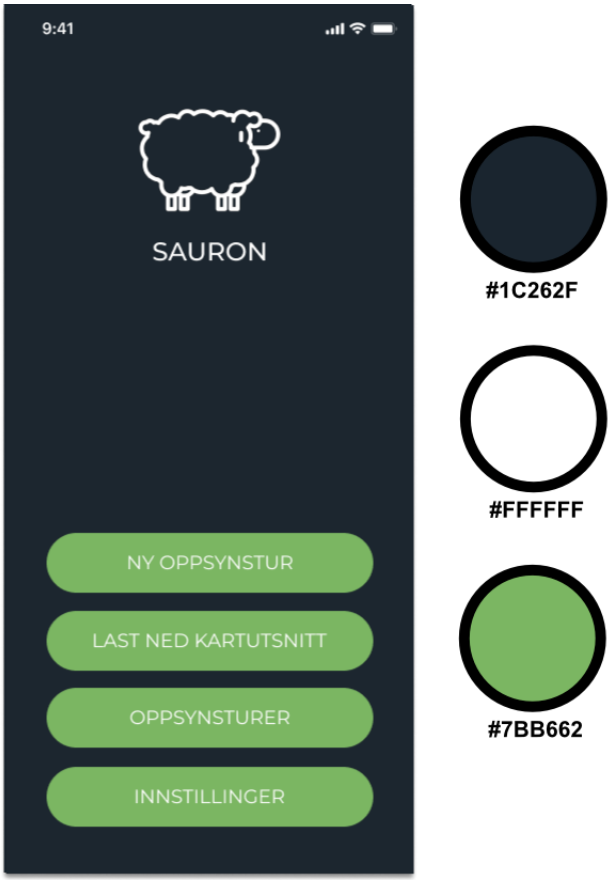
Figma gjør det mulig å designe fullstendige brukergrensesnitt som kan gjøres om til enkle prototyper for testing av basisfunksjonalitet. Figma er også laget for at det skal være enkelt å samarbeide og tilbyr funksjonalitet for å designe elementer sammen i sanntid over nettet.

11.2 Resultater

Resultatene fra designfasen gjennomgås i dette kapittelet. Dette gjelder alle de ulike grensesnittene som har blitt designet i Figma, samt fargevalget for applikasjonen vil bli presentert og forklart.

11.2.1 Fargevalg

Valget av fargepaletten i applikasjonen falt på en applikasjon med mørkt tema, både fordi at det er populært i dagens applikasjonsmarked, men også fordi det kan føre til at applikasjonen på mobiler med OLED-skjermer [147] vil ta mindre strøm under bruk [148]. Dette er viktig ettersom at applikasjonen skal kunne brukes lenge uten tilgang på strøm under oppsynstur. Bakgrunnsfargen (hovedfargen) i applikasjonen er en mørk marineblå farge. Denne blir komplementert med en dus grønnfarge som blir brukt på knapper og enkelte andre elementer. Fargen på tekst gjennom nærmest hele applikasjonen er hvit. Fargevalget har resultert i en applikasjon hvor det er god kontrast mellom ulike elementer. Det er også god kontrast mellom de ulike elementene og tilhørende tekst. For å sørge for at applikasjonen har god tilgjengelighet for personer med de vanligste formene for fargeblindhet ble nettleserutvidelsen-utvidelsen *Colorblinding* [149] brukt under testing av ulike fargekombinasjoner.



Figur 11.1: Illustrasjon av hovedmenyen i applikasjonen med de ulike fargene som er brukt og deres tilhørende hex-verdier

11.2.2 Nedlasting av kartutsnitt

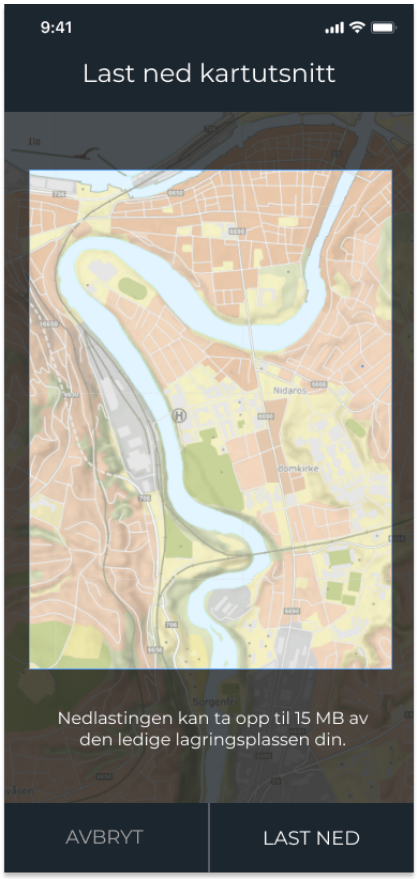
Brukeren kan navigere seg til oversikten over nedlastede kartutsnitt fra hovedmenyen (figur 11.1) ved å trykke på "Last ned kartutsnitt"-knappen. Oversikten viser alle nedlastede kartutsnitt, hvor mye plass kartutsnittene tar på mobilen og når de ble lastet ned.

Figur 11.2: Prototypedesign av side for oversikt over nedlastede kartutsnitt

De tre prikkene på høyresiden av hvert nedlastede kartutsnitt kan trykkes på for å åpne en valgmeny (figur 11.3). Valgmenyen tilbyr fire valg. Brukeren kan trykke på "Oppdater" for å oppdatere et valgt kartutsnitt og dermed laste ned kartutsnittet på nytt igjen. Ved å trykke på "Gi nytt navn" kan brukeren gi kartutsnittet et nytt navn framfor å bruke det autogenererte navnet gitt av applikasjonen. Dette kan være nytting om brukeren har flere forskjellige kartutsnitt det ønskes å holde kontroll på. "Slett"-knappen sletter et valgt kartutsnitt og "Avbryt"-knappen lukker valgmenyen.

Figur 11.3: Prototypedesign av side for oversikt over nedlastede kartutsnitt med åpen valgmeny

Om brukeren trykker på den grønne knappen med et pluss-tegn nede i høyre hjørne på "Nedlastede kartutsnitt"-siden, vil det navigeres til siden hvor det kan lastes ned nye kartutsnitt (figur 11.4). Ved å flytte og zoome på kartet slik at det ønskede kartutsnittet kommer innenfor det markerte rektangelet kan brukeren selv velge hvor stort et kartutsnitt skal være. Når brukeren har funnet området på kartet det er ønskelig å lage et kartutsnitt av trykker brukeren på "Last ned"-knappen. Ved å trykke på "Avbryt"-knappen vil applikasjonen bli tatt tilbake til oversikten over nedlastede kartutsnitt.



Figur 11.4: Prototypedesign av side for å laste ned nye kartutsnitt

11.2.3 Ny oppsynstur

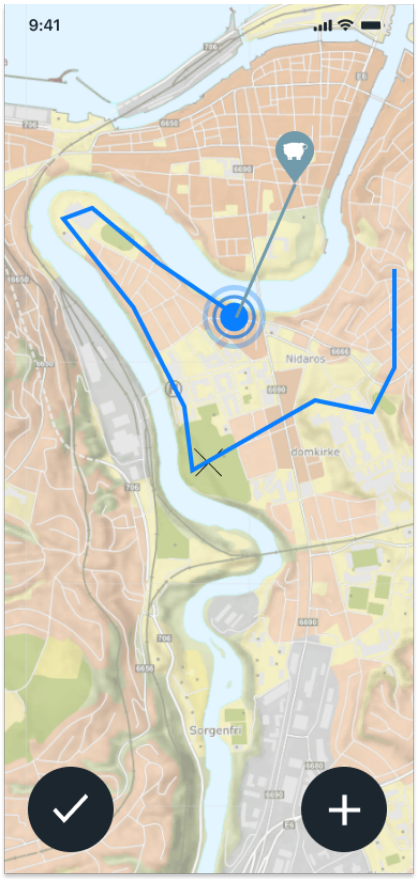
Registrering av informasjon

Brukeren kan trykke på "Ny oppsynstur"-knappen på hovedmenyen (figur 11.1) for å registrere en ny oppsynstur. Da blir brukeren først tatt til "Ny oppsynstur"-siden (figur 11.5). Her kan det registreres navnet på deltagerne som er med på oppsynsturen, samt at legge til en beskrivelse for turens formål og baktanke. Når registreringen av informasjon er fullført kan brukeren trykke på "Start oppsynstur"-knappen for å starte selve oppsynsturen. Brukeren vil da bli tatt til kartsiden som er hovedsiden under en oppsynstur.

Figur 11.5: Prototypedesign av side for å registrere informasjon til en ny oppsynstur

Kart

Kartsiden (figur 11.6) i applikasjonen er, som nevnt tidligere, siden som blir mest brukt under en oppsynstur. Kartsiden viser ved hjelp av et ikon formet som en blå sirkel hvor brukeren befinner seg på kartet, mens den blå streken viser hvor brukeren har gått.



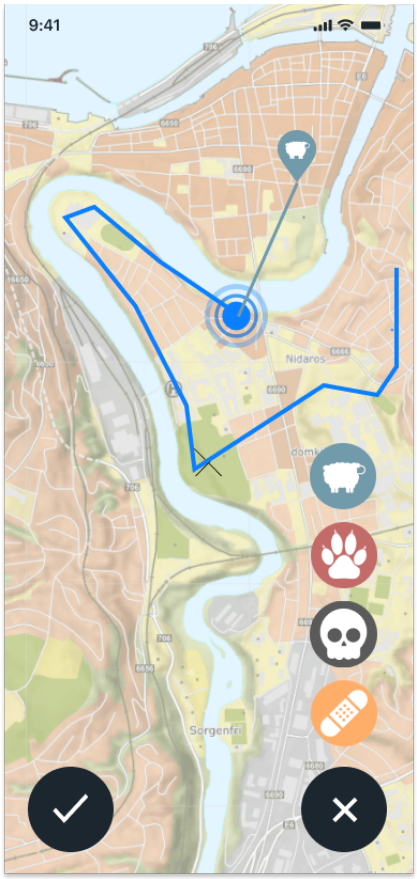
Figur 11.6: Prototypedesign av side for kart

Meny i kart

Ved å trykke på pluss-knappen nede i høyre hjørne åpnes valgmenyen for å legge til en ny registrering (figur 11.7). Her kan brukeren velge mellom (fra øverst til nederst) å legge til en registrering for en observasjon av en saueflokk, rovdyr, død sau eller skadet sau. Når en registrering har blitt fullført legges det til to ny elementer på kartsiden:

- Et merke som viser hvilken type registrering som har blitt utført (død sau, rovdyr etc.). Dette merket blir plassert på posisjonen hvor observasjonen ble gjort.
- En linje som peker fra hvor observasjonen ble gjort fra og til merket.

På den måten kan brukeren i etterkant av en oppsyntur vite både hvor eventuelle registreringer har blitt observert i tillegg til hvor de ble observert fra.



Figur 11.7: Prototypedesign av side for kart med åpen valgmeny

11.2.4 Fullføre oppsynstur

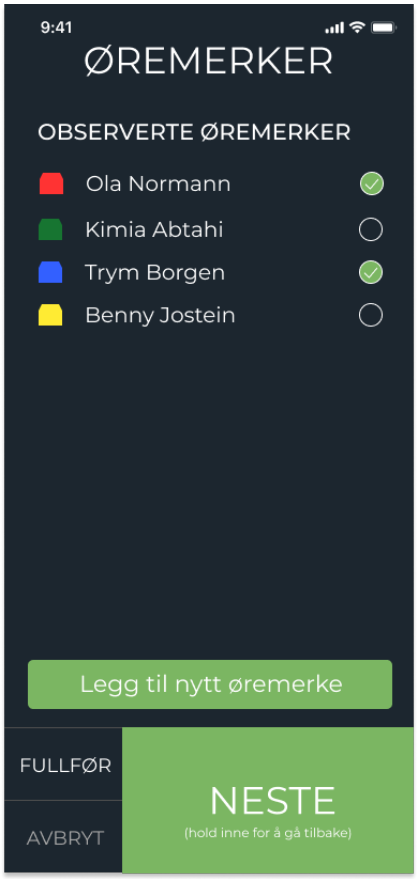
En oppsynstur kan fullføres ved å trykke på fullfør-knappen symbolisert med en hake nede i venstre hjørne på kartsiden (figur 11.6). Ved å gjøre dette vil applikasjonen navigere til "Oppsummering"-siden (figur 11.8). Oppsummeringssiden viser generell informasjon om turen, samt hvor brukeren har gått og hvilke observasjoner som har blitt gjort. Her er det også mulig å legge til en kommentar før det trykkes "Fullfør" og oppsynsturen blir lagret.

Figur 11.8: Prototypedesign av side for oppsummering av en oppsynstur

11.2.5 Legge til en registrering

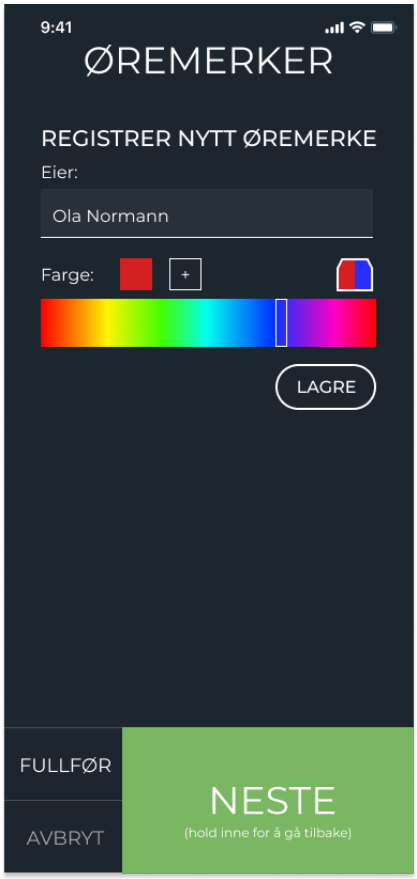
Registrering av sau

Grensesnittet for registrering av sau ble nesten ferdig utformet i fordypningsprosjektet og kan leses om i sin helehet der [1, s.21-32]. Det eneste som har blitt lagt til grensesnittet for registrering av sau siden da er funksjonalitet for å legge til øremerker. Figur 11.9 viser den nye siden for registrering av øremerker. Listen er en flervalgsliste som inneholder fargen på merket og navnet på bonden som øremerkene tilhører.



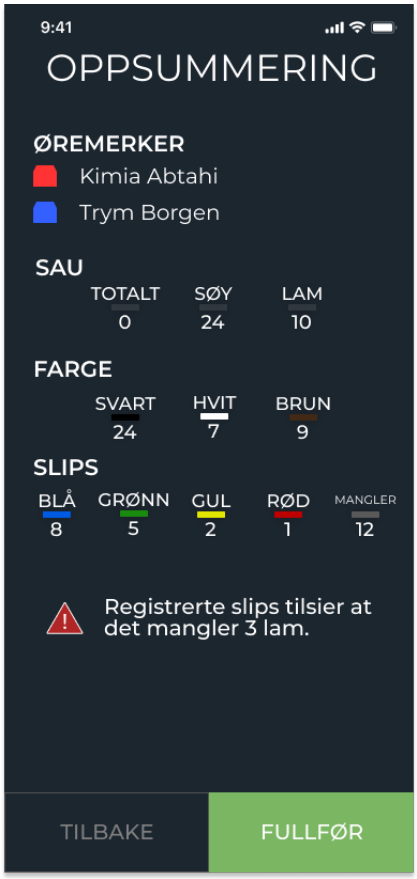
Figur 11.9: Prototypedesign av side for å legge til øremerker i en registrering

For å legge til et nytt øremerke i lista trykker brukeren på "Legg til nytt øremerke"-knappen (figur 11.9). Det vil åpne et annet grensesnitt hvor brukeren kan registrere et nytt øremerke (figur 11.10). I "Eier"-feltet kan brukeren skrive inn navnet på bonden som eier sauene som bærer det nye øremerket. Fargen(e) på merket legges til i fargevalg-grensesnittet under. Mattilsynets regler for øremerker [32] sier at alle farger er tillatt, med unntak av hvit, lilla og lakserød. Etter en undersøkelse av flere kilder [150, 151, 152] for beitelag og saumerker virker det som om de fleste øremerker består av enten en eller to farger. Den foreslåtte løsningen tillatter at det legges til maks to farger per øremerke. Fargevelgeren gjør det mulig å velge mellom alle mulige fargekombinasjoner som måtte være ønskelig for brukeren.



Figur 11.10: Prototypedesign av side for å registrere et nytt øremerke

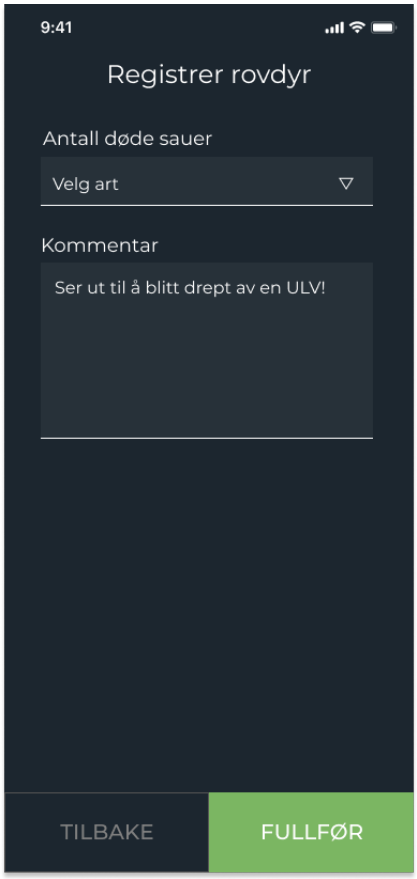
Oppsummeringssiden (figur 11.11) for registrering av sau har også blitt oppdatert for å vise hvilke øremerker som har blitt valgt under en registrering. Ellers er den identisk med oppsummeringssiden som ble laget under fordypningsprosjektet.



Figur 11.11: Prototypedesign av side for oppsummering for registrering av sau

Registrering av rovdyr

Brukeren kan trykke på det røde ikonet i valgmenyen med et poteavtrykk (figur 11.7) for å registrere et rovdyr. Herfra navigeres applikasjonen til registreringssiden for rovdyr (figur 11.12). Rovdyrene brukeren kan legge til er forhåndsbestemt og kan enkelt velges fra nedtrekkslisten. Valget av rovdyr begrunnes i underkapittel 3.1.3. I tillegg til dette kan brukeren legge til en kommentar om ønskelig. Brukeren kan fullføre registreringen ved å trykke på "Fullfør"-knappen. Registreringen kan også avbrytes ved å trykke på "Tilbake"-knappen som tar brukeren tilbake til kartsiden.



Figur 11.12: Prototypedesign av side for registrering av rovdyr

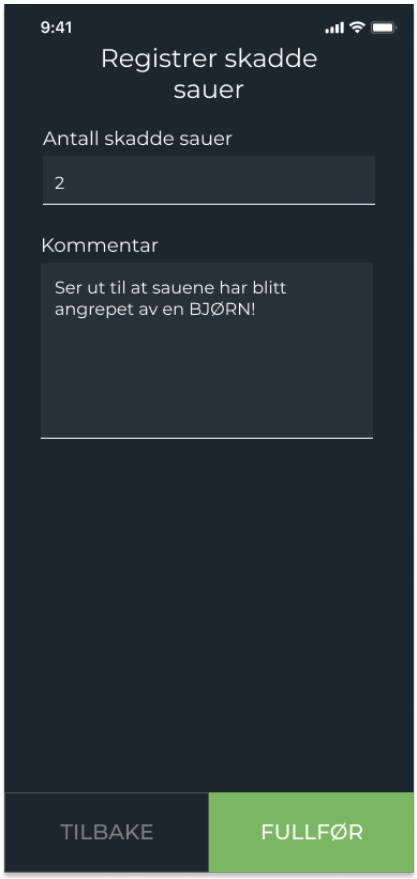
Registrering av døde sauer

For å registrere død sau trykker brukeren på den mørkegrå knappen med et dødninghode i valgmenyen på kartsiden (figur 11.7). Deretter blir brukeren tatt til registreringssiden for døde sauer (figur 11.13). Antallet døde sauer registreres under feltet "Antall døde sauer". Brukeren kan også legge til en kommentar om ønskelig. I følge Statsforvalteren skal det ved mistanke om skade på sau som følge av rovdyr tas bilde av kadaveret som knyttes til GPS-koordinater [30, s.19]. Denne informasjonen blir brukt som grunnlag når det søkes om erstatning for tap av sau [7, 8]. Det er derfor lagd design for funksjonalitet som lar brukeren legge til et eller flere bilder som blir lagret sammen med registreringen. Etter at ønsket informasjon er fylt ut kan brukeren trykke på "Fullfør"-knappen for å lagre registreringen og gå tilbake til kartsiden. Hvis brukeren ikke ønsker å lagre registreringen kan det trykkes på "Tilbake"-knappen.

Figur 11.13: Prototypedesign av side for registrering av døde sauer

Registrering av skadde sauer

Den gule knappen med et plaster i valgmenyen på kartsiden (figur 11.7) kan trykkes på for å registrere skadet sau. Herfra blir brukeren tatt til siden for registrering av skadet sau (figur 11.14). Antall skadde sauer registreres under feltet "Antall skadde sauer". Det er også mulig å legge ved en eventuell kommentar. Når registreringen er fullført trykkes det på "Fullfør"-knappen for å lagre registreringen og gå tilbake til kartsiden. Hvis brukeren ønsker å forkaste registreringen kan det trykkes på "Tilbake"-knappen for å gå tilbake til kartsiden uten å lagre.

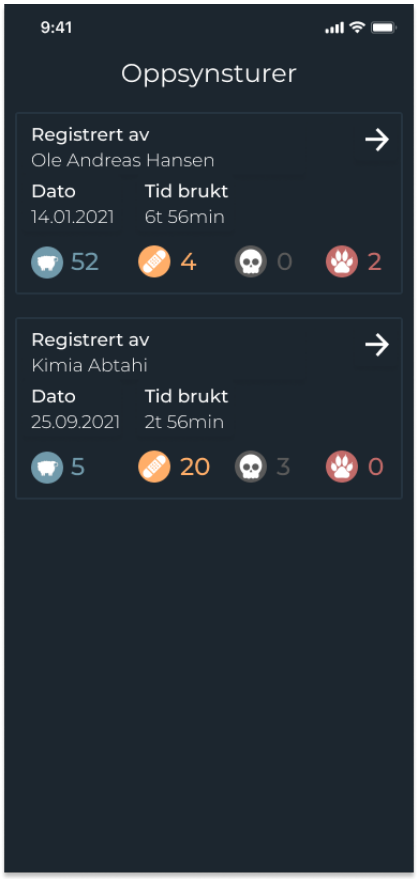


Figur 11.14: Prototypedesign av side for registrering av skadde sauer

11.2.6 Lagrede oppsynsturer

Liste over lagrede oppsynsturer

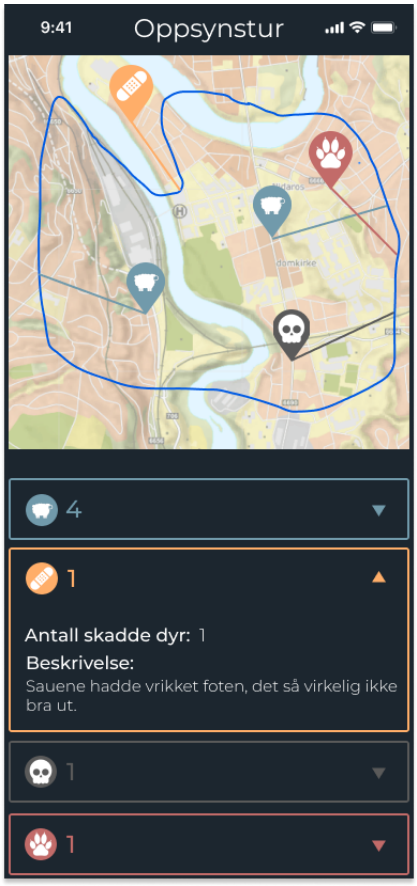
Fra hovedmenyen (figur 11.1) kan brukeren navigere seg til en liste over alle lagrede oppsynsturer (figur 11.15) ved å trykke på "Oppsynsturer"-knappen. Hvert element i lista viser en lagret oppsynstur og et utvalg av generell informasjon om turen, deriblant hvem som registrerte oppsynsturen, hvor lang tid det tok og hvilken dato den ble utført på. Symbolene nederst i hvert element viser (fra venstre) hvor mange sau som har blitt registrert totalt, hvor mange skadde sau som har blitt registrert, hvor mange døde sau som har blitt registrert og hvor mange rovdyr som har blitt registert.



Figur 11.15: Prototypedesign av side for lagrede oppsynsturer

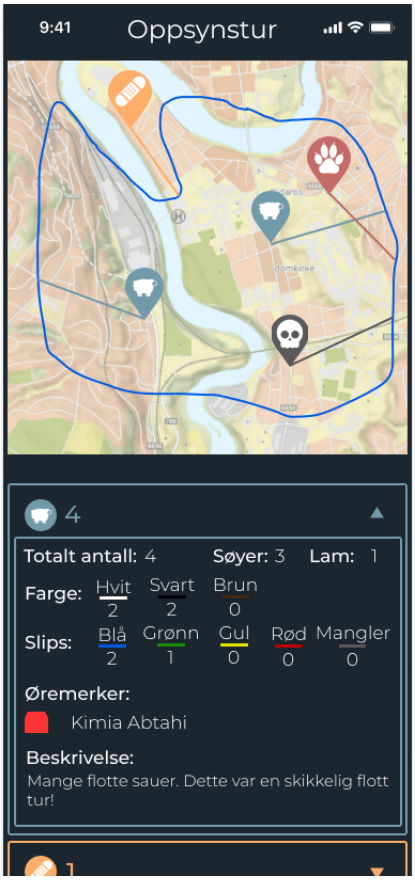
Oppsummering av en enkelt oppsynstur

Ved å trykke på pila i et element i lista over oppsynsturer (figur 11.15) blir brukeren tatt til siden som viser detaljert informasjon om den valgte oppsynsturen (figur 11.16). Siden viser ruta brukeren har gått på kartet og alle registreringer som har blitt gjort samt hvor de ble gjort fra. Under kartet er fire ekspanderbare elementer som kan åpnes og lukkes for å vise de forskjellige registreringene som har blitt gjort under hver av de fire kategoriene. Ikonet med tallet på siden viser det totale antallet sau, skadde sau, døde sau eller rovdyr som har blitt registrert under kategorien. Ved å trykke på pila til høyre i boksen ekspanderes lista, og alle registreringer som er gjort innenfor den valgte kategorien vises med all registert informasjon.



Figur 11.16: Prototypedesign av side for oppsummering av en lagret oppsynstur som viser registreringer for skadet sau

Ettersom at de forskjellige kategoriene inneholder forskjellig informasjon vil elementene inne i de ekspanderbare boksene være ulike fra kategori til kategori. Figur 11.17 viser hvordan informasjonen inne i den ekspanderte boksen for registrering av en saueflokk ser ut.



Figur 11.17: Prototypedesign av side for oppsummering av en lagret oppsynstur som viser registreringer for en saueflokk

11.3 Oppsummering

For å designe de ulike sidene i grensesnittet har design- og prototypeverktøyet Figma blitt brukt. Designet av de ulike sidene er laget for å dekke kravene stilt i kravspesifikasjonen. For å potensielt kunne spare strøm under bruk ble det bestemt å bruke et mørkt design i applikasjonen. Den valgte fargepaletten ble generert ved hjelp av verktøyet Coolors. Ikoner som ikke lot seg designe i HTML og CSS ble laget i Affinity Designer.

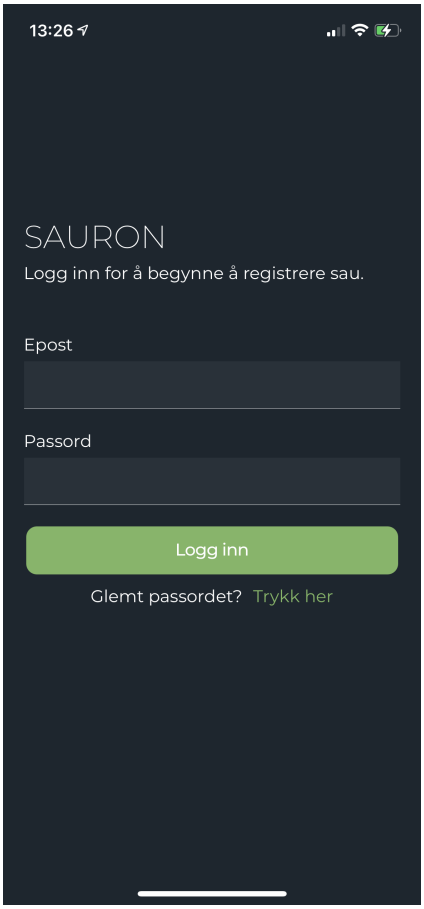
Kapittel 12

Utforming

Dette kapittelet tar for seg resultatene av utformingen av selve applikasjonen. Brukergrensesnittet er utviklet i henhold til prototype-designet (se kapt. 11) som ble laget i Figma under design-fasen av prosjektet. Det ble tidlig besluttet å ikke utføre brukertester på prototypedesignet som ble laget i Figma. Årsaken til dette var at det ble sett på som lite hensiktsmessig da applikasjonen i stor grad ville ta i bruk funksjonalitet som ikke kunne testes i prototypeverktøyet. Her tenkes det spesifikt på bruk av kart og GPS, som er en av hovedfunksjonalitetene i applikasjonen. Implementasjonen ble derfor startet på direkte etter at prototypedesignet i Figma var ferdigstilt. Alle figurene vist i dette underkapittelet er skjermbilder hentet fra iOS-versjonen av *Sauron* på en iPhone 12 Pro.

12.1 Innlogging

Innloggingssiden er den eneste siden som ikke har et prototype-design laget i Figma. Dette er fordi det først var tenkt å bruke et verktøy som tilbyr ferdiglagde innloggingsgrensesnitt i Angular med Firebase kalt *FirebaseUI-Angular* [153]. Grunnet problemer under implementasjonen av FirebaseUI-Angular ble denne idéen skrinlagt og en standard innloggingsside som bruker e-post og passord ble heller implementert. Logikken og funksjonaliteten rundt selve innloggingen blir forklart i kapt. 13.2.1.

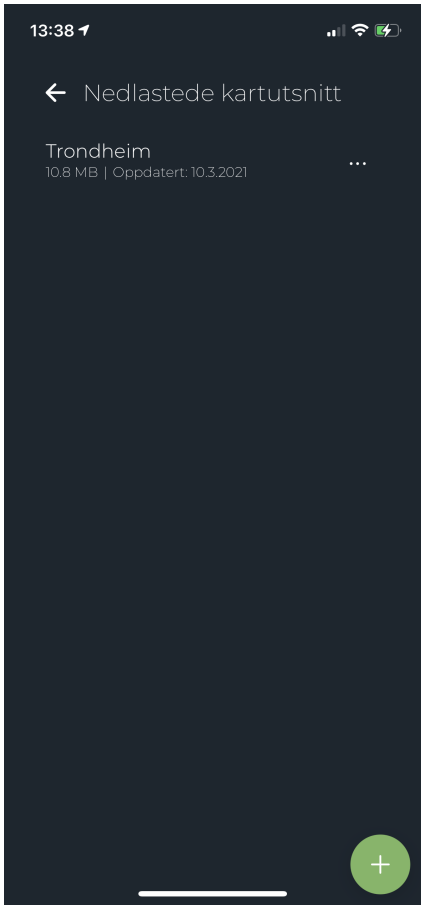


Figur 12.1: Innloggingsside for applikasjonen

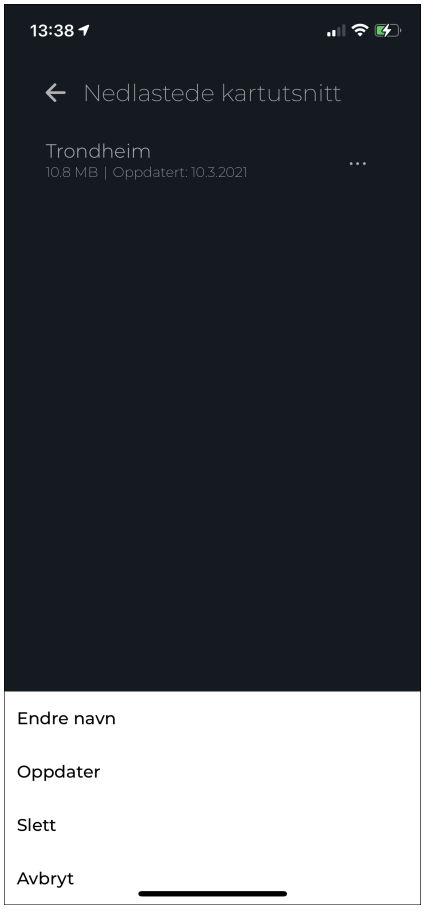
12.2 Nedlasting av kartutsnitt

12.2.1 Oversikt over nedlastede kartutsnitt

Siden som viser en liste over alle nedlastede kartutsnitt er utformet i henhold til design-prototypen. Det er også valgmenyen (se figur 12.3) som kommer opp hvis brukeren trykker på de tre prikkene til høyre for hvert listeelement. Under intern testing av grensesnittet viste det seg at det var behov for en navigasjonsknapp som gjør det mulig å navigere tilbake til hovedmenyen om ønskelig. Denne har blitt implementert i form av en pil som peker til venstre oppe i det venstre hjørnet av grensesnittet (se figur 12.2).



Figur 12.2: Oversikt over nedlastede kartutsnitt

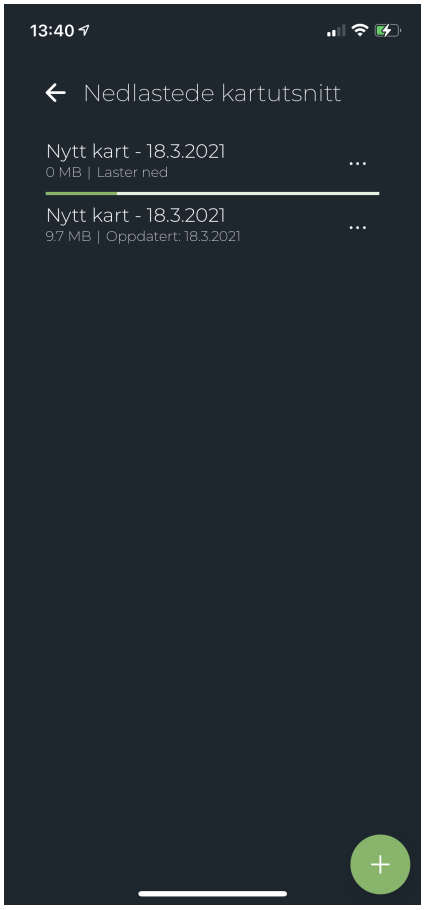


Figur 12.3: Valgmeny åpen for et nedlastet kartutsnitt

12.2.2 Laste ned nytt kartutsnitt

For å laste ned et nytt kartutsnitt trykker brukeren på den grønne plussknappen nede i høyre hjørne vist i figur 12.2. Figur 12.4 viser grensesnittet for nedlasting av nye kartutsnitt. Et nytt kartutsnitt kan lages ved å dra kartet slik at området det er ønskelig å ha med havner innenfor det markerte rektangelet. Man kan zoome inn og ut for å minske eller øke arealet som det skal lages kartutsnitt av. Grensesnittet er laget helt i henhold til design-prototypen med unntak av pilen for navigering tilbake til siden for nedlastede kartutsnitt. Når man trykker på "Last ned"-knappen starter nedlastingen og applikasjonen navigerer tilbake til siden med nedlastede kartutsnitt. Progresjon for nedlastingen vises i grensesnittet i form av en framdriftsindikator som fylles opp etterhvert som nedlastingen blir ferdig. Når nedlastingen er ferdig vil framdriftsindikatoren forsvinne og størrelsen på nedlastingen og dato den ble lastet ned vil presenteres til brukeren i elementet. Dette er funksjonalitet som ble designet og implementert etter intern testing da det viste seg at det var behov for en visuell indikator på progresjonen til nedlastingen.

Figur 12.4: Side for å laste ned et valg kartutsnitt



Figur 12.5: Progresjon for nedlasting av nytt kartutsnitt

12.3 Ny oppsynstur

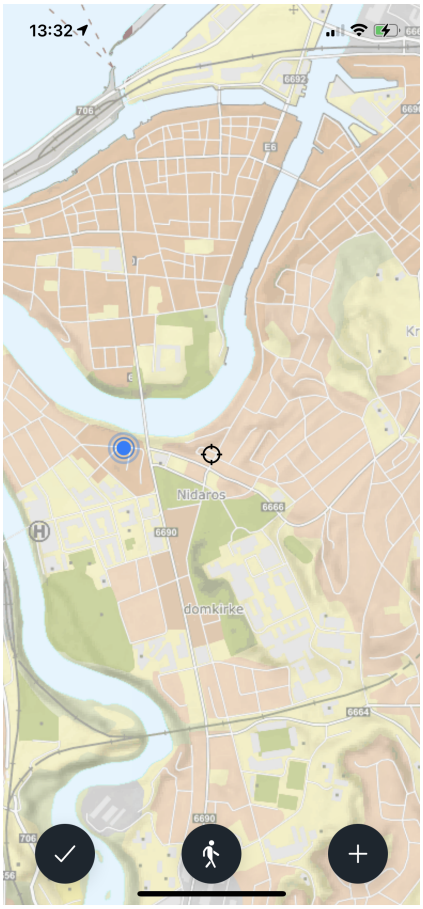
12.3.1 Registrering av informasjon for ny oppsynstur.

Grensesnittet for registrering av informasjon i sammenheng med en ny oppsynstur har blitt implementert i henhold til design-prototypen. Navnet til brukeren som allerede er logget inn legges til automatisk i listen over deltagere.

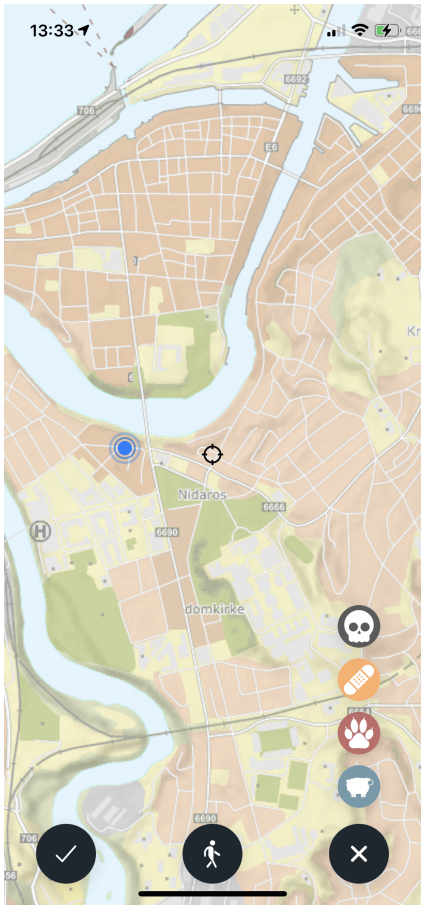
Figur 12.6: Implementert grensesnitt for registrering av informasjon før ny oppsynstur

12.3.2 Kart

Kartgrensesnittet som brukes under en en oppsynstur har blitt implementert i henhold til design-prototypen med visse endringer. For det første har størrelsen på symbolene blitt redusert for å gi mer plass til selve kartet. Markøren som viser hvor en registrering blir plassert, som vist i midten av kartet på figur 12.8, har blitt byttet ut med en tydeligere markør som gjør det mulig se sentrum av hva som registreres. Det har også blitt lagt til en knapp nederst i midten av grensesnittet. Denne knappen kan brukes til å sette applikasjonen i strømsparingsmodus som forklares nærmere i kapt. 14.3.



Figur 12.7: Implementert grensesnitt for interaksjon med kart under registrering

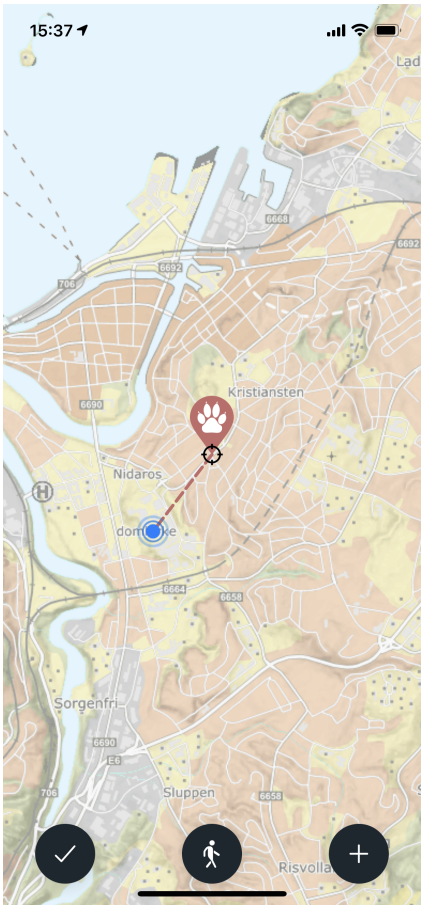


Figur 12.8: Kart-grensesnitt med åpen meny for å legge til en ny registrering

Kartgrensesnittet tar i bruk *Norgeskart bakgrunn cache* [154] fra Geonorge som er utviklet av Kartverket. Kartinformasjonen brukes sammen med JavaScript-biblioteket Leaflet [101] for å vise fram kartet. Kartfilene i *Norgeskart bakgrunn cache* oppdateres daglig slik at tilsynsansvarlige og bønder alltid vil ha tilgang til de nyeste kartbildene under en oppsynstur. Den blå markøren på kartet viser GPS-posisjonen til brukeren av applikasjonen. Markørens posisjon oppdateres i sanntid. Brukeren kan navigere kartet ved å dra det rundt med bruk av én finger. "Pinch-zoom"-funksjonalitet er innebygd i Leaflet [155] og tillater å zoome inn og ut ved hjelp av to fingre. Hvis mobiltelefonen ikke er koblet til internett vil applikasjonen automatisk velge det nedlastede kartutsnittet som passer best for den nåværende lokasjonen til mobiltelefonen. Hvis ingen nedlastede kartutsnitt passer vil brukeren få tilbakemelding om dette. Applikasjonen har også implementert funksjonalitet for å skifte mellom ulike nedlastede kartutsnitt hvis dette er nødvendig. Logikken bak dette forklares nærmere i kapt. 14.2.

For legge til en ny registrering i kartet kan brukeren trykke på pluss-knappen nede i høyre hjørne av skjermen. Dette vil åpne valgmenyen (se figur 12.8) for en ny registrering der brukeren kan velge mellom å registrere en saueflokk, et rovdyr, skadde sau eller døde sau. Registreringsmarkøren i midten av kartet plasseres over den posisjon hvor man har gjort en observasjon. Denne posisjonen brukes sammen med

brukerens GPS-lokasjon og type registrering (sau, død, skadet eller rovdyr) for å kunne legge til registreringen i kartet. Figur 12.9 viser hvordan kartgrensesnittet ser ut rett etter registrering av et rovdyr. En pin blir lagt til på kartet på posisjonen hvor observasjonen ble gjort. En striplet linje peker fra GPS-lokasjonen til brukeren under observasjonstidspunktet og fram til pinnen. Både pinnen og den striplede linjen er fargekodet for å gjøre det lettere å skille mellom ulike observasjoner.

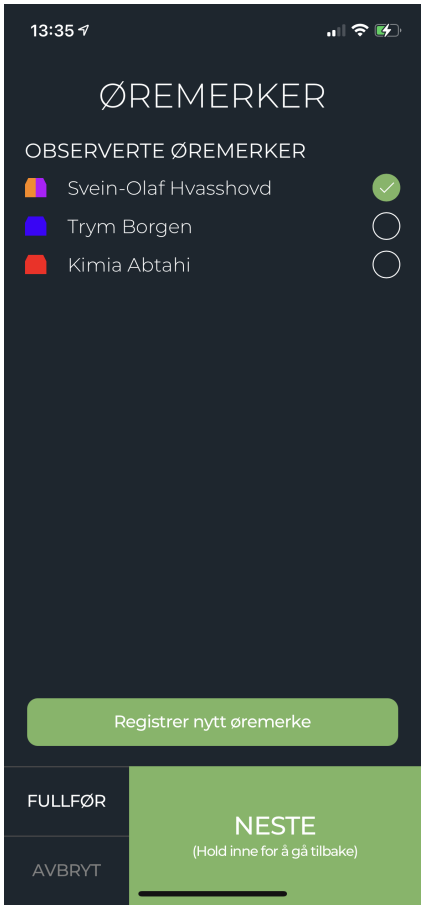


Figur 12.9: Eksempel på hvordan kartgrensesnittet ser ut etter at det har blitt registrert et rovdyr

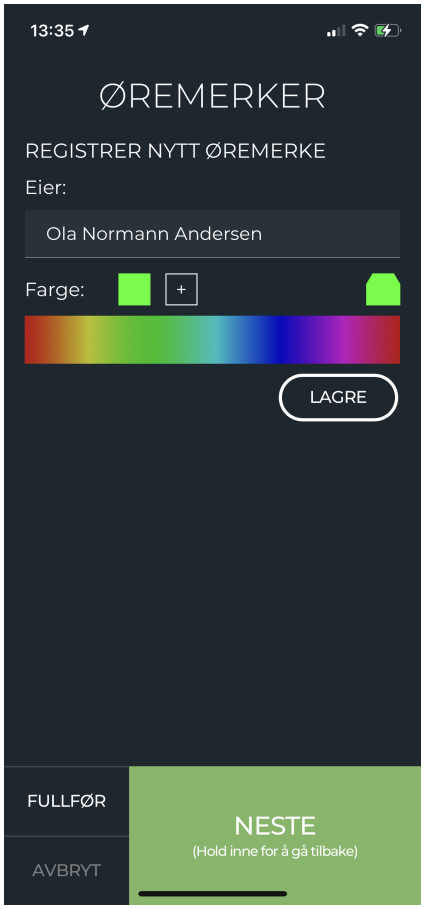
12.3.3 Registrering av en saueflokk

Grensensittet for registrering av en saueflokk nås ved å trykke på ikonet som viser en sau i figur 12.8. Grensesnittet er utviklet med bakgrunn i funnene gjort i fordypningsprosjektet [1, s.54-58]. Den implementerte løsningen tar i bruk en kombinasjon av ulike gester og virtuelle knapper, tekst-til-tale og haptisk tilbakemelding som mulighør blind registrering av sau. Funksjonaliteten nevnt i *Videre arbeid* i fordypningsprosjektet [1, s.64] har blitt implementert. For en full gjennomgang av grensesnittet henvises det til kapt. 7 i fordypningsprosjektet [1, s.44-52].

Den nye funksjonaliteten i forbindelse med registrering og oppretting av øremerker har blitt utformet i henhold til designprotoypene.



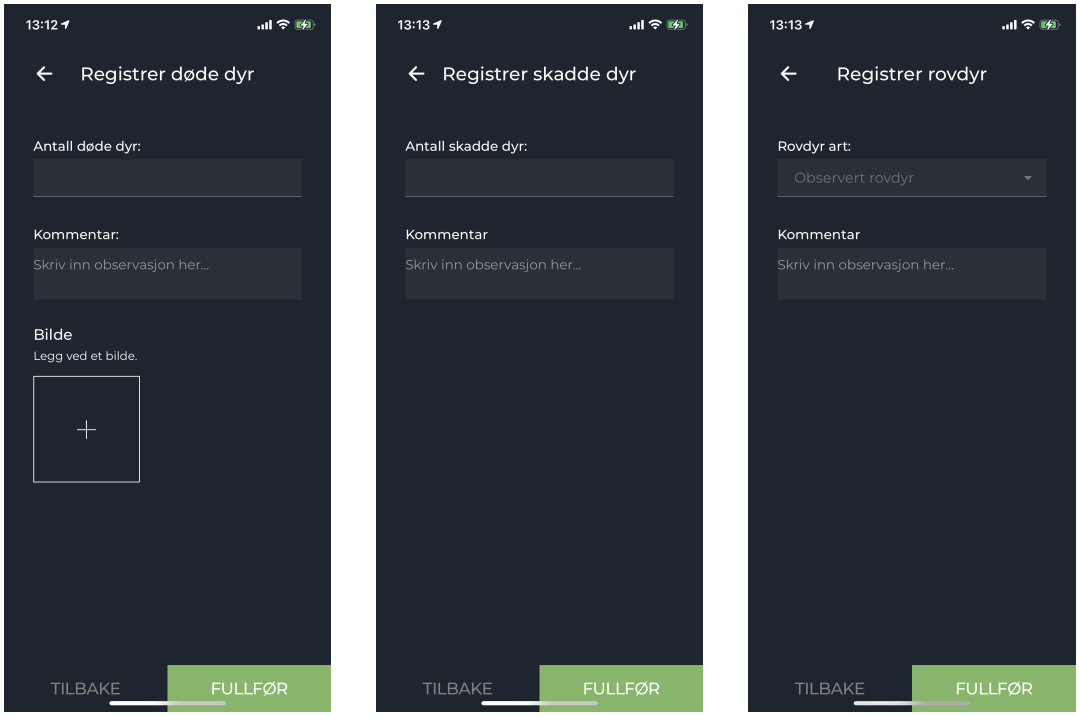
Figur 12.10: Grensesnitt for å registrere ett eller flere øremerker



Figur 12.11: Grensesnitt for å legge til et nytt øremerke

12.3.4 Registrering av skadet sau, død sau og rovdyr

Alle de tre grensesnittene for registrering av skadet sau, død sau og rovdyr har blitt implementert i henhold til designprototypene. Disse kan nås ved å trykke på henholdsvis plaster-ikonet, dødninghode-ikonet og pote-ikonet fra kartsiden. I grensesnittet for registrering av død sau vist i figur 12.12a har det blitt lagt til funksjonalitet for å legge til ett eller flere bilder. Ved å trykke på firkanten med et pluss i får brukeren valget om å ta ett nytt bilde eller velge et eksisterende fra mobilens kamerarull.



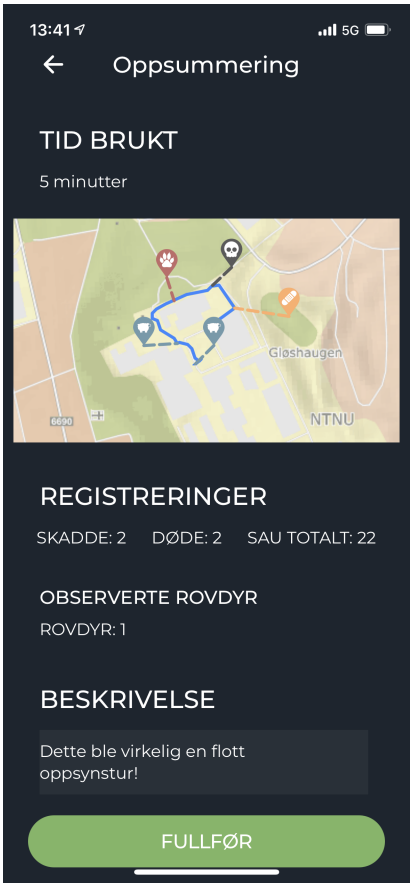
(a) Grensesnitt for registrering av død sau (b) Grensesnitt for registrering av skadet sau (c) Grensesnitt for registrering av rovdyr

Figur 12.12: Implementerte grensesnitt for registrering av rovdyr, skadet sau og død sau

12.3.5 Fullføre oppsyntur

For å fullføre en oppsynstur trykker brukeren på fullfør-knappen nede til venstre i kartgrensesnittet (se figur 12.9). En dialogboks, som vist i figur 12.13, vil da komme fram hvor brukeren må bekrefte handlingen. Hvis brukeren trykker "Ja" vises oppsummeringssiden fram (se figur 12.14).

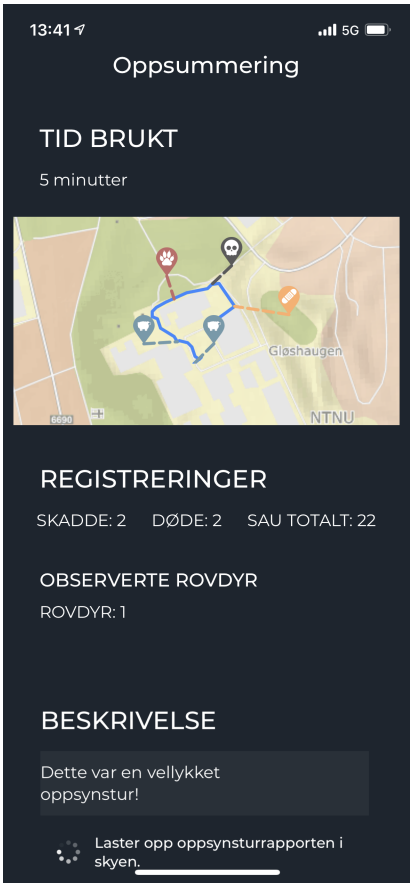
Figur 12.13: Dialogboks som kommer opp hvis brukeren trykker på fullfør-knappen



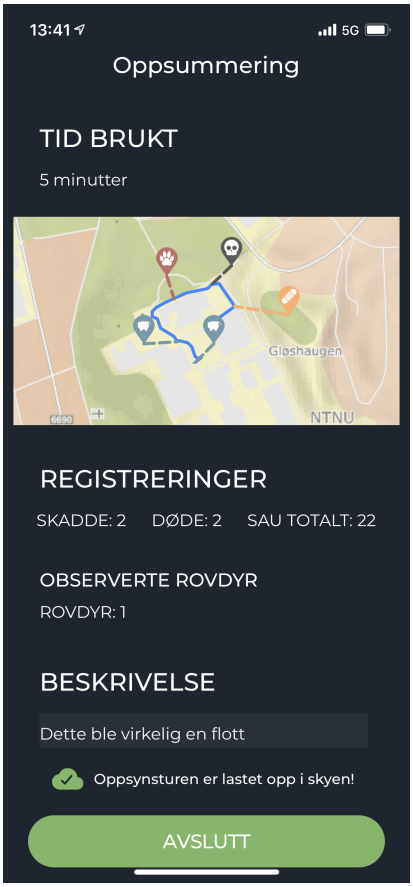
Figur 12.14: Grensesnitt som viser oppsummering av en fullført oppsynstur

12.3.6 Oppsummering av en fullført oppsynstur

Figur 12.14 viser grensesnittet for oppsummering av en fullført oppsynstur. Sammenlignet med oppsummeringssiden fra designprototypen har det blitt lagt til et felt som også viser informasjon om observerte rovdyr. Ved å trykke på "Fullfør"-knappen vil et lasteikon og en forklarende tekst som forteller at oppsynsturen lastes opp i skyen (se figur 12.15) bli vist fram. Etter fullført opplasting kan vil brukeren ha mulighet til å avslutte oppsynsturen som vist i figur 12.16.



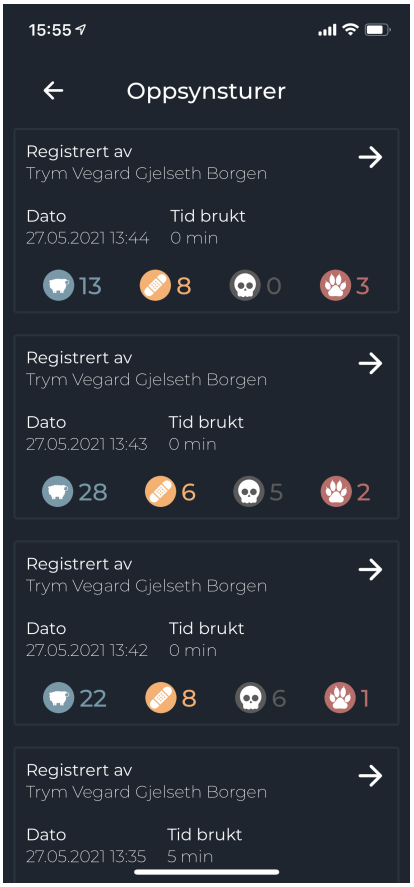
Figur 12.15: Spinner som forteller brukerne at oppsynsturen blir lastet opp i skyen



Figur 12.16: Grensesnitt for oppsummering etter fullført opplasting

12.4 Opplastede oppsynsturer

En liste over alle opplastede oppsynsturer fra eget beitelag kan nås ved å trykke på "Oppsynsturer"-knappen fra hovedsiden. Figur 12.17 viser et eksempel på hvordan listen kan se ut når det har blitt lastet opp flere oppsynsturer. Grensesnittet er implementert helt i henhold til designprototypen. Ved å trykke på pilen oppe til høyre i en oppsynstur vil brukeren få fram en oversikt over den valgte oppsynsturen. Figur 12.18 viser en slik oversikt hvor brukeren har trykt for å ekspandere listen over alle observasjonene av saueflokker.



Figur 12.17: Liste over alle opplastede oppsynsturer

Figur 12.18: Oversikt over en opplastet oppsynstur

12.5 Oppsummering

De ulike grensesnittene i applikasjonen har blitt utformet i henhold til designet beskrevet i kapt. 11. Ny funksjonalitet og nytt design har blitt lagt til på de sidene hvor det under intern testing kom fram at det originale designet hadde mangler. En innloggingsside har blitt designet og implementert for å tillate autentisering. Kartet i applikasjonen tar i bruk tjenesten Norgeskart og fungerer sammen med mobilens GPS for å vise fram informasjon om rute og posisjon til brukeren. Det har blitt lagt til funksjonalitet for nedlasting av kartutsnitt som muliggjør bruk av kartet selv på steder hvor mobiltelefonen ikke har internettilgang. Registreringsgrensesnittet for registrering av en saueflokk har blitt implementert og forbedret med bakgrunn i funnene fra fordypningsprosjektet. Fullførte oppsynstur lastes opp i skyen ved endt oppsynstur.

Kapittel 13

Programvarearkitektur

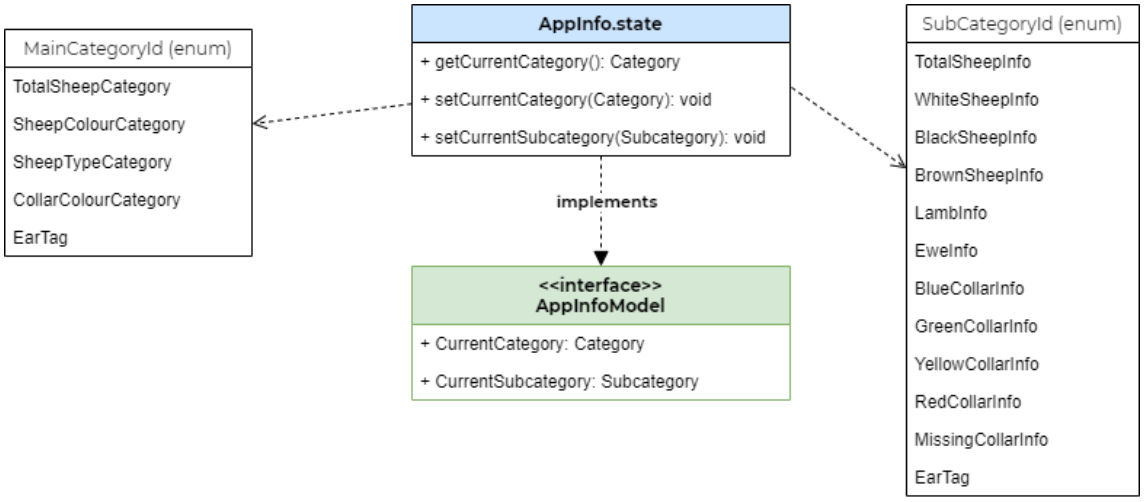
Dette kapittelet tar for seg løsningen for hvordan tilstandshåndteringen er implementert og hvordan applikasjonen samhandler med backendløsningen i Firebase.

13.1 Tilstandshåndtering

For at applikasjonen skulle fungere som ønsket var det viktig å få på plass et solid system for håndtering av tilstanden i applikasjonen. Med tilstandshåndtering menes det at det er programmatisk kontroll over tilstanden i applikasjonen og har mulighet til å lese eller endre denne. Hvis en del av applikasjonen endrer tilstanden, vil alle andre deler av applikasjonen få vite om dette. Tilstand kan, i dette tilfellet, for eksempel være hvor mange søyer med grønt slips som har blitt registrert. Applikasjonen må til en hver tid å ha kontroll over hvor brukeren befinner seg i grensesnittet under registreringen av en saueflokk, samt hvilke registreringer som har blitt gjort for å kunne lese opp korrekt tekst til brukeren under en blind registrering. For å løse dette ble det bestemt å bruke NGXS. NGXS er et rammeverk og et bibliotek for tilstandshåndtering som er godt brukt og spesielt laget for Angular (NGXS omtales mer i kapt. 10.2.2). Tilstandshåndteringen for applikasjonen er delt i tre, hvor en del tar seg av hvilken kategori og underkategori brukeren er i under registrering av en saueflokk. En annen del tar seg av hvor mange sauer som har blitt registrert under hver underkategori, og den siste delen tar seg av alle registreringen som har blitt gjort for en hel oppsynstur.

13.1.1 Tilstand for kategori og underkategori under registrering av saueflokk

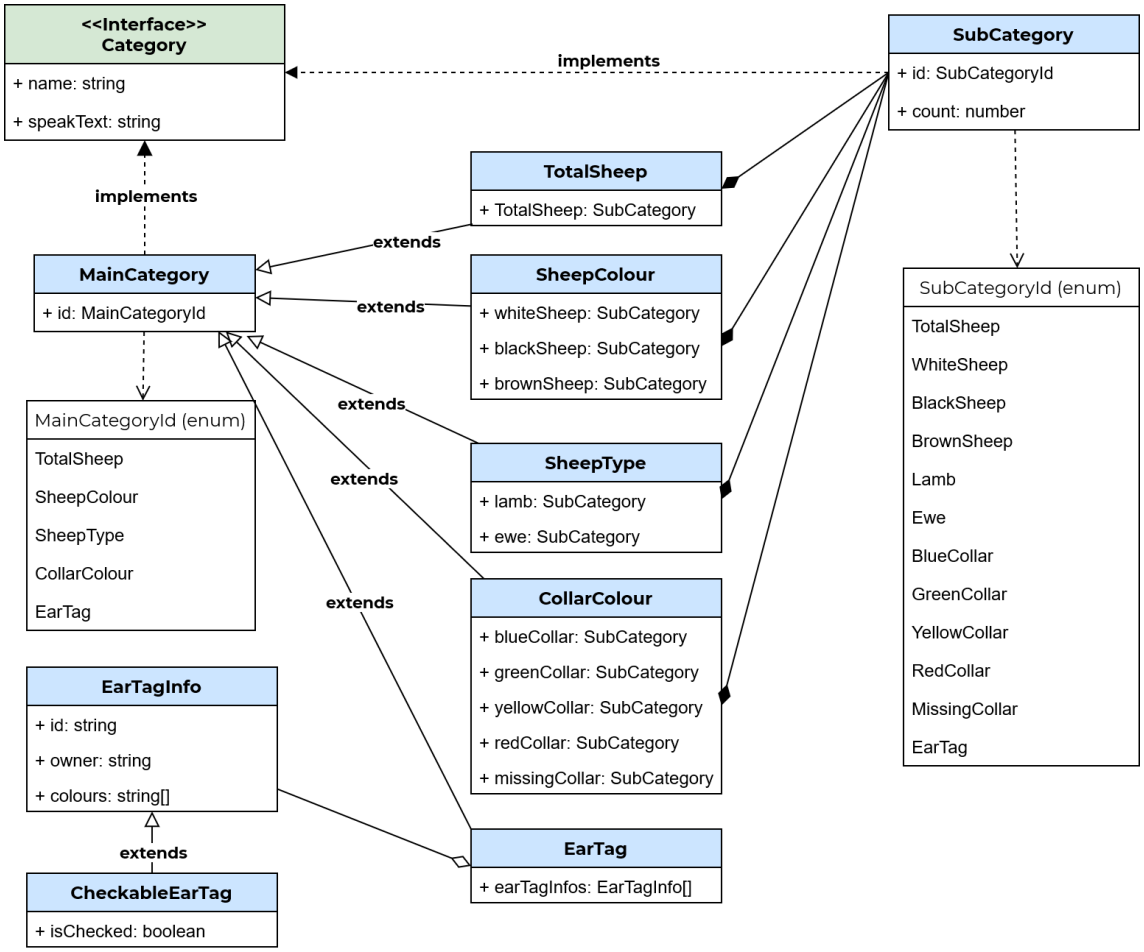
Tilstanden for kategori og underkategori under registrering av en saueflokk, kalt *AppInfo.state* består av bare to verdier. En verdi som forteller hvilken underkategori brukeren befinner seg på (*CurrentSubcategory*) og en som forteller hvilken hovedkategori brukeren befinner seg på (*CurrentCategory*). *AppInfo.state* inneholder også metoder for å endre og hente denne tilstanden og tilhørende informasjon.



Figur 13.1: Klassediagram for tilstandshåndtering av kategori og underkategori under registrering av en saueflokk

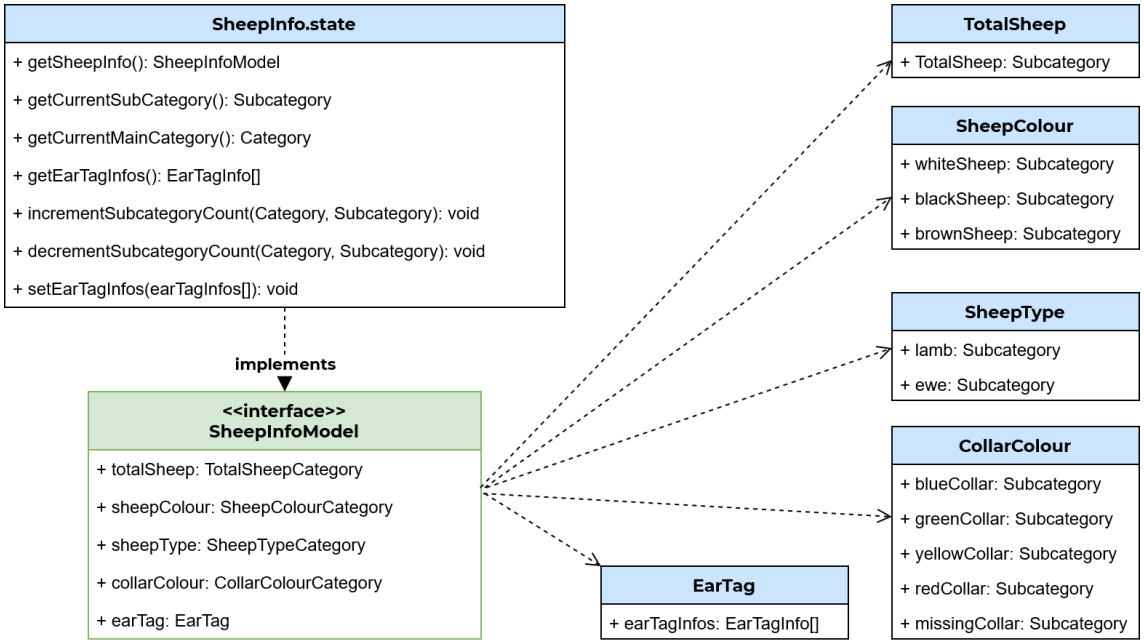
13.1.2 Tilstand for registrering av en sauflokk

Figur 13.2 viser et klassediagram for hvordan informasjonen for registreringen av en sauflokk blir delt inn i kategorier og underkategorier. Både klassen *SubCategory* og *MainCategory* implementerer interfacet *Category*. Dette gjør det enkelt å utvide disse klassene om det skulle være nødvendig. Det samme kan man si om klassene *TotalSheep*, *SheepColour*, *SheepType*, *CollarColour* og *EarTag* som alle arver fra *MainCategory*. Målet med å utforme klassene på denne måten var for å gjøre det enkelt å legge til nye kategorier og underkategorier i framtiden.



Figur 13.2: Klassediagram for tilstandshåndtering av opptalte sauer del 1

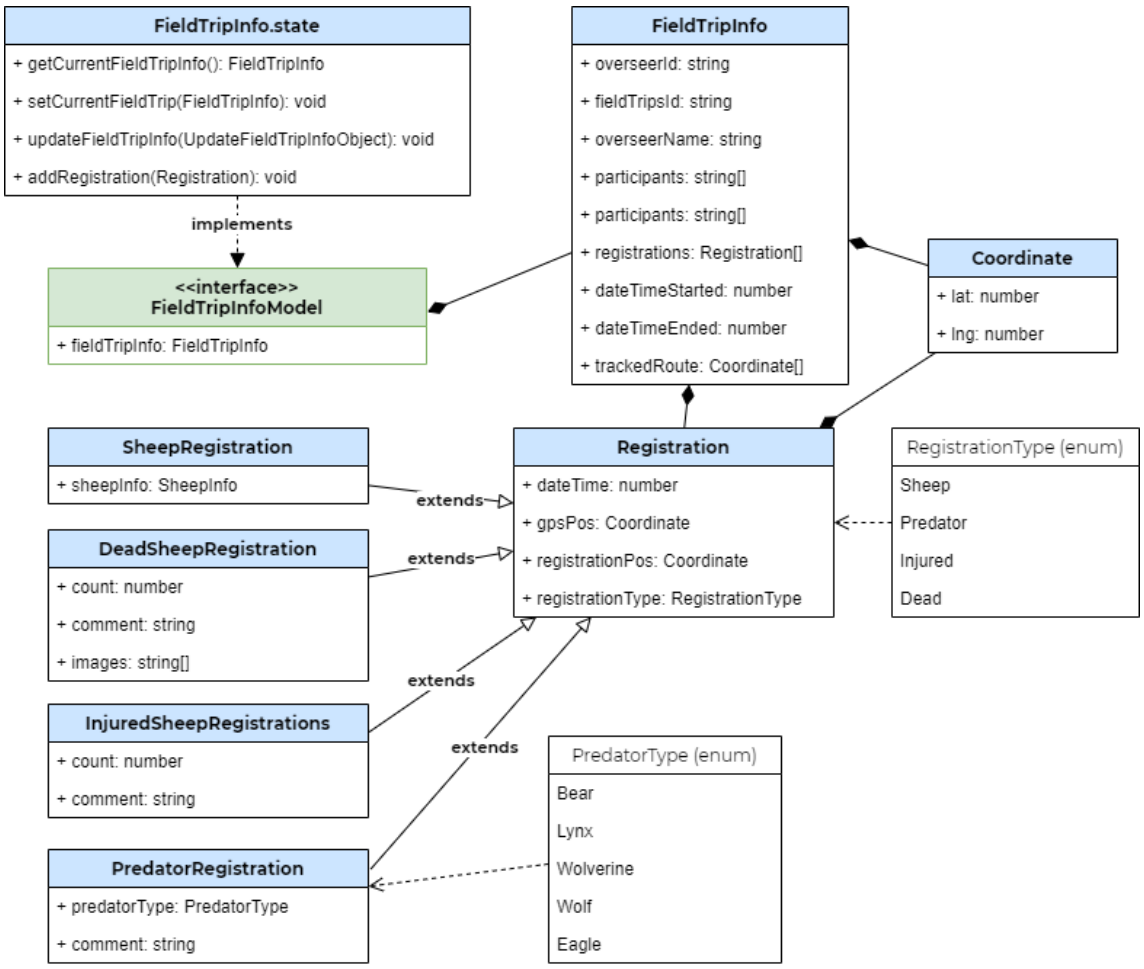
SheepInfo.state vises i figur 13.3. *SheppInfo.state* inneholder tilstanden for all de fem hovedkategoriene, samt forskjellige metoder for å gjøre det enkelt å inkrementere eller dekrementere antallet registrerte for hver underkategori.



Figur 13.3: Klassediagram for tilstandshåndtering av opptalte sauer del 2

13.1.3 Tilstand for en oppsynstur

I likhet med strukturen for registrering av en saueflokk er også strukturen for en oppsynstur laget modulært for å gjøre det mulig å for eksempel legge til en ny type registrering i framtiden. Klassen *FieldTripInfo* innholder all informasjon som er nødvendig i forbindelse med en oppsynstur. Den inneholder også alle registreringer som er gjort i form av en liste med objekter av typen *Registration*. Klassene *SheepRegistration*, *DeadSheepRegistration*, *InjuredSheepRegistration* og *PredatorRegistration* arver alle fra *Registration*, noe som gjør det enkelt å legge til en ny type registrering om ønskelig. *FieldTripInfo.state* inneholder én versjon av klassen *FieldTripInfo*, samt metoder for å hente, oppdatere og legge til nye registreringer.



Figur 13.4: Klassediagram for tilstandshåndtering av en oppsynstur

13.2 Backend

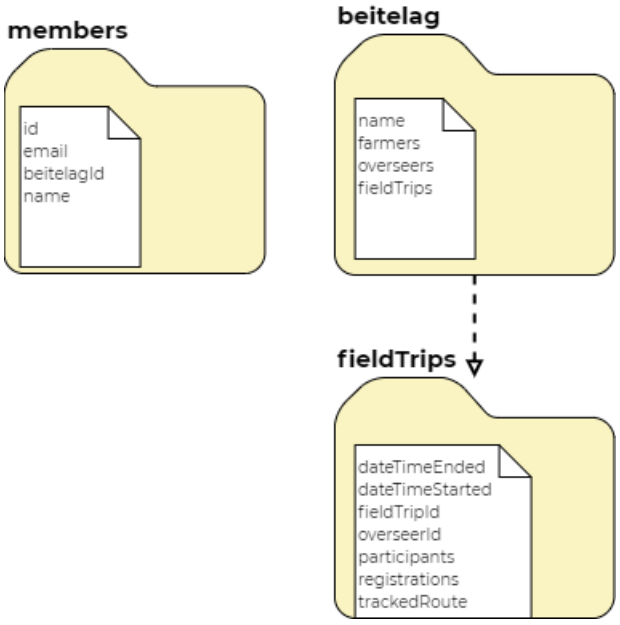
Framfor å bruke et mer tradisjonelt oppsett med en dedikert backend-tjener og tilhørende database, bruker applikasjonen Firebase som backend i form av Backend-as-a-Service. Ettersom at Firebase har løsninger for både autentisering, autorisering og database, betydde dette at vi kunne bruke mer tid på å implementere klientdelen av applikasjonen framfor å bruke tid på eventuell backend-logikk.

13.2.1 Innlogging

For innlogging brukes Firebase Authentication gjennom Firebase Cloud. Personer som har fått tildelt en bruker kan logge seg inn ved hjelp av e-post og passord. Firebase har løsninger som tillater innlogging ved hjelp av en tredjepart som for eksempel Google eller Facebook. Dette kan enkelt implementeres på et senere tidspunkt om ønskelig. Per dags dato kan nye brukere bare registreres av utviklere gjennom konsollen til Firebase.

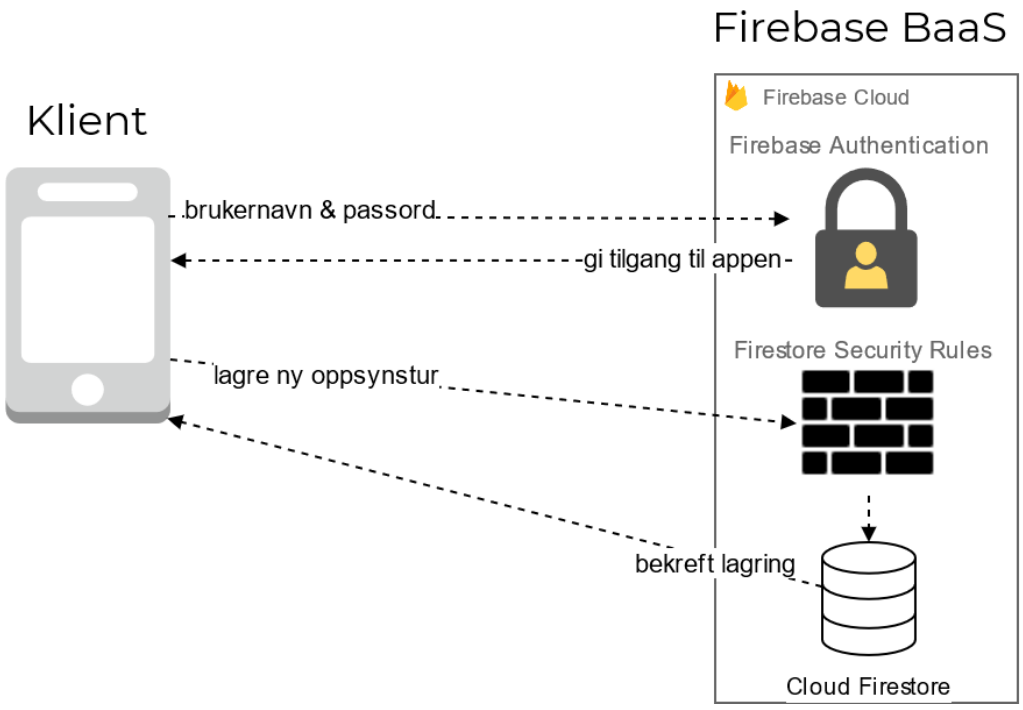
13.2.2 Database

For lagring av data i forbindelse med applikasjonen brukes Firebase Cloud Firestore. Cloud Firestore er en dokumentdatabase hvor dokumenter lagres i samlinger kalt *collections*. Databasen er delt opp i tre forskjellige samlinger; *members*, *beitelag* og *fieldTrips*. Samlingen *members* inneholder et dokument for hver bruker av applikasjonen med informasjon om bruker-id som er knyttet til Firebase Authentication, navn, beitelagId og e-postadresse. Samlingen *beitelag* inneholder et dokument for hvert beitelag som registreres i applikasjonen. Per nå kan nye beitelag bare registreres av utviklere gjennom konsollen til Firebase. Et beitelag innholder fire forskjellige data; navn, bønder (*farmers*), gjetere (*overseers*) og oppsynsturer (*fieldTrips*). Datafeltet *farmers* inneholder en liste med id-er over brukere som er registrert som bønder i et beitelag. En bonde vil ha tilgang til å se på alle oppsynsturer registrert av alle brukere for et beitelag. I motsetning til dette vil brukere registrert med sin id i *overseers*-listen for et beitelag bare ha tilgang til å se oppsynsturer registrert av seg selv. Logikken for dette ligger i Firestore Security Rules og håndteres på tjenersiden av Firebase. Hvert beitelagsdokument peker til en egen *fieldTrips*-samling som inneholder alle oppsynsturene som er registrert for det gitte beitelaget.



Figur 13.5: Databasestrukturen for Cloud Firestore

Figur 13.6 viser en flyt for innlogging og registrering av en oppsynstur ved hjelp av Firebase. Når en bruker åpner applikasjonen på mobilen sin (klient) vil brukeren bli bedt om å oppgi e-post og passord. Dette sendes til Firebase Authentication som sjekker om opplysningene er korrekte og deretter gir brukeren tilgang til applikasjonen. Når brukeren skal laste opp en oppsynstur går kallet gjennom Firestore Security Rules. Her sjekkes det som brukeren er logget inn, om oppsynsturen inneholder alle nødvendige data og om brukeren er den den oppgir seg for å være og er registrert i det oppgitte beitelaget. Om alt er som det skal lagres oppsynsturen i Cloud Firestore som et dokument i *fieldTrips*-samlingen. En bekreftelse sendes deretter tilbake til brukeren og forteller om alt gikk som det skulle.



Figur 13.6: Diagrammet viser interaksjon mellom BaaS og en klient for innlogging og lagring av en ny oppsynstur

13.3 Oppsummering

Programvarearkitekturen som har blitt implementert er laget for at den både skal fungere nå, men også for at den skal være lett å utvide og vedlikeholde i framtiden. Tilstandshåndteringen i applikasjonen blir tatt hånd om av rammeverket NGXS og sørger for at spesielt tekst-til-tale-funksjonalitet fungerer som ønskelig. Firebase brukes som backend i form av Backend-as-a-Service. Dette har resultert i kortere utviklingstid uten å gå på bekostning av funksjonalitet.

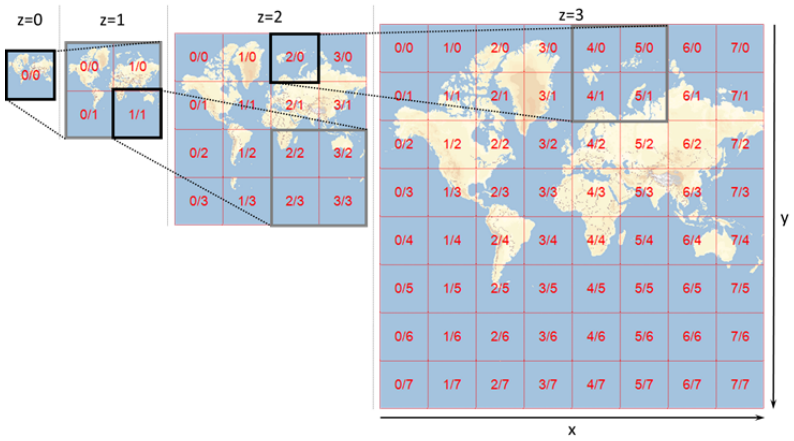
Kapittel 14

Programlogikk

Dette kapittelet tar for seg logikk som har blitt implementert i forbindelse med funksjonalitet knyttet til nedlasting av kart for bruk uten internettilgang og bakgrunnsoppdatering av GPS-lokasjon.

14.1 Nedlasting av kart for bruk uten internettilgang

For å laste ned et kartutsnitt brukes cache-tjenesten til Geonorge i form av *opencache*-endepunktene. Cache-tjenestene bygger på underliggende WMS-tjenester, noe som gjør dem svært brukbare i webapplikasjoner [102]. Kartet er lagret på tjeneren i form av fliser. Med dette menes det at kartet er delt inn i rektangulære bilder med ulikt detaljnivå basert på hvor zoomet inn kartet er. For å få tak i en bestemt flis må det oppgis x- og y-koordinaten til flisen, samt zoom-nivå. Et eksempel på et kall til tjenesten er: `https://opencache.statkart.no/gatekeeper/gk/gk.open_gmaps?layers=norges_grunnkart&zoom=14&x=8665&y=4429`. Her hentes flisen med x-koordinat 8665, y-koordinat 4429 og zoom-nivå 14 ut fra tjeneren.



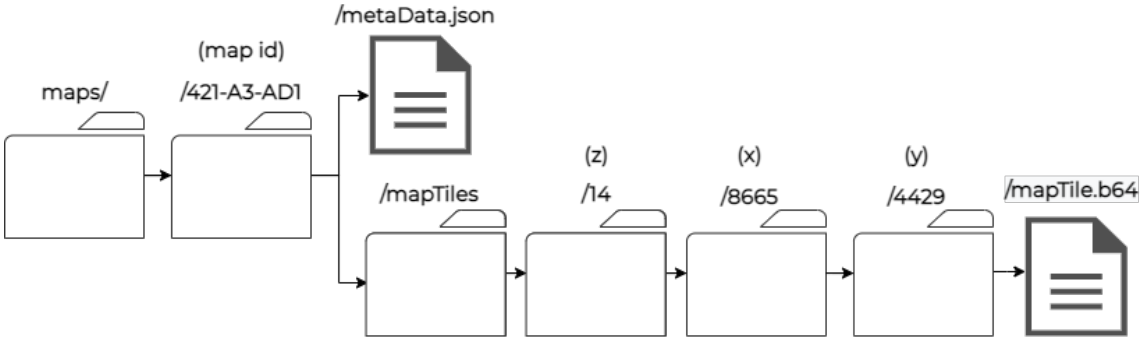
Figur 14.1: Illustrasjon av hvordan kartet er delt inn i fliser

Bildekilde: [156]

For at kartet skal fungere uten internett må hver eneste flis for det valgte kartutsnittet lastes ned og lagres på en måte som gjør det mulig å hente ut en bestemt flis basert på x- og y-koordinat og zoom-nivå. Denne prosessen starter med at piksel-koordinatene

for øvre venstre hjørne og nedre høyre hjørne for rektangelet som markerer kartutsnittet (se figur 11.4) gjøres om til GPS-koordinater. Det kan enkelt gjøres med en innbygd metode i Leaflet kalt containerPointToLatLng(x, y) [105]. Deretter kan GPS-koordinater gjøres om til flis-koordinater med en standardisert metode [157]. Under nedlastingen sendes forespørsler til Cache-tjenesten for å laste ned alle kart-flisene fra øvre venstre til nedre høyre hjørne, for hvert spesifiserte zoom-nivå. For å balansere lasten på tjeneren veksles det mellom å bruke endepunktene opencache.statkart.no, opencache2.statkart.no og opencache3.statkart.no.

For at Lealet skal klare å hente ut korrekt kart-flis fra det lokale filsystemet på mobiltelefonen, må kartflisene lagres på en måte som gir en URI lik den som benyttes når kart-flisene hentes direkte fra tjeneren. Figur 14.2 viser hvordan denne filstrukturen er delt opp. For å hente ut kart-flisen spesifisert i figuren bruker Leaflet URI-en `maps/421-A3-AD1/mapTiles/14/8665/4429/mapTile.b64`. En metadata-fil lagres også sammen med kart-flisene med informasjon som tilhører det nedlastede kartutsnittet. Metadata som lagres om kartet er kartutsnittets navn, størrelse (i bytes), nedlastingsdato og lokasjon. Denne metadata-filen brukes for å vise fram nyttig informasjon til brukeren, men også for å utføre intern logikk i programmet som for eksempel når et kart skal oppdateres.



Figur 14.2: Eksempel på lagringsstruktur for kartutsnitt med id 423-A3-AD1 og tilhørende kart-flis med koordinater z = 14, x = 8665 og y = 4429

14.2 Automatisk valg av kart for bruk uten internettilgang

Hvis brukeren starter en oppsynstur uten tilgang på internett vil applikasjonen automatisk velge ut det nedlastede kartutsnittet som passer best for området brukeren befinner seg i. For å gjøre dette hentes koordinatene fra midten av hvert nedlastede kartutsnitt. Deretter sammenlignes denne koordinaten med nåværende GPS-koordinat. Det kartet som har kortest avstand fra eget sentrum til nåværende GPS-koordinat blir valgt til å brukes. Hvis brukeren beveger seg utfor et nedlastet kartutsnitt mens en oppsynstur utføres vil algoritmen igjen forsøke å finne et annet nedlastet kartutsnitt som passer for brukerens nåværende GPS-koordinat.

14.3 Implementasjon av strømsparingsmodus

Under utvikling av den første prototypen av applikasjonen viste det seg at det ikke eksisterte et programtillegg for Capacitor som gjorde det mulig å implementere bakgrunnsoppdatering av GPS-lokasjon for både iOS og Android. Etter flere mislykkede

forsøk på å implementere denne funksjonaliteten på egenhånd, endte det til slutt opp med å skrinlegge idéen og heller legge til en funksjon som ville gjøre det mulig å spare strøm mens applikasjonen kjørte i forgrunnen. Ved å trykke på knappen nede i midten av kartgrensesnittet (se figur 12.9), ble strømsparingsmodusen aktivert. Skjermen ville da bli svart og en forklarende tekst som fortalte hvordan brukeren kommer seg ut av strømsparingsmodus ble vist fram. Ved å gjøre hele skjermen svart ville nyere mobilskjermer som tar i bruk OLED-teknologi kunne spare betydelig mengder strøm som ellers ville blitt brukt til å vise fram bilder på skjermen. I en undersøkelse gjort av Mian et al. [158] utforskes bruk av mørke energisparende brukergrensesnitt for OLED-skjermer. Resultatene viser at det var mulig å bruke 75% mindre strøm ved bruk av svarte og grønne grensesnitt framfor vanlige grensesnitt. Ettersom at den utviklede applikasjonen viste fram en helt svart skjerm under strømsparingsmodus kunne en anta at dette tallet ville være enda høyere for *Sauron*. Funksjonaliteten for strømsparingsmodus ble fjernet når bakgrunnsoppdatering av GPS-lokasjon ble implementert, se kapt. 14.4.

Figur 14.3: Skjermbilde av hvordan applikasjon ser ut rett etter at knappen for strømsparingsmodus har blitt trykket på

14.4 En sen løsning for bakgrunnsoppdatering av GPS-lokasjon

I slutten av april 2021 kom det et nytt programtillegg for Capacitor kalt *@capacitor-community/background-geolocation* [159]. Programtillegget gjør det mulig å hente

bakgrunnslokasjon for både iOS og Android. Tillegget har blitt implementert i applikasjonen og fungerer som ønsket.

14.5 Oppsummering

Kartet i applikasjonen tar i bruk cache-tjenestene til Geonorge i form av *opencache*-endepunktene. Det har blitt implementert en løsning for nedlasting av kart for bruk uten internettilgang. Løsningen tar i bruk en metadata-fil som gjør det enkelt å oppdatere nedlastede kartutsnitt, samt automatisk velge hvilke kartutsnitt som egner seg for bruk avhengig av brukeren GPS-lokasjon. Det ble implementert en strømsparingsmodus som kunne brukes da bakgrunnsoppdatering av GPS-lokasjon enda ikke var implementert. Når programtillegget *@capacitor-community/background-geolocation* ble publisert sent april 2021 ble strømsparingsmodusen forkastet og bakgrunnsoppdatering lagt inn i applikasjonen.

Del IV

Resultater

I denne delen blir teorien, planen og resultatene for brukertestene presentert.

Kapittel 15

Brukertest

I dette kapittelet blir teorien bak brukervennlighetstester kort presentert og valget av brukervennlighetstest begrunnes. Deretter beskrives forberedelsene og gjennomføringen av brukertestene i detalj.

15.1 Brukervennlighetstester

Det finnes ingen offisiell definisjon på hva som kjennetegner en brukervennlighetstest. Ekspert på brukeropplevelse (UX) Steve Krug definerer brukervennlighetstester som [160, s.13]:

> *Å observere folk som prøver å bruke det du designer/utvikler/bygger (eller noe som du allerede har designet/utviklet/bygget) med intensjonen om å gjøre det enklere for folk å bruke eller bevise at det er enkelt i bruk*.

Det er vanlig å kategorisere brukervennlighetstester i to ulike kategorier: kvalitative og kvantitative brukertester.

Kvalitative brukertester går ut på å innhente funn fra observasjoner som kan identifisere designfunksjoner som er enkle eller vanskelige å bruke [161]. Det betyr med andre ord at målet med slike brukervennlighetstester er å få innsikt som gjør at produktet som utvikles kan forbedres [160, s.14]. Ettersom målet ikke er å samle inn sammenlignbar data for å kunne bevise noe, er kvalitative brukertester gjerne mer uformelle i strukturen og protokollene kan gjerne endres på under gjennomføringen hvis det ses på som nødvendig [160, s.14]. Slike tester gjennomføres ofte med få brukertestdeltakere der de får oppgaver som skal løses mens de forteller hva de tenker når de samhandler med applikasjonen i en såkalt tenk-høyt protokoll, slik at observatørene kan notere hva deltakerene tenker om applikasjonen [162].

Kvantitative brukertester går ut på å innhente målbar data for å vurdere om oppgavene var enkle eller vanskelige å utføre [161]. Det vil si at målet med kvantitative brukertester er å bevise noe ved å samle inn data som kan måles, slik som suksessrate, tid eller antall feil for å kunne trekke en konklusjon [160, s.13] [161]. Derfor har slike brukertester en mye mer gjennomtenkt og strukturert testprotokoll som skal følges konsekvent for alle deltakere. Dette fører til et større behov for mange testdeltakere for å kunne trekke statistiske konklusjoner [160, s.13] .

I fordypningsprosjektet ble brukertester utført for å finne ut av hvilken interaksjonsmetode som var best egnet for registrering av sau i blinde. Dette ble gjort ved å samle data på hvor lang tid deltakerne brukte på å oppgavene, feilrate og tilbakemeldinger fra deltakerene hentet fra en spørreundersøkelse inspirert av System Usability Scale (SUS). Disse testene var dermed i hovedsak kvantitative brukertester med elementer fra kvalitative brukertester ettersom spørreundersøkelsen gitt til deltakerne ble utformet for å hente kvalitative tilbakemeldinger for å forbedre funksjonaliteten i applikasjonen til videre utvikling under masteroppgaven. Under brukertestene for masteroppgaven skal deltakerne teste hele systemet for å finne ut hva som kan forbedres i *Sauron*. Disse testene er derfor kvalitative brukervennlighetstester.

15.2 Gjennomføring av brukertest

Før gjennomføring ble det utformet en plan for brukertestene. Dette var for å forsikre at alle funksjonene som det var ønskelig å få tilbakemelding på skulle utføres av deltakerne og at samme rute ble fulgt ettersom applikasjonens GPS-funksjonalitet også skulle utprøves i testene. Det ble i tillegg bestemt at det ikke var nødvendig å teste grensesnittet for registrering i blinde ettersom det allerede var gjennomført og vurdert i fordypningsprosjektet.

15.2.1 Valg av testdeltakere

Med kvalitative brukertester er det ikke behov for mange testdeltakere ettersom de fleste store feil vil bli oppdaget tidlig og videre testing på flere deltakere vil ikke gi mer innsikt [160, s.43] [161, 163]. Det vanligste antallet testedeltakere for kvalitative tester er fem [160, s.43] [161, 163], og dette ble brukt som utgangspunkt for brukertestene for masteroppgaven. Opprinnelig var det ønsket å utføre brukertestene på minst én tilsynsansvarlig for å undersøke om applikasjonen innfridde deres behov under oppsynsturer. På grunn av den pågående koronaviruspandemien ble dette ikke prioritert (beskrives mer i kapt. 15.3). Dermed ble brukertestene utført av utviklerenes eksisterende nærkontakter. Enkelte av disse hadde vært med i gjennomføringen av brukertestene for fordypningsoppgaven. I disse brukertestene fikk testdeltakerne en demonstrasjon og innføring i hvordan registreringen fungerte i applikasjonen, og disse deltakerne var dermed godt kjent med dette grensesnittet. Dette ble sett på som en mulighet til å undersøke om det var en forskjell mellom brukerne som tidligere var kjent med grensesnittet for registrering og brukere som ikke fikk demonstrasjon i det hele tatt. Det ble derfor valgt ut to deltakere som ikke hadde kjennskap til applikasjonen og tre deltakere som hadde vært med på tidligere brukertester og var kjent med deler av brukergrensesnittet. Ellers ble ingenting annet lagret om testdeltakernes demografi som kjønn eller alder av hensyn til personvern.

15.2.2 Før testen

En av utviklerene vil få rollen som forsøksleder og den andre som observatør. Forsøkslederens ansvar er å informere deltakere om hvordan testen skal foregå og følge planen som er utarbeidet for gjennomføringen av brukertesten slik at det blir utført på riktig måte. Observatøren skal under hele testens forløp observere deltakerne og notere deres kommentarer på en strukturert og oversiktlig måte. For å forsikre at brukertestene ble startet på like premisser ble det utformet en sjekkliste som skulle utføres før selve brukertesten:

1. Fjerne tidligere oppsynsturer som er blitt gjennomført på testbrukeren.
2. Fjerne tidligere kartutsnitt.
3. Skru av skjermsparer.
4. Desinfisere mobiltelefonen (mer om smittevernstiltak er utdypet i kapt. 15.3).

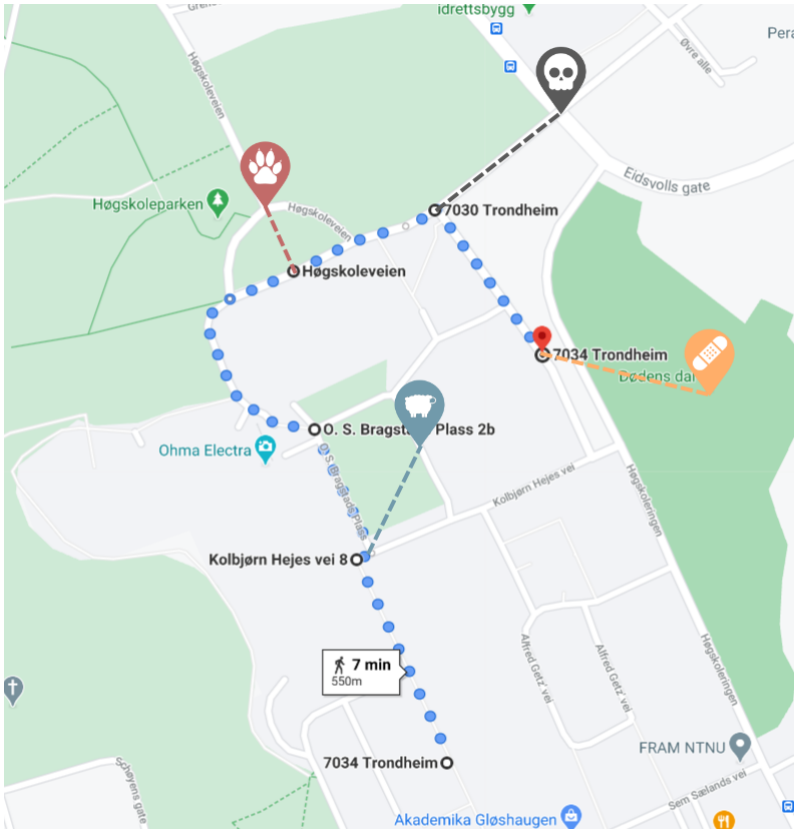
For å være sikker på at alle testdeltakerne fikk lik informasjon om hvordan testen skulle foregå, ble det også utformet et dokument som forsøksklederen leste opp før testen ble gjennomført:

1. Forsøksleder introduserer seg selv og observatøren.
2. Forsøksleder forklarer hensikten med brukerstesten:
   "Jeg vil at du skal se for deg at du er en sauebonde, og du skal sjekke hvordan det står til med saueflokken din som er på utmarksbeite for sommeren. Det er viktig at du raskt oppdager om det er skadet, død eller rovdyr på beitet slik at du kan griper inn med tiltak. På oppsynsturer skal du sjekke hvor mange sauer det er totalt, hvor mange sauer det er av hver farge, antall søyer og lam og antall sauer som har slips (som er en krage rundt halsen med farge som viser hvor mange lam en søye skal ha) og sauenes øremerker. Problemet er at disse sauene ofte går langt og beveger seg hele tiden, så det er en utfordring å få øye på dem. Derfor foregår registrering av sau som oftest mens man observerer sauene gjennom kikkert. I tillegg vil observasjoner av skadet, død sau eller rovdyr bli registrert dersom det oppdages på turen. I denne brukertesten skal du teste en applikasjon som skal bistå som verktøy for å registrere oppsynsturer for sau."

   Husk at det er systemet vi evaluerer og tester, ikke deg. Vi skal ikke teste hvor flink du er til å bruke appen, men hvor bra appen fungerer for deg.
3. Forsøksleder forklarer hvordan brukertesten gjennomføres:
   "Vi har laget en rute for en oppsynstur der vi vil gi deg oppgaver som du skal løse med appen underveis. Det vil være 8 oppgaver til sammen. Du kan bryte brukertesten når som helst. Si gjerne hva du tenker mens du løser oppgavene. Hvis du sliter og sitter fast med en oppgave, kan du spørre om hjelp fra forsøksleder, ellers vil ikke forsøksleder eller observatør kommentere underveis."
4. Deltakeren får mulighet til å stille spørsmål om noe er uklart.

15.2.3 Selve brukertesten

Brukertestene ble gjennomført ved at forsøksleder, testdeltakeren og observatøren gikk en planlagt rute ved campus Gløshaugen i Trondheim, se figur under. Forsøksleder ga testdeltakeren oppgaver underveis, og det var tilsammen åtte oppgaver med tilhørende deloppgaver som skulle utføres under turen:



Figur 15.1: Rute for brukertestene på Gløshaugen campus

1. Last ned kartutsnitt

1.1. Endre navnet på det nedlastede kartutsnittet.

2. Ny oppsynstur

2.1. Legg til alle deltakere som er med på turen (testdeltaker, forsøksleder og observatør).

2.2. Skriv inn beskrivelse om ønskelig.

3. Registrere sauer

- Gå til Kolbjørn Hejes vei 8. Registrer sauer ved toget på andre siden av plenen.

3.1. 5 sauer totalt.

3.2. 3 hvite, 1 svart og 1 brun.

3.3. 2 søyer og 3 lam.

3.4. 1 grønt og 1 gul slips.

3.5. Registrer nytt øremerke med navn og farge og velg dette øremerket.

3.6. Fullfør registreringen.

4. Registrer rovdyr

- Gå til 7034 Trondheim. Registrer rovdyr på fotballbanen.

4.1. Registrer observasjon av gaupe.

4.2. Skriv kommentar om ønskelig.

5. Registrer død sau

- Gå til trappene ved hovedbygningen. Registrer død sau ved krysset nedenfor.

5.1. 1 dødt dyr.
5.2. Skriv kommentar om ønskelig.
5.3. Ta et bilde.
5.4. Fjern bildet. (lagt til etter første brukertest var gjennomført, mer informasjon i neste kapittel)
5.5. Fullfør registrering.

6. Registrer skadet sau

- Gå til 7030 Trondheim. Registrer skadede sauer på broa rett fram.

6.1. Registrer 2 skadede sau.
6.2. Skriv kommentar om ønskelig.

7. Fullfør oppsynstur

7.1. Sjekk om registreringene stemmer.
7.2. Oppdater beskrivelse om ønskelig.
7.3. Trykk "Fullfør".

15.2.4 Etter testen

Etter testen ble testdeltakerne spurt om de hadde noen generelle tilbakemeldinger og deres helhetlige inntrykk av applikasjonen.

15.3 Gjennomføring av brukertest under Covid-19

Gjennomføringen av brukertestene har vært sterkt preget av den pågående Covid-19 pandemien for både fordypningsprosjektet og masteroppgaven. Selv om det var svært lite smitte i Trondheim da brukertestene skulle gjennomføres, ble det innført strenge nasjonale smittevernstiltak 23.03.21, dagen før det var planlagt å utføre brukertestene. Økt avstand og færre nære kontakter er to sentrale smittereduserende tiltak i håndteringen av koronapandemien ifølge Folkehelseinstituttet (FHI) [164]. Derfor ble det tatt noen forholdssregler og iverksatt smittevernstiltak under selve brukertestene slik at brukertestene ble gjennomført på en trygg måte og tilfredstilte FHI sine råd og anbefalinger. Det ble bestemt at alle testdeltakerne skulle være venner som allerede ble betegnet som utviklerens nærkontakter slik at ingen utenforstående eller utviklerne selv måtte bli utsatt for særskilt risiko. Dette medførte at alle testdeltakerne var medstudenter i 20-årene. For å få best mulig data ville det vært mer optimalt med en jevnere demografisk spredning av deltakerne og testing av faktiske sauebønder, men det ble uaktuelt i en slik smittesituasjon. I tillegg ble det også utført praktiske smittevernstiltak for å sørge for at brukertestene ble gjennomført på tryggest mulig måte. Mobiltelefonen som testdeltakerne benyttet seg av i brukertestene ble desinfisert før og etter testen og utviklerne sørget for å holde minst to meter avstand slik de daværende smittereglene anbefalte.

15.4 Oppsummering

Det ble utført brukervennlighetstester på *Sauron* for å få tilbakemeldinger på hvor godt egnet applikasjonen er til å bistå i registrering av sau på oppsynstur. Testene som ble gjennomført var såkalte kvalitative tester med fokus på å avdekke problemområder i designet og funksjonaliteten til applikasjonen. Det ble utformet en plan og rute for gjennomføringen av brukertestene som ble fulgt. Tilsammen fem personer ble testet. På grunn av den pågående Covid-19 ble det satt smittevernstiltak og begrensninger for brukertestene Dette påvirket demografien til testdeltakerne. For å minimere smitterisiko var alle testdeltakere bekjente som allerede var nærkontakter til utviklerne.

Kapittel 16

Resultater fra brukertestene

I dette kapittelet presenteres resultatene fra de fem brukervennlighetstestene som ble gjennomført. Dette omhandler de kvalitative resultatene fra observasjoner gjort av testdeltakerne mens de utførte oppgavene fra kapt. 15.2.3 og deres tilbakemeldinger til applikasjonen.

16.1 Kvalitative resultater

Resultatene beskrives her som et sammendrag av alle testdeltakernes prestasjoner og tilbakemeldinger oppgave for oppgave. Utfylte rapportskjema for hver brukertest kan leses i sin helhet i vedlegg C. Avslutningsvis blir alle resultatene presenteres i en tabell som fremhever oppgavene der testdeltakerne støtte på problemer i brukertestene.

16.1.1 Oppgave 1: Laste ned kartutsnitt

De aller fleste testdeltakerne klarte å utføre oppgave 1 uten problemer, både det å laste ned kartutsnittet og endre navnet på utsnittet etterpå. For en deltaker tok det svært lang tid for applikasjonen å vise kartutsnittet som var tilgjengelig for nedlasting, noe som førte til at deltakeren trykket på "Last ned"-knappen før kartet ble vist på skjermen. Dermed måtte deltakeren ut av siden for nedlasting av kart og så tilbake igjen for å få kartet til å laste. Samme deltaker gikk også ut av siden og klikket rundt på andre knapper da personen fikk i oppgave å endre navnet på kartsiden. Etter at forsøksleder presiserte at det var kartutsnitt og ikke oppsynstur som skulle endres navn på, klarte deltakeren å gjennomføre oppgaven.

16.1.2 Oppgave 2: Ny oppsynstur

Alle testdeltakerne klarte å opprette en ny oppsynstur og la til alle medlemmene som var med på oppsynsturen uten å støte på problemer. En av deltakerne la ikke merke til at personen allerede var lagt inn på oppsynsturen og la dermed til seg selv på nytt i tillegg til forsøksleder og observatør. Deltakeren fant selv ut at personen var lagt til dobbelt og fjernet seg selv. Denne personen kommenterte at "appen var veldig intuitiv og jeg skjønte alt selv jeg ikke vet hva en oppsynstur er".

16.1.3 Oppgave 3: Registrere sauer

Alle deltakerne bortsett fra én forstod ikke at markøren på kartet er der lokasjonen til en registrering vil bli plassert, og flyttet derfor ikke kartet slik at markøren pekte

på den eksakte lokasjonen til observasjonen. Tre av deltakerne forstod det etter å ha registert en observasjon. Da oppdaget de at det ble opprettet en pin der markøren var plassert. Disse testdeltakerne brukte markøren på riktig måte resten av brukertesten. En av deltakerne forstod det ikke gjennom hele brukertesten.

En deltaker ga uttrykk for at ikonene for registrering var for små og at det var lett for å trykke på feil ikon. Deltakeren mente også at det gjerne kunne være mer luft mellom selve ikonet og sirkelen rundt for estetikkens skyld. Samme bruker gav uttrykk for at rovdyrikonet kunne forveksles med et en hundepote og burde gjøres "farligere" for å tydeliggjøre at det er ikonet for rovdyr.

For en av deltakerne ble det ikke registrert at personen trykket på symbolet for registrering av sau. Deltakeren ble først forvirret, men skjønte etterhvert at personen bommet på knappen. Samme deltaker gikk også videre fra registrering av øremerke før det nye øremerket som personen hadde opprettet ble lagret ettersom tastaturet presset "Neste"-knappen opp til "Lagre"-knappen. Dette gjorde at ble det kort avstand mellom knappene og deltakeren trykket på "Neste" i stedet for "Lagre". Deltakeren oppdaget feilen i oppsummerings-siden, gikk tilbake og rettet opp i feilen selv. En annen deltaker gikk også videre til oppsummeringssiden uten å trykke på det nylig opprettede øremerket, men det var på grunn av at deltakeren ikke fikk med seg at man måtte huke av avkrysningsboksen for å registrere øremerket og ikke grunnet tastaturet som beskrevet for forrige deltaker.

I denne oppgaven ønsket utviklerne å undersøke om det var en forskjell på deltakerne som allerede var kjent med grensesnittet for registrering sammelignet med deltakerne som ikke hadde tidligere kunnskap. Begge de nye deltakerne brukte tid på å forstå at det måtte sveipes til høyre eller venstre og ikke trykkes for å bytte underkategori i registreringen. En av deltakerne fant ut av sveipingen på egen hånd etter litt prøving og feiling, resten av registreringen gikk deretter feilfritt. Den andre deltakeren som heller ikke hadde erfaring med registreringsgrensesnittet forsto ikke at det måtte sveipes for å bytte kategori, og måtte til slutt få hjelp av forsøksleder for å fullføre registreringen. Det var ingen problemer med registreringen for personene som var kjente med registreringsgrensesnittet fra før. En kjent deltaker lurte på hvorfor det var piler på sidene dersom de ikke kunne trykkes på, bare sveipes.

16.1.4 Oppgave 4: Registrere rovdyr

En av deltakerne opplevde at GPS-lokasjonen hoppet et stykke bort idet personen skulle registrere rovdyr, som førte til forvirring. Klarte likevel å fullføre registreringen slik det var ment.

16.1.5 Oppgave 5: Registrere død sau

En av deltakerne trodde symbolet med plaster, som er ment for skadet sau, symboliserte en død sau. Deltakeren oppdaget feilen da personen leste tittelen på siden for skadet sau og gikk tilbake til kartsiden. Deltakeren forstod deretter at det er dødninghodet som er symbolet for død sau og derfra gikk resten av registreringen greit.

To av deltakerne prøvde å trykke på bildeikonet i overskriften for bildetaking for å

ta bilde og ikke +-knappen. Begge deltakerne forstod det selv etter å ha prøvd seg fram på siden.

16.1.6 Oppgave 6: Registrere skadet sau

Alle deltakerne av brukertesten utførte registrering av skadet sau feilfritt. En deltaker sa også under registreringen: "Dette er litt gøy".

16.1.7 Oppgave 7: Fullfør oppsynstur

Den første testdeltakeren klarte å finne oppsummeringssiden til oppsynsturen som ment, men da personen trykket på "Fullfør"-knappen kom det en tilbakemelding fra applikasjonen at det forekom en feil i opplastingen av oppsynsturen til Firebase databasen. Utviklerne måtte derfor lage en falsk oppsynstur for denne deltakeren slik at siste oppgave ble gjennomført og brukertesten deretter kunne fullføres. Etter omfattende testing ble det avdekket at feilen stammet fra hvordan bildene som ble tatt i siden for registrering av døde sauer blir lagt til i databasen. Bildene førte til at dataene fra oppsynsturen ble for store til å lastes opp i databasen (mer om dette i 20.1). For de neste brukertestene ble det besluttet å endre oppgave 5 slik at deltakerne slettet bildet som ble tatt før oppsynsturen ble fullført. Etter denne korrigeringen av oppgave 5 hadde ingen andre testdeltakere problemer med å utføre oppgave 7.

16.1.8 Oppgave 8: Sjekk liste av oppsynsturer

Alle deltakerne fant fram til oversikten over oppsynsturene. Derimot var det to deltakere som ikke kom inn på sin nylig registrerte oppsyntur ettersom de trykket på hele boksen for oppsynsturen og ikke pilen til høyre. Ingen av deltakerne fikk med seg at det var mulig å trykke på firkanten inne i boksen for hver registrering slik at kartet automatisk flyttet seg til den tilhørende pinnen. To av deltakerne kommenterte også på at det var lite mellomrom mellom tittelen på siden og kartet. Ellers var det en deltaker som kommenterte "For en fin oversikt".

16.1.9 Generelle tilbakemeldinger

Etter oppgavene var fullførte, spurte forsøkslederen testdeltakerne om hva slags inntrykk og generelle tilbakemeldinger de hadde om applikasjonen. Flere av testdeltakerne ga uttryk for at applikasjonen var enkel og lett å forstå. Spesielt ikonene for de ulike registreringene var forståelige uten tekstlige beskrivelser. Ikonene kunne derimot være større for å være enklere å trykke på. En av deltakerne synes det var vanskelig å se alternativene på enkelte av dialogboksene grunnet lav kontrast på bakgrunnsfargen og tekstfargen. Tekstskriften på knappene ble også kommentert på for å være for tynn og dermed vanskelig å lese. En annen deltaker delte at systemet burde lagre informasjonen i oppsynsturene regelmessig til fil for å ta vare på data dersom mobilen skulle gå tom for strøm. En av deltakerne som ikke var kjent med applikasjonen fra tidligere ønsket at det skulle være en pil eller forklaring som beskrev hvordan grensesnittet for registrering fungerte første gangen det skulle benyttes. Applikasjonens mørke tema ble roset av en testdeltaker, som også likte at all funksjonalitet er tilgjengelig fra hjemskjermen.

16.1.10 Resultatstabell

Table 16.1: Tabell som viser resultater fra de utførste brukertestene

Deltaker nr.	Tidligere deltaker	Oppgave	Problem	Tag 1	Følelse
1	Nei	3: Registrere sauer	Litt vanskeligheter i starten med å forstå at man skal sveipe sidelengs for å bytte underkategorier.	Registrering	Forvirret
1		5: Registrere død sau	GPS hopper et stykke bort.	GPS	Forvirret
2	Ja	1: Laste ned kartutsnitt.	Kartet lastes inn tregt, deltaker trykker på "Last ned kartutsnitt" før kartet er ferdig innlastet.	GPS	Forvirret, irritert
2		3: Registrere sauer.	Får litt hjelp til å zoome, flytter ikke markør på sted for eksakt lokasjon av registrering.	Markør	Forvirret
2		3.6: Registrere øremerker.	Går videre fra siden for registrering av øremerker til oppsummeringssiden før øremerket har blitt lagret. Trykker på "Neste"-knappen framfor å lukke tastaturet og trykke lagre.	Registrering	Forvirret
2		5.3: Registrere død sau.	Trykker på bildeikonet for å ta bilde og ikke "+"-knappen.	Bildetaking	
3	Ja	3: Registrere sauer.	Flytter ikke markøren for eksakt lokasjon for registrering.	Markør	

Fortsetter på neste side

Table 16.1 – fortsettelse fra forrige side

Deltake nr.	Tidligere deltaker	Oppgave	Problem	Tag 1	Følelse
3		5: Registrere død sau.	Klikker først på plaster-ikonet som er ikonet for skadet sau. Skjønner at det er feil og bytter selv til rett registrerings-side.	Ikon	Forvir-ret
3		8.2: Sjekke liste med oppsyns-turer.	Forsøker å klikke på hele boksen for å komme inn på en oppsynstur framfor å bruke pilen til høyre.	Oppsyns-turer	Forvir-ret, irritert
4	Ja	3: Registrere sauer.	Flytter ikke markøren til eksakt lokasjon for registrering.	Markør	
4		8.2: Sjekke liste med oppsyns-turer.	Forsøker å klikke på hele boksen for å komme inn på en oppsynstur framfor å bruke pilen til høyre.	Oppsyns-turer	Forvir-ret
5	Nei	2: Ny oppsyntur.	Legger ikke merke til at personen allerede er lagt inn som deltaker av oppsyn-sturen og legger til seg selv på nytt. Oppdager selv at det er registrert dobbelt og fjerner seg selv fra listen.	Ny opp-synstur	
5		3: Registrere sauer.	Flytter ikke markøren for eksakt lokasjon for registreringen.	Markør	

Fortsetter på neste side

Table 16.1 – fortsettelse fra forrige side

Deltake nr.	Tidligere deltaker	Oppgave	Problem	Tag 1	Følelse
5		3.6: Registrere øremerker.	Sjekker ikke av boksen for å velge øremerket som personen akkurat har laget. Oppdager heller ikke feilen i oppsummeringssiden.	Registrering	
5		5.3 Registrere død sau.	Trykker på bildeikonet for å ta bilde og ikke "pluss"-knappen.	Bildetaking	Forvirret

16.2 Oppsummering

Det ble utført fem kvalitative brukertester. Resultatene fra brukertestene er testdeltakernes prestasjoner under de tildelte oppgavene og deres tilbakemeldinger. Ved første brukertest ble det oppdaget at bildene som ble tatt under brukertesten førte til at oppsynsturen ikke kunne bli lastet opp på skyen, som gjorde at resten av brukertestene ble endret slik at bildene ikke ble lagt til. Andre mindre problemområder som ble avdekket var at deltakeren ikke forstod bruken av markøren for å plassere observasjoner og at øremerkene ikke alltid ble registrert. Applikasjonen fikk stort sett positive tilbakemeldinger som at den var enkel å forstå og bruke, men at det var et behov for litt større ikoner, luft mellom titlene og sterkere font for enda bedre lesbarhet og brukervennlighet.

Del V

Diskusjon

Denne delen diskuterer rundt valgene som ble tatt i løpet av prosjektet. Dette omhandler en evaluering av ulike sider ved prosessen og løsningen som ble utviklet. Måloppnåelse i forbindelse med kravspesifikasjonen vil også gjennomgås, etterfulgt av en analyse av resultatene fra brukertestene. Til slutt blir en evaluering av prosjektet presentert hvor forskningsspørsmålene F1.1 til F1.3 besvares.

Kapittel 17

Evaluering av prosessen

Dette kapittelet gjennomgår og evaluerer prosessene som ble brukt i forbindelse med design, utvikling og testing av applikasjonen under prosjektet.

17.1 Designfasen

Designet av *Sauron* ble utført i en egen designfase og ble gjort i sin helhet i design- og prototypeverktøyet Figma. Dette har ført til at applikasjonen har et helhetlig og konsist design. Likevel kan det ikke utelukkes at designet kunne blitt mer brukervennlig om det hadde blitt kjørt to iterasjoner av designprosessen. Den første iterasjonen ville da bli brukt til å utvikle et førsteutkast av et funksjonelt prototypdedesign i Figma. Denne prototypen ville blitt brukertestet for å luke ut de groveste designproblemene. I andre iterasjon ville det da blitt utviklet et andreutkast til designet i Figma med bakgrunn i tilbakemeldingene fra første iterasjon. Dette kunne gjort at selve designet hadde vært mer brukervennlig før *Sauron* ble utviklet i kode og når de funksjonelle brukertestene av applikasjonen ble gjort i slutten av oppgaven.

17.2 Utviklingsfasen

Selve utviklingen av applikasjonen startet etter at designet var ferdig utformet i Figma. Som forklart i kapt. 8 ble en versjon av utviklingsmetodikken *Kanban* brukt for å organisere og prioritere arbeidsoppgaver under utviklingsprosessen. Valget av en agil utviklingsmetodikk gjorde det mulig å tilpasse seg endringene som ble gjort i kravspesifikasjonen i løpet av prosjektet. Ettersom at utviklingsteamet bare besto av to personer og at applikasjonen som skulle utvikles var såpass enkel, ville nok ikke valget av en annen agil utviklingsmetodikk enn Kanban ført til betydelig forandring for resultatet av utviklingsfasen.

17.3 Gjennomføring av brukertestene

Hvordan brukertestene skulle gjennomføres og hvilke elementer av applikasjonen som skulle testes ble planlagt på forhånd. Som nevnt i kapt. 15.2.3 ble det utformet en plan for brukertestene, både for å være mest mulig forberedt og for å utvikle en brukertest som lignet reelle situasjoner der *Sauron* ville blitt brukt. Denne planen ble skrevet ut på papir og tatt med på brukertestene for å forsikre seg at all informasjon ble formidlet riktig til testdeltakerne og at alle oppgavene ble utført. Å benytte seg av

en slik plan gjorde også at brukertestene hadde et likt utgangspunkt for alle deltakerne. I tillegg gjennomførte utviklerne brukertesten selv flere ganger i henhold til planen for å undersøke om den var godt utformet og for å avdekke eventuelle kritiske problemområder i applikasjonen. Dette luket ut en del programvarefeil som ble fikset før brukertestene ble utført. Et par alvorlige programvarefeil ble også oppdaget under selve brukertestene (eksempler og mer utdypende informasjon i kapt. 20). Dette viser viktigheten av å jevnlig brukerteste applikasjoner, både fordi det avslører programvarefeil og problemområder under mer reelle situasjoner enn under utvikling og fordi testdeltakerne bruker applikasjonen på helt annen måte enn det utviklerne gjerne har sett for seg. Det kan også tenkes at det kunne vært lurt å ha brukervennlighetstester tidligere i designfasen som nevnt over i kapt. 17.1.

Det å gi utviklerne en bestemt rolle for alle brukertestene effektiviserte gjennomføringen av brukertestene. Rollene som forsøkleder og observatør ble avtalt på forhånd. Utviklerne beholdt samme rolle for alle brukertestene og ble dermed fort vante med sin rolle. Slik kunne forsøkslederen fokusere på å veilede deltakerne under testen og eventuelt hjelpe dem, mens observatøren noterte problemer som oppstod. Observatøren stod gjerne bak deltakerene slik at deltakernes interaksjoner med *Sauron* kunne observeres.

17.4 Oppsummering

Bruken av designverktøyet Figma for å utforme prototyper før utvikling gjorde at applikasjonen har fått et helhetlig og gjennomført design, men for å luke ut flere problemområder og sørge for enda bedre brukervennlighet hadde det vært ønskelig med to iterasjoner av designprosessen. Gjennomføringen av brukertestene ble gjort i henhold til den forhåndsbestemte planen som spesifiserte oppgavene og ruten. Planen sørget for at alle brukertestene hadde likt utgangspunkt og at utviklerne var forberedt til hver brukertest. Ved at utviklerne gjennomførte testen selv ble noen problemområder avdekket og fikset før de faktiske brukertestene, men selve brukertestene avslørte også større programvarefeil. Dette viser viktigheten av kontinuerlige brukertester, samt at det vil være et behov mer omfattende testing dersom applikasjonen skal utrulles.

Kapittel 18

Evaluering av løsningen

I denne delen diskuteres det rundt løsningen som ble utviklet og implementert, altså mobilapplikasjonen *Sauron*. Kapittelet går gjennom utforming av design, kodebasen og backend. Det diskuteres rundt valgene som ble tatt og hvilke fordeler og ulemper det er med disse sammenlignet med andre løsninger.

18.1 Utforming

Designet til applikasjonen ble laget i Figma før implementasjonen av *Sauron* startet. En evaluering av designet blir gjort på bakgrunn av resultatene fra brukertestene og gjennomgås i kapt. 20.

18.2 Programvarearkitektur

Som det kommer fram i kapt. 13.1 har koden blitt strukturert på en måte som vil gjøre det enkelt å utvide applikasjonen i framtiden. Å oppnå dette har vært et mål helt siden fordypningsprosjektet ettersom det var planlagt å utvide funksjonaliteten allerede under masteroppgaven når applikasjonen skulle fullføres. Ulempen med den valgte arkitekturen er at den er mer abstrakt og skaper økt kompleksitet, som kan gjøre det vanskelig for nye utviklere å sette seg inn i.

18.3 Bruk av Ionic og Capacitor

Ved bruk av Ionic med Capacitor har det blitt utviklet en kodebase som fungerer både på Android og iOS. Dette har minimert behovet for plattform-spesifikk kode. Løsningen gjør at det i framtiden vil være enklere å utvide eller fikse feil i applikasjonen ettersom enderinger bare må implementeres en gang for både iOS og Android. Enkelte deler av koden måtte likevel utvikles spesifikt for hver av plattformene. For at tekst-til-tale skulle fungere som ønsket på iOS under blind registrering av en saueflokk ble det nødvendig å utvikle et eget programtillegg for Capacitor til akkurat dette. Et slikt programtillegg utvikles i Java for Android og Swift for iOS og bindes sammen ved hjelp av Capacitor. Dette betyr at dersom det blir gjort endringer i rammeverkene for tekst-til-tale for enten Android eller Swift vil programtillegget måtte oppdateres.

Under utviklingen av den første versjonen av *Sauron* fantes det ingen programtillegg for registrering av GPS-lokasjon i bakgrunnen som fungerte for både iOS og Android. Dette førte til at når brukertestene ble utført var denne funksjonaliteten ikke

implementert. Programtillegget *capacitor-community/background-geolocation* [159] ble publisert i slutten av april 2021 og fungerer som ønsket både på iOS og Android. Tillegget har blitt implementert i applikasjonen men har ikke blitt brukertestet. Dette er en av ulempene ved å bruke Ionic og Capacitor. Begge teknologiene er relativt nye og funksjonalitet som kanskje har eksistert i andre rammeverk i lang tid kan potensielt være mangelfull i Ionic og Capacitor eller ikke eksistere i det hele tatt.

18.4 Backend

Ved å bruke en tjenester fra Firebase for både autentisering og lagring i database i form av Backend-as-a-Service ble det ikke nødvendig å utvikle en spesifikk tjenerløsning. Med funksjonaliteten som er utviklet per nå tilbyr Firebase de tjenestene som er nødvendige uten kompromiss. Hvis det i framtiden skulle være ønskelig med mer funksjonalitet i backend enn det Firebase muliggjør, hadde det potensielt vært et behov for å flytte deler av logikken over på en dedikert tjener. Mesteparten av logikken for autentisering og lagring kan beholdes hos Firebase da de tilbyr løsninger hvor en kan benytte egne tjenerløsninger sammen med autentisering og lagring hos Firebase.

18.5 Oppsummering

Programvarearkitekturen er strukturert for å gjøre *Sauron* enklere å utvide i fremtiden. En ulempe med dette er at det har økt abstraksjonsnivået og kompleksiteten for applikasjonen. Bruken av Ionic med Capacitor gjør at applikasjonen når kravet med å være en kryssplattform applikasjon. For å få dette til det ble nødvendig å utvikle et eget programvaretillegg til Capacitor for å få på plass all ønsket funksjonalitet. Å bruke Firebase som Backend-as-a-Service uten en spesifikk tjenerløsning dekker dagens behov for backend-funksjonalitet. Innloggingsløsningen og databasen kan integreres med en egen tjenerløsning i framtiden om det blir et behov for mer avansert logikk på tjenersiden.

Kapittel 19

Gjennomgang av kravspesifikasjonen

Dette kapittelet tar for seg kravspesifikasjonene satt i kapt- 7 og diskuterer hvorvidt disse kravene er nådd.

19.1 Gjennomgang av funksjonelle krav

De funksjonelle kravene for applikasjonen beskrives i tabell 7.1 i kapt. 7.1 og ble utviklet i samarbeid med professor Hvasshovd i starten og underveis i prosjektet. Gjennom prosjektet har nesten alle de funksjonelle kravene blitt oppnådd eller delvis oppnådd. Dette underkapittelet tar for seg de kravene som ikke ble fullstendig innfridd.

19.1.1 F3.1 (ikke oppnådd)

Det funksjonelle kravet F3.1 har beskrivelsen "Brukeren skal ha egen profil og mulighet til å se og endre på den". Kravet har prioriteten *Lav* og det ble derfor ikke prioritert under utviklingen av applikasjonen. Det har blitt opprettet en egen *Collection* for brukere i Firebase Cloud Firestore. Det eneste som gjenstår for å få på plass en redigerbar profil er å implementere et grensesnitt som muliggjør redigering.

19.2 Gjennomgang av ikke-funksjonelle krav

De ikke-funksjonelle kravene IF1, IF2 og IF4 fra tabell 7.2 er nådd. All funksjonalitet har blitt testet for både iOS og Android, både med og uten tilgang til internett. For IF4 har grensesnittet for blind bruk i denne applikasjonen blitt implementert på bakgrunn av resultatene i forbindelse med brukervennlighet og effektivitet funnet i brukertestene under fordypningsprosjektet [1, s.64].

Det ikke-funksjonelle kravet IF3 sier at "Applikasjonen skal tillate bruk på inntil 10 timer uten tilgang på strøm". I en enkel test kom det fram at under en oppsynstur på 33 minutter brukte applikasjonen 5% strøm på en iPhone 12 Pro. Testen tyder på at applikasjon vil kunne brukes på oppsynsturer som varer over 5 timer på de fleste mobiltelefoner. Det kan derimot ikke sies sikkert om applikasjonen vil kunne brukes på turer som varer inntil 10 timer.

Kapittel 20

Analyse av resultater fra brukertestene

Dette kapittelet tar for seg resultatene fra brukertestene og fokuserer på problemområdene som ble avdekket i applikasjonen. Her skilles det mellom et problem og en feil i applikasjonen. Med problemer menes funksjonalitet eller designvalg i applikasjonen som hindrer eller forvirrer i sånn måte at brukeren ikke klarer å bruke *Sauron* som tiltenkt. Feil vil regnes som programvarefeil i koden som gir diverse uheldige konsekvenser for applikasjonen. For å evaluere brukertestene tas det utgangspunkt i resultatene som ble presentert i tabell 16.1.10. Problemene i tabellen blir kategorisert i kritiske, alvorlige og mindre problemer som beskrevet av analyse- og markedsføringselskapet *Hotjar* [165]. I følge dem kategoriseres problemene som oppdages i brukertestene slik [166]:

- **Kritiske problemer:** Umulig for brukerne å fullføre oppgaver.
- **Alvorlige problemer:** Frustrerende for mange brukere.
- **Mindre problemer:** Irriterende problem, men ikke nok til å drive vekk brukere.

20.1 Programvarefeil

Det ble oppdaget en kritisk programvarefeil ved første brukertest der testdeltakeren ikke klarte å laste opp informasjonen som var hentet på oppsynsturen opp til databasen i Cloud Firestore. Ettersom brukertestene var kvalitative, ble det improvisert en falsk oppsynstur som deltakeren kunne laste opp slik at siste oppgave som var å sjekke den nylige opplastede oppsynsturen kunne fullføres. De resterende brukertestene ble satt på vent til feilen ble fikset. Denne feilen viste seg å være på grunn av at oppsynsturen ble for stor til å lagres i Firestore når bildet som ble tatt under oppgave 5 ble lagret som en base64-streng.

For å fikse problemet ble det forsøkt å skru ned bildekvaliteten slik at oppsynsturen kunne lastes opp i databasen, men resultatet ble fortsatt ikke stabilt nok for faktisk bruk. Det ble derfor bestemt å endre brukertestene slik at bildet fjernes før oppsynsturen lagres i databasen slik at brukertestene kunne gjennomføres som planlagt. I tillegg gjorde denne tilnærmingen at grensesnittet for å fjerne bilder ble testet, noe som ikke var en del av den opprinnelige brukertesten. En potensiell løsning på problemet legges fram i kapt. 23.

Et annet alvorlig problem som oppstod for én av deltakerne var at GPS-posisjonen ble unøyaktig og hoppet på kartet før den fant riktig posisjon. Dette førte både til forvirring for deltakeren og at GPS-ruten for oppsynsturen ble unøyaktig. Problemet skyldtes at GPS-signalet rekalibrerte seg dersom *Sauron* gikk i dvale eller ble lukket, som var tilfellet før bakgrunnsoppdatering av GPS-lokasjon ble implementert. Dermed ville den første koordinaten ofte være unøyaktig. For å få bukt på problemet under brukertestene ble det implementert en algoritme som sammenlignet hvor lang tid det gikk mellom hver registrering av et nytt koordinat og avstanden mellom koordinatene. Ved hjelp av algoritmen kunne applikasjonen ignorere koordinater som tilsa at personen hadde bevegd seg i hastigheter lang over rask gange. Algoritmen eliminerte problemer for resten av brukertestene. Rekalibreringen er det ikke behov for etter at bakgrunnsoppdatering av GPS-lokasjonen ble implementert. Denne funksjonaliteten har ikke blitt brukertestet innen ferdigstillingen av denne masteroppgaven.

20.2 Kritiske problemer

Det var ingen kritiske problemer knyttet til applikasjonen som ble avdekket under brukertestene.

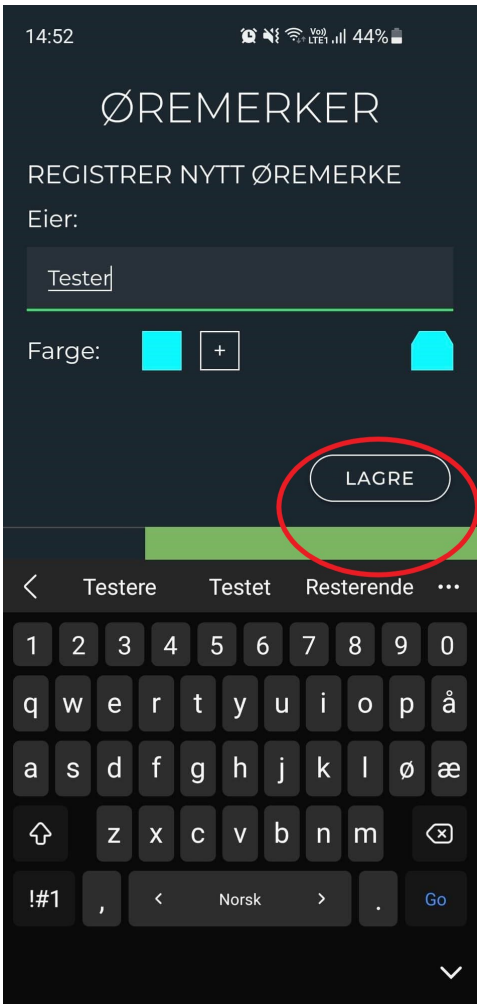
20.3 Alvorlige problemer

Det ble også avdekket alvorlige problemer som hindret applikasjonen å være like effektiv og behjelpelig for brukerne som tiltenkt. Et av problemene som gikk igjen flest ganger blant testdeltakerne var at de ikke forstod at markøren på kartet viste lokasjonen der registrerte observasjoner ville plasseres og at kartet derfor burde flyttes på før en ny registrering. Noen av deltakerne oppdaget dette selv etter å ha registrert en observasjon. Flere måtte få hjelp av forsøkslederen for å forstå hvordan dette foregikk. Selv om dette problemet ikke nødvendigvis frustrerte deltakerne ettersom de ikke var klar over problemet, kan det potensielt bli svært frustrerende i ettertid når observasjonene ikke blir plassert på faktisk lokasjon. Dette kan løses med å forklare bruken av markøren med en demonstrasjonvideo eller animasjon første gangen brukeren benytter seg av *Sauron*. I tillegg kan det legges til et vindu som kommer opp med en beskrivende tekst når markøren holdes på dersom brukerne blir usikre på hvordan markøren brukes. Slike vindu tilbys allerede i Leaflet-biblioteket som benyttes for kartet.

Et alvorlig problem som var forventet av utviklerne var at deltakerne som ikke var kjent med registreringsgrensesnittet ville slite med å forstå sveipingen i begynnelsen. Utviklerne ønsket å se hvor vanskelig det var å mestre grensesnittet uten hjelp. Som forventet slet begge deltakerene uten kjennskap markant mer enn deltakerne som hadde fått vist registreringsgrensesnittet tidligere. En av deltakerne måtte få hjelp for å kunne gå videre i brukertesten. Det har lenge vært en tanke å legge til en demonstrasjon av hvordan registreringsgrensesnittet brukes i blinde, enten med en video eller animasjon som vises første gang applikasjonen brukes. Brukertestene viser at dette er essensielt for at brukerene skal kunne utnytte registreringsgrensesnittet på best mulig vis slik at *Sauron* blir et hjelpsomt verktøy på oppsynstur.

En av testdeltakerne fikk problemer med å opprette og lagre et nytt øremerke i reg-

istreringsgrensesnittet. Når brukerne skriver navnet på personen som øremerket skal registreres på i tekstfeltet, presser mobiltastaturet navigasjonsknappene nederst på siden lengre opp slik at disse knappene kommer nært "Lagre"-knappen, se figur 20.1 under. Dette førte til at deltakeren trykket på "Neste"-knappen og applikasjonen gikk dermed videre i registreringen uten at det øremerket ble lagret. Deltakeren måtte gå tilbake til siden for øremerker og registrere øremerket på nytt. Problemet kan rettes på ved at tastaturet bare legges over navigasjonsknappene uten at de presses opp, slik at deltakeren aktivt må trykke vekk tastaturet for å bruke "Neste"-knappen.



Figur 20.1: Figur som viser hvordan tastaturet skyver opp navigasjonsknappene til "Lagre"-knappen i siden for registrering av øremerker

20.4 Mindre problemer

Et par mindre problemer ble oppdaget i brukertestene. Et av problemene som oppstod flest ganger blant testdeltakerne var at brukerne i siden for oppsynsturer trykket på hele boksen for å gå inn på en spesifikk oppsynstur og ikke pilen til høyre. Siden hver oppsynstur er fremhevet som en boks med en farget kant rundt, er det naturlig å tro at hele boksen skal trykkes på. Det er i tillegg enklere å trykke på boksen enn pilen med tanke på at boksen har større flate. Hele boksen bør implementeres til å

være trykkbar slik de fleste brukerne forventet.

En deltaker krysset ikke av for det øremerket som personen akkurat hadde opprettet og gikk dermed videre til oppsummeringssiden der øremerket ikke var registrert. Dette kan tyde på at brukerne antar at når det opprettes og lagres et nytt øremerke så blir det automatisk krysset av. En enkel fiks kan være å legge til automatisk avkryssing av nyopprettede øremerker.

To av testdeltakerne trykket på bildeikonet og ikke "Pluss"-knappen for å ta bilde i siden for registrering av død sau. For å unngå forvirring og at andre brukere støter på samme problem, kan bildeikonet enten gjøres mindre enn det er nå eller fjernes helt.

En av deltakerne tok feil av plaster- og dødninghodeikonet da personen skulle registrere en død sau. Personen oppdaget raskt at plasterikonet var for skadet sau og gikk videre til dødninghodet etterpå. Selv om dette var et enkelttilfelle blant deltakerene i brukertesten og resten forstod betydningen av ikonene på første forsøk, kan det være viktig å presisere ikonene slik at flere brukere ikke møter på dette problemet. En beskrivelse av ikonene kan legges inn i demonstrasjonen av applikasjonens funksjoner som nevnt tidligere, slik at brukerne får en innføring i hva ikonene representerer. Det å gjøre ikonene større kan også minske sannsynligheten for at brukerne trykker på feil ikon og vil gjøre det enklere å bruke *Sauron* med én hånd.

20.5 Oppsummering

Brukertestene avdekket flere programvarefeil og problemer i applikasjonen. En feil i implementasjonen for opplasting av oppsynsturer gjorde at testen måtte endres slik at bilder ikke ble lastet opp. Brukertestene avslørte også at det var et behov for en forklaring av mer avansert funksjonalitet som deriblant bruken av markøren på kartet og registreringsgrensesnittet. Disse alvorlige problemene gikk igjen flest ganger hos testdeltakerne.

Kapittel 21

Evaluering av prosjektet

Dette kapittelet tar for seg resultatene som ble funnet i prosjektet og om løsningen som ble utviklet tilfredsstiller forskningsspørsmålene F1.1 til F1.3 stilt i kapt. 1.3.

21.1 F1.1: Kan det utvikles et digitalt system som erstatter dagens løsning med penn og papir, men fortsatt dekker alle brukerens behov?

Applikasjonen *Sauron* har blitt utviklet med bakgrunn i dagens løsning, bøndenes behov, krav fra myndighetene og erfaringer fra professor Hvasshovd. Applikasjonen muliggjør registreringer av saueflokker, rovdyr, skadet sau og død sau. For en saueflokk kan det registreres informasjon om totalt antall sau, farge på sau, antall søyer og lam, farge på bjelleslips og øremerker gjennom et grensesnitt som muliggjør blind enhånds bruk. For å etterkomme framtidige behov har applikasjonen blitt utformet til å være enkel å utvide. Bondens lokasjon registreres automatisk under en oppsynstur og kobles sammen med registrerte observasjoner for å gi et oversiktlig bilde over den nedlagte turen. Funksjonalitet for nedlasting av kartutsnitt gjør det også mulig å bruke applikasjonen på steder hvor en ikke har internettilgang. Brukertestene viser at applikasjonen er relativt enkel å ta i bruk, men at det bør implementeres en form for introduksjon til mer avansert funksjonalitet. I teorien vil *Sauron* kunne erstatte dagens løsning med penn og papir, dekke brukernes behov og i tillegg tilby utvidet funksjonalitet for å lette på de tilsynsansvarliges arbeidsoppgaver. Det er ønskelig å få gjennomført en reell brukertest på en tilsynsansvarlig for å få bekreftet dette.

21.2 F1.2: Hvordan kan et digitalt system bistå sauebønder, beitelag og tilsynsansvarlige slik at de kan samhandle og dele informasjon om oppsynsturer med hverandre og norske myndigheter?

Et av målene med prosjektet har vært å lage et helhetlig system, som i motseting til dagens løsning, gjør det enkelt å dele informasjon om oppsynsturer innad i et beitelag og med myndighetene. Applikasjonen som er laget bistår med informasjonsdeling innad i beitelaget. Oppsynsturer som blir registrert er tilgjengelig for bønder innenfor samme beitelag. Dette er løst ved at oppsynsturene lagres i en database i skyen ved hjelp av Firebase. Logikk på tjenersiden sørger for at bøndene bare får tilgang til de oppsynsturene som er registrert av medlemmer innenfor samme beitelag.

Det har ikke blitt utviklet funksjonalitet for å dele ferdiglagde oppsynsturrapporter med myndighetene direkte fra applikasjonen. *Sauron* vil likevel kunne hjelpe med å lage rapporter da systemet vil, i motsetning til det eksisterende løsningen med penn og papir, samle all informasjon for utførte oppsynsturer fra et helt beitelag på et sted. Det vil potensielt gjøre det enklere å lage en mer fullstendig rapport, da man vil ha tilgang på mer informasjon gjennom applikasjonen enn det som var tilgjengelig tidligere.

21.3 F1.3: Hvordan utvikle et brukergrensesnitt som muliggjør registrering av sau uten å måtte se på mobilskjermen?

Under fordypningsprosjektet ble det utforsket flere ulike metoder for hvordan det kan utformes et brukergrensesnitt som gjør det mulig for en tilsynsansvarlig eller bonde å bruke en kikkert til å inspisere en saueflokk mens personen samtidig registrerer informasjonen på mobilen. Brukergrensesnittet som skulle lages måtte i praksis kunne brukes helt blindt. Grensensittet som ble utviklet bruker kombinasjon av haptisk tilbakemelding, tekst-til-tale og ulike interaksjonsmetoder som sveiping, vanlig trykking og lengre trykk for å oppnå dette. Med bakgrunn i undersøkelsene som ble utført under fordypningsprosjektet [1] vil grensesnittet som er laget teoretisk sett fungere til å registrere informasjon om saueflokker mens det samtidig blir brukt en kikkert. Likevel hadde det vært ønskelig å kunne utføre en ekte test på et utmarksbeite under beitesesongen med en person som har erfaring med oppsynsturer for å undersøke hvordan applikasjonen fungerer i en reell situasjon.

Del VI

Konklusjon og videre arbeid

Denne delen presenterer en konklusjon for prosjektet i sin helhet. Deretter legges det fram veien videre for *Sauron* med potensielt videre arbeid.

Kapittel 22

Konklusjon

Målet med dette prosjektet, som det fremlegges i forskningsspørmål **F1**, var å undersøke om en applikasjon som *Sauron* kunne bistå sauebønder, beitelag og tilsynsansvarlige på oppsynstur slik at arbeidet med manuell registrering blir raskere og mer effektivt.

Med bakgrunn i litteratursøket og samtaler med professor Hvasshovd har det blitt utviklet en applikasjon som kan erstatte dagens løsning med penn og papir. Applikasjonen gjør det mulig å registrere den informasjonen som skal være nødvendig for å lage rapporten myndighetene krever i forbindelse med tilsyn på utmarksbeite. Implementasjonen av et brukergrensesnitt som muliggjør bruk av applikasjonen med en hånd mens det samtidig tas i bruk kikkert, gjør også at det vil være enklere å registrere mer detaljert informasjon om sau på lang avstand. Brukertestene viser også at applikasjonen er relativt enkel å forstå seg på, men at det er et behov for en form for introduksjon til de mer avanserte delene av grensesnittet.

For å effektivisere innsamlingen av informasjon deles lagrede oppsynsturer med alle bønder innenfor samme beitelag. Dette kan potensielt føre til at det utføres mindre dobbeltarbeid i forbindelse med tilsyn på utmarksbeite. Likevel er funksjonaliteten som tilbys relativt enkel og vil trolig ikke være tilfredsstillende for bønder som ønsker en mer detaljert oversikt over utførte oppsynsturer. Funksjonaliteten har heller ikke blitt testet i en reell situasjon hos et beitelag. Det kan derfor ikke sies sikkert at den implementerte løsningen vil fungere som tiltenkt. For at bonden virkelig skal kunne dra nytte av informasjonsdelingen som tilbys av *Sauron* vil det være nødvendig å implementere webapplikasjonen som beskrives i kapt. 23.

Sammenlignet med dagens løsning kan applikasjonen i teorien forenkle tilsynsansvarliges arbeid, forenkle prosessen med å registrere observasjoner med bruk av kikkert og tilrettelegge for økt samhandling innad i beitelag. Det er likevel et behov for både videre utvikling og testing av applikasjonen.

Kapittel 23

Videre arbeid

I dette kapittelet gås det gjennom videre arbeid med applikasjonen både med tanke på funksjonalitet og testing.

23.1 Mer reelle brukertester

For å få et bedre innblikk i hvordan applikasjonen ville fungert i en reell situasjon ville det vært ønskelig å få gjennomført noen enkle brukertester med personer som driver med oppsynsturer til vanlig. Et ideelt scenario ville vært å være med en bonde eller tilsynsansvarlige på oppsynstur for å observere og gjøre tester på systemet.

Når applikasjonen etterhvert hadde blitt ferdigstilt slik det hadde vært mulig å rulle den ut i en testversjon på App Store for iOS og Google Play Store for Android, ville det vært mulig å kjøre tester innad i et beitelag. Dette ville gitt innsikt i hvordan funksjonaliteten som skal være med på å muliggjøre samhandling fungerer i praksis. Det ville også kunne være med på å anslå eventuelle kostander applikasjonen vil ha i forbindelse med bruk av Firebase.

23.2 Implementering av manglende og ønsket funksjonalitet

Denne delen går gjennom funksjonalitet som det enten ikke ble rukket å implementere i løpet av prosjektet eller som det ble ønsket endringer på av deltagerne fra brukertestene.

23.2.1 Demonstrasjonsvideo eller animasjon

Resultatene fra brukertestene avdekket at det er et behov for å ha en demonstrasjonsvideo eller animasjon for å forklare deler av grensesnittet i applikasjonen første gang den brukes. Dette gjelder særlig bruk av kartet og plasseringen av pins. Registreringsgrensesnittet for sauflokker trenger også en rask introduksjon. I tillegg til dette bør det også legges inn tekstlige beskrivelser av de forskjellige elementene på kartet som kan vises ved at brukeren holder inne på elementet. En gjennomgang av den mer avanserte funksjonaliteten kan legges inn på siden for innstillinger slik at brukerene kan lese seg opp på det om ønskelig. Dette vil gjøre det enkelt for brukerne å friske opp minnet mellom for eksempel beitesesonger.

23.2.2 Endring og redigering av observasjoner under oppsynstur

Etter brukertestene kom det fram at det potensielt kan være nødvendig med funksjonalitet som tillater at brukeren har mulighet til å slette eller endre en registrert observasjon under en oppsynstur. Hvis tilsynsansvarlig gjør en feilaktig registrering eller oppdager en annen feil er det ikke implementert en løsingen for å rette opp i dette. Redigering eller sletting av registrerte observasjoner er funksjonalitet som i teorien bør være enkel å få på plass.

23.2.3 Redigering og sletting av registrerte oppsynsturer

Per nå er det ingen implementert funksjonalitet for at brukere, hverken bønder eller tilsynsansvarlige, kan redigere eller slette lagrede oppsynsturer. Dette er funksjonalitet som det potensielt kan være ønskelig at superbrukere kan få ta nytte av hvis det har skjedd noe galt eller hvis en registrering viser seg å være feilaktig.

23.2.4 Brukerprofil med med mulighet for tilpasning av grensesnitt

Et av de funksjonelle kravene som ikke ble oppfylt var at brukerene skulle ha tilgang til en egen profil som kunne tilpasses etter eget ønske. Tanken her er å muliggjøre tilpasning av grensesnittet og funksjonalitet slik at det passer med beitelaget og brukeren. Et eksempel er å ha mulighet til å endre posisjonen for enkelte knapper slik at de er lettere å nå for personer som bruker mobilen med venstre hånd. Per nå er alle slike knapper designet for høyrehendte. Det kan også forekomme at farger på slips har forskjellige betydninger basert på hvilket beitelag man er medlem i. Mulighet for å endre dette burde være tilgjengelig gjennom en slik brukerprofil.

23.2.5 Rette opp feil som ikke tillater lagring av bilder i skyen

Under brukertestene kom det fram at det er en bug i applikasjonen som gjør at oppsynsturer som inneholder bilder ikke kan lastes opp i skyen. Denne feilen kan rettes opp på flere måter. Et av alternativene er å bruke Firebase sin Cloud Storage-funksjonalitet [167]. Cloud Storage gjør det mulig å lagre filer i skyen direkte fra applikasjonen. Selv ved dårlig nettverksforbindelse ved ned- og opplasting av filer vil applikasjonen få mulighet til å prøve å sende eller hente filene nytt idet nettverksforbindelsen returnerer [167]. Med tanke på at prosjektet allerede benytter seg av Firebase Authentication og Cloud Firestore, er dette det beste alternativet for opp- og nedlasting av bilder i prosjektet.

23.2.6 Integrering med radiobjeller

Ettersom stadig flere sauebønder benytter seg av radiobjeller for å spore opp dyreflokken ville det vært hensiktsmessig å integrere *Sauron* med aktørene for sporingstjenester som beskrevet i kapt. 4.1.1. Da vil lokasjonen til dyrene kunne vises direkte på kartet og det vil ikke være nødvendig å veksle mellom to forskjellige digitale system.

23.3 Implementering av webgrensesnitt for bønder

Det er også ønskelig å implementere en fullverdig webapplikasjon som primært skal brukes av bonden. Selv om det er mulig å se gjennom alle registrerte oppsynsturer

på dagens løsning er funksjonaliteten begrenset. Årsaken til dette er at det er en grense for hvor mye avansert funksjonalitet det er mulig å få plass til på skjermen til en mobil enhet. Tanken er at webapplikasjonen potensielt ville kunne kombinere resultatet fra flere turer og gjøre det mulig for brukerene å behandle informasjonen grundigere med mer avansert funksjonalitet. Et system for automatisk generering av årsrapporter vil være implementert i en slik løsning.

Referanser

[1] K. D. Abtahi and T. V. Gjelseth-Borgen, "Tilsyn med sau på beite."

[2] (04.23.21) Husdyrhald - Statistisk Sentralbyrå. [Online]. Available: https://www.ssb.no/jord-skog-jakt-og-fiskeri/jordbruk/statistikk/husdyrhald (Lastet ned: 2021-05-14).

[3] Beitebrukskart - Nibio. [Online]. Available: https://nibio.no/tema/landskap/kart-over-beitebruk-og-seterdrift/beitebrukskart (Lastet ned: 2021-05-14).

[4] I. Hansen and R. Rødven, *Tap Av Lam På Beite - Sammenheng Mellom Slippvekt Og Predasjon Av Jerv, Gaupe Og Rødrev*. NIBIO, Seksjon for utmarksressurser og naturbasert næringsutvikling, vol. 1. [Online]. Available: https://nibio.brage.unit.no/nibio-xmlui/bitstream/handle/11250/2380080/NIBIO%5C_POP%5C_2015%5C_1%5C_4.pdf?sequence=3&isAllowed=y (Lastet ned: 2021-01-19).

[5] Dyrebeskyttelsen Norge. Tap av sau på beite. [Online]. Available: https://www.dyrebeskyttelsen.no/tap-sau-pa-beite/ (Lastet ned: 2021-01-29).

[6] Mattilsynet, "Årsrapport - 2019," pp. 45–45. [Online]. Available: https://www.mattilsynet.no/om%5C_mattilsynet/aarsrapport%5C_2019%5C_%5C_mattilsynet.38708 (Lastet ned: 2021-01-21).

[7] "Forskrift om erstatning når husdyr blir drept eller skadet av rovvilt." [Online]. Available: https://lovdata.no/dokument/SF/forskrift/2014-05-30-677?q=erstatning%5Com%5Chusdyr

[8] Miljødirektoratet. Erstatning for sau drept av fredet rovvilt. [Online]. Available: https://soknadssenter.miljodirektoratet.no/ErstatningSauDreptAvFredetRovviltSkjema/Startside/Index?s%C3%B8knadstypeId=1 (Lastet ned: 2021-03-04).

[9] Miljødirektoratet. Tap til rovdyr holder seg på lavere nivå enn før. [Online]. Available: https://www.miljodirektoratet.no/aktuelt/nyheter/2020/januar-2020/tap-til-rovdyr-holder-seg-pa-lavere-niva-enn-for/ (Lastet ned: 2021-02-15).

[10] S. Bakke, "Tap av lam på sommerbeite som ikke skyldes rovdyr." [Online]. Available: https://nibio.brage.unit.no/nibio-xmlui/bitstream/handle/11250/2506617/Bioforsk-Rapport-2006-01-32.pdf?sequence=3&isAllowed=y (Lastet ned: 2021-01-27).

[11] J. R. E. Johanssen and K. M. Sørheim. Atferd og velferd hos sau. AgroPub. [Online]. Available: https://www.agropub.no/fagartikler/atferd-og-velferd-hos-sau (Lastet ned: 2021-03-24).

[12] B. Tjørhom, A. Odden, A. Domke, T. Tollersrud, and V. Tømmerberg. Forebygging av parasittproblemer med god beitebruk. Sauehelsenett, animalia. [Online]. Available: https://www.animalia.no/no/Dyr/sauehelsenett/arstid/beiteperioden/forebygging-av-parasittproblemer-med-god-beitebruk/ (Lastet ned: 2021-03-02).

[13] L. H. Engen and V. Tømmerberg, "Fôring av kopplam - slik lykkes du," no. 2, pp. 44–47.

[14] W. M. Velle, "Alveld." [Online]. Available: https://snl.no/alveld (Lastet ned: 2021-02-25).

[15] Sauehelsenett. Alveld. Animalia. [Online]. Available: https://www.animalia.no/no/Dyr/sauehelsenett/sjukdommer/forgiftninger/alveld/ (Lastet ned: 2021-02-25).

[16] Dyrevernalliansen. Sau og geit som husdyr. [Online]. Available: https://dyrevern.no/landbruksdyr/sau-og-geit/ (Lastet ned: 2021-02-22).

[17] Miljødirektoratet. Rovbase. [Online]. Available: https://www.miljodirektoratet.no/tjenester/nettsteder/rovbase/?_t_id=hgj1Sh4LjdRHFHksPJOQBQ%3d%3d&_t_uuid=sb%2fCMVKRQBOQIQ0KcqsKKQ&_t_q=rovbase&_t_tags=language%3ano%2csiteid%3a3fe17408-2bcc-45f8-86af-9820a42b8e53%2candquerymatch&_t_hit.id=Miljodirektoratet_Web_Features_Landing_Pages_LandingPage/_30a6c3ec-de8c-46c1-bfee-223e5b8b0b51_no&_t_hit.pos=1 (Lastet ned: 2021-02-27).

[18] Rovbase. Erstatning for sau. Rovbase. [Online]. Available: https://www.rovbase.no/erstatning/sau (Lastet ned: 2021-02-15).

[19] Klima- og miljødepartementet. Rekordlave tap av sau til rovvilt. [Online]. Available: https://www.regjeringen.no/no/aktuelt/rekordlave-tap-av-sau-til-rovvilt/id2724363/ (Lastet ned: 2021-02-25).

[20] Miljødirektoratet. Rekordlavt antall påviste rovviltskader på sau. [Online]. Available: https://www.miljodirektoratet.no/aktuelt/nyheter/2020/august-2020/rekordlavt-antall-paviste-rovviltskader-pa-sau/ (Lastet ned: 2021-02-25).

[21] Miljødirektoratet. Rovvilt tok færre sauer i 2020. [Online]. Available: https://www.miljodirektoratet.no/aktuelt/nyheter/2021/januar-2021/rovvilt-tok-farre-sauer-i-2020/ (Lastet ned: 2021-02-26).

[22] Statsforvalteren. Historia om statsforvalteren. Statsforvalteren. [Online]. Available: https://www.statsforvalteren.no/portal/Om-oss/historia-om-statsforvalteren/ (Lastet ned: 2021-03-02).

[23] Statsforvalteren. Styringsdokumenter. Statsforvalteren. [Online]. Available: https://styringsportalen.fylkesmannen.no/2021/styringsdokumenter/ (Lastet ned: 2021-03-05).

[24] Landbruksdirektoratet. Tilskudd til spesielle miljøtiltak i jordbruket (SMIL). Landbruksdirektoratet. [Online]. Available: https://www.landbruksdirektoratet.no/nb/jordbruk/ordninger-for-jordbruk/tilskudd-til-spesielle-miljotiltak-i-jordbruket-smil?openStep=9bd67683-82ff-41db-b03a-c841ef7b40e1-2 (Lastet ned: 2021-03-01).

[25] Mattilsynet. Om mattilsynet. Mattilsynet. [Online]. Available: https://www.mattilsynet.no/om_mattilsynet/ (Lastet ned: 2021-05-03).

[26] Mattilsynet. Sau og geit. Mattillsynet. [Online]. Available: https://www.mattilsynet.no/dyr_og_dyrehold/produksjonsdyr/sau_og_geit/ (Lastet ned: 2021-03-04).

[27] Mattilsynet and K. J. Baalsrud, "Tap av sau til fredet rovvilt."

[28] Mattilsynet, "Mattilsynets arbeid med dyrevelferd 1. og 2 tertial 2020." [Online]. Available: https://www.mattilsynet.no/dyr_og_dyrehold/dyrevelferd/rapporter_om_dyrevelferd/rapport_mattilsynets_arbeid_med_dyrevelferd_landdyr_1_og_2_tertial_2020.40525/binary/Rapport%20Mattilsynets%20arbeid%20med%20dyrevelferd%20landdyr%201.%20og%202.%20tertial%202020 (Lastet ned: 2021-02-19).

[29] M. Kjørstad. Statens naturoppsyn (SNO). Miljødirektoratet. [Online]. Available: https://www.miljodirektoratet.no/om-oss/miljodirektoratets-organisasjon/statens-naturoppsyn/ (Lastet ned: 2021-02-20).

[30] Statsforvalteren i Innlandet, "Informasjon til beitebrukere, rovviltkontakter og kommunene i Innlandet." [Online]. Available: www.miljovedtaksregisteret.nohttps://www.statsforvalteren.no/contentassets/3a7dff7c59194c3485db991a71526624/41305-informasjonsbrosjyre-rovvilt-og-beitesesong-2020-a5_web.pdf (Lastet ned: 2021-03-02).

[31] "Forskrift om merking, registrering og rapportering av småfe." [Online]. Available: https://lovdata.no/dokument/SF/forskrift/2005-11-30-1356

[32] Mattilsynet. Øremerking av småfe. Mattilsynet. [Online]. Available: https://www.mattilsynet.no/dyr_og_dyrehold/produksjonsdyr/merking_og_registrering_av_produksjonsdyr/oremerking_av_smaafe.4885 (Lastet ned: 2021-02-18).

[33] A. Bunger, M. Eide, H. Christian, and A. Smedshaug, "Småfenaeringen-største sektoren i norsk jordbruk," pp. 2–56. [Online]. Available: https://www.agrianalyse.no/getfile.php/133038-1525420084/Dokumenter/Dokumenter%202018/Rapport%205%20-%202018%20Sm%C3%A5fen%C3%A6ringen%20i%20Norge.pdf (Lastet ned: 2021-01-20).

[34] OS ID AS. Combi 3000 små Øremerke. OS ID AS. [Online]. Available: https://www.osid.no/produkter/sau/combi-3000-sma-oremerke/ (Lastet ned: 2021-03-01).

[35] Norsk Sau og Geit. Organisert beitebruk. [Online]. Available: https://www.nsg.no/beitebruk/utmarksbeite/organisert-beitebruk/ (Lastet ned: 2021-02-22).

[36] Norsk Sau og Geit. Bjelleslips. [Online]. Available: https://www.nsg.no/a-a/merking-av-smafe/bjelleslips/ (Lastet ned: 2021-03-05).

[37] Fylkesmannen i Oslo og Akershus, Statens Naturoppsyn, and Mattilsynet, "Beitedyr og rovdyr." [Online]. Available: https://www.statsforvalteren.no/contentassets/b61236a25d024880b6b15032e0e11ab2/rovviltbrosjyre_elektronisk-versjon--17-juli-2017.pdf (Lastet ned: 2021-03-03).

[38] Fostertelling og en sniktitt på høstens produksjonsreultater – Øvre Rønningen Gård. Øvre rønningen gård. [Online]. Available: https://gardsdraumen.wordpress.com/2015/02/03/fostertelling-og-en-sniktitt-pa-hostens-produksjonsreultater/ (Lastet ned: 2021-05-06).

[39] H. Øyrehagen. Beite og utmark. Norsk sau og geit. [Online]. Available: https://www.nsg.no/fylkeslaga/sogn-og-fjordane/lokallag/sogndal-sau-og-geit/utmarksbeite/ (Lastet ned: 2021-03-08).

[40] Norsk Sau og Geit. Riktig slippetidspunkt. Norsk sau og geit. [Online]. Available: https://www.nsg.no/dyrevelferd-pa-beite/riktig-slippetidspunkt/ (Lastet ned: 2021-03-05).

[41] Y. Rekdal, M. Angeloff, E. Skurdal, F. Avdem, V. Tømmerberg, and T. Skeidsvoll Trollersrud, "Temahefte - utmarksbeite til sau." [Online]. Available: https://www.geno.no/contentassets/68e6876772f147fe8bbecf28c473044f/utmarksbeite_sau_web.pdf (Lastet ned: 2021-02-24).

[42] O. K. Stornes and G. R. Rauset, "Tidlig nedsanking av sau og bare innmarksbeite-Sats per dyr og dag ved mer innmarksbeite." NIBIO. [Online]. Available: https://nibio.brage.unit.no/nibio-xmlui/handle/11250/2452541 (Lastet ned: 2021-03-04).

[43] J. Messel, "Bonde om forslag om tidlig nedsanking av sauer: – Helt feil medisin." [Online]. Available: https://www.nationen.no/article/bonde-om-forslag-om-tidlig-nedsanking-av-sauer-helt-feil-medisin/ (Lastet ned: 2021-03-08).

[44] Nortura Medlem. Utforming av sankegjerde - medlemsportal for nortura SA. Nortura medlem. [Online]. Available: https://medlem.nortura.no/arkiv-nyhetsartikler/utforming-av-sankegjerde-article36552-12003.html (Lastet ned: 2021-03-03).

[45] J. J. Fremstad. Sauebønder får gevinst av radiobjeller - ruralis. Ruralis. [Online]. Available: https://ruralis.no/2020/07/06/sauebonder-far-gevinst-av-radiobjeller/ (Lastet ned: 2021-03-09).

[46] Telespor. [Online]. Available: http://www.telespor.no/ (Lastet ned: 2021-03-12).

[47] Findmy | GPS sporing av husdyr på utmarksbeite - uten mobildekning. [Online]. Available: https://www.findmy.no/ (Lastet ned: 2021-05-23).

[48] Smartbjella sporing. [Online]. Available: https://smartbjella.no/ (Lastet ned: 2021-03-11).

[49] Produkt | Findmy | GPS sporing av husdyr på utmarksbeite. [Online]. Available: https://www.findmy.no/nb/produkt (Lastet ned: 2021-05-24).

[50] G.-T. Kvam, R. B. Hårstad, H. E. Almaas, and E. P. Stræte, "The role of advisory services in farmers' decision making for innovation uptake. Insights from case studies in Norway." [Online]. Available: https://www.agrilink2020.eu/wp-content/uploads/2020/04/D2.2_NO.GDPR_.pdf (Lastet ned: 2021-03-15).

[51] Produkt – Telespor AS. [Online]. Available: https://telespor.no/produkt/ (Lastet ned: 2021-03-12).

[52] Telespor-systemet. Telespor. [Online]. Available: http://telespor.no/informasjon/telespor-systemet/ (Lastet ned: 2021-03-12).

[53] M. Lorentzen. "Tingenes internett" inntar Norge: Men det er usikkert hvor fort det blir penger av det – E24. [Online]. Available: https://e24.no/teknologi/i/0np27o/tingenes-internett-inntar-norge-men-det-er-usikkert-hvor-fort-det-blir-penger-av-det (Lastet ned: 2021-03-12).

[54] Telenor. Dekning for IoT på 4G. [Online]. Available: https://www.telenor.no/bedrift/iot/dekning/ (Lastet ned: 2021-03-16).

[55] Telespor nettbutikk. [Online]. Available: https://nettbutikk.telespor.no/ (Lastet ned: 2021-03-16).

[56] Findmy | om oss. [Online]. Available: https://www.findmy.no/nb/om-oss (Lastet ned: 2021-03-15).

[57] FindMy | Satelitt. [Online]. Available: https://www.findmy.no/nb/satellitt (Lastet ned: 2021-03-15).

[58] FindMy | Dekning i hele Norge. [Online]. Available: https://www.findmy.no/nb/dekning-i-hele-norge (Lastet ned: 2021-03-15).

[59] FindMy | Slik fungerer satellittene. [Online]. Available: https://www.findmy.no/nb/gps (Lastet ned: 2021-03-15).

[60] FindMy | Funksjoner. [Online]. Available: https://www.findmy.no/nb/funksjoner (Lastet ned: 2021-03-15).

[61] FindMy | Shop. [Online]. Available: https://www.findmy.no/shop (Lastet ned: 2021-03-16).

[62] FindMy | Brukeravgift. [Online]. Available: https://www.findmy.no/nb/brukeravgift (Lastet ned: 2021-03-16).

[63] Home - StalkIT.no. [Online]. Available: https://stalkit.no/ (Lastet ned: 2021-03-18).

[64] SMARTBELLS AS emisjonskampanje | Folkeinvest. [Online]. Available: https://folkeinvest.no/kampanje/smartbells-as (Lastet ned: 2021-03-18).

[65] T. K. Henriksen. Konkurransen i beite-tech hardner til: "Alle" vil passe på dyrene dine - Shifter. [Online]. Available: https://shifter.no/nyheter/konkurransen-i-beite-tech-hardner-til-alle-vil-passe-pa-dyrene-dine/165196 (Lastet ned: 2021-03-18).

[66] PRODUKT - Smartbjella Sporing. [Online]. Available: https://smartbjella.no/produkt/ (Lastet ned: 2021-03-18).

[67] Smartbjella 2 (pre-salg 2021) - Smartbjella Sporing. [Online]. Available: https://smartbjella.no/product/smartbjella-2-kjop/ (Lastet ned: 2021-03-18).

[68] Landbruksdirektoratet. Tilskudd til tiltak i beiteområder. [Online]. Available: https://www.landbruksdirektoratet.no/nb/jordbruk/ordninger-for-jordbruk/tilskudd-til-tiltak-i-beiteomrader?openStep=7178ef2e-7275-43fb-ba48-91e1548a6d21-0 (Lastet ned: 2021-03-16).

[69] S. landbruksforvaltning, "Nasjonalt Beiteprosjekt 2009 - 2012." [Online]. Available: http://www.kore.no/wp-content/uploads/2015/02/sluttrapport-nasjonalt-beiteprosjekt.pdf (Lastet ned: 2021-03-15).

[70] Beitesnap - Revolusjonerende verktøy for husdyr på beite! [Online]. Available: https://www.beitesnap.no/ (Lastet ned: 2021-03-17).

[71] S. Dysthe and A. Kjerstad, "Effektivisering av manuell oppfølging av sau på utmarksbeite."

[72] BeiteSnap | Facebook. [Online]. Available: https://www.facebook.com/beitesnap/?ref=page_internal (Lastet ned: 2021-03-17).

[73] Norgeskart | Kartverket.no. [Online]. Available: https://www.kartverket.no/til-lands/kart/norgeskart (Lastet ned: 2021-03-17).

[74] Beitesnap, "Komplett brukermanual for beitebrukere." [Online]. Available: https://www.beitesnap.no/filer/ (Lastet ned: 2021-03-17).

[75] G. Rachiele. Software Development Methodologies. Medium. [Online]. Available: https://medium.com/@gianpaul.r/software-development-methodologies-a856883a7630 (Lastet ned: 2021-04-08).

[76] Waterfall Model: What Is It and When Should You Use It? Airbrake. [Online]. Available: https://airbrake.io/blog/sdlc/waterfall-model (Lastet ned: 2021-04-09).

[77] W. W. Royce, "Managing the Development of Large Software Systems: Concepts and Techniques," in *Proceedings of IEEE WESCON*. The Institute of Electrical and Electronics Engineers, pp. 328–338.

[78] Waterfall Model (Software Engineering) - javatpoint. javaTpoint. [Online]. Available: https://www.javatpoint.com/software-engineering-waterfall-model (Lastet ned: 2021-04-09).

[79] SDLC - Waterfall Model - Tutorialspoint. [Online]. Available: https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm (Lastet ned: 2021-04-09).

[80] J. A. Livermore, "Factors that impact implementing an agile software development methodology," in *Conference Proceedings - IEEE SOUTHEASTCON*, pp. 82–86.

[81] P. Rannikko, "User-Centered Design in Agile Software Development." [Online]. Available: https://trepo.tuni.fi/bitstream/handle/10024/82310/gradu04854.pdf?sequence=1&isAllowed=y (Lastet ned: 2021-04-09).

[82] C. Larman, *Agile and Iterative Development: A Manager's Guide*. Addison-Wesley Professional.

[83] K. Beck, M. Beedle, A. van Bennekum, and A. Cockburn. Manifesto for Agile Software Development. Agile Manifesto. [Online]. Available: https://agilemanifesto.org/ (Lastet ned: 2021-04-09).

[84] K. Beck and M. Beedle. Principles behind the Agile Manifesto. Agile Manifesto. [Online]. Available: https://agilemanifesto.org/principles.html (Lastet ned: 2021-04-12).

[85] Why Agile is important for Software Development - Digital Template Market. [Online]. Available: https://digitaltemplatemarket.com/agile-important-software-development/ (Lastet ned: 2021-04-12).

[86] Home | Scrum.org. [Online]. Available: https://www.scrum.org/ (Lastet ned: 2021-04-12).

[87] D. Wells. Extreme Programming: A Gentle Introduction. [Online]. Available: http://www.extremeprogramming.org/ (Lastet ned: 2021-04-12).

[88] K. Schwaber and M. Beedle, *Agile Software Development with Scrum*. Prentice Hall, vol. 1.

[89] What is Scrum? Scrum.org. [Online]. Available: https://www.scrum.org/resources/what-is-scrum (Lastet ned: 2021-04-12).

[90] S. M. Nes. En kort introduksjon til Scrum | Visma Blog. Visma. [Online]. Available: https://www.visma.no/blogg/en-kort-introduksjon-til-scrum/ (Lastet ned: 2021-04-12).

[91] D. Radigan. Kanban - A brief introduction | Atlassian. Atlassian. [Online]. Available: https://www.atlassian.com/agile/kanban (Lastet ned: 2021-04-12).

[92] Trello. [Online]. Available: https://trello.com/en (Lastet ned: 2021-04-12).

[93] B. J. Oates, *Researching Information Systems and Computing*. SAGE Publications Ltd.

[94] Flutter - Beautiful native apps in record time. [Online]. Available: https://flutter.dev/?gclsrc=aw.ds&&gclid=Cj0KCQjwi7yCBhDJARIsAMWFScMsWOhzUdKWUbmC45d_uzQFzIFloQsROKtuYb2nhLAyP6SZuof4GkYaAjG5EALw_wcB (Lastet ned: 2021-03-15).

[95] Native mobile apps with Angular, Vue.js, TypeScript, JavaScript - NativeScript. [Online]. Available: https://nativescript.org/ (Lastet ned: 2021-03-15).

[96] React Native · A framework for building native apps using React. [Online]. Available: https://reactnative.dev/ (Lastet ned: 2021-03-15).

[97] React – A JavaScript library for building user interfaces. [Online]. Available: https://reactjs.org/ (Lastet ned: 2021-03-15).

[98] Ionic - Cross-Platform Mobile App Development. [Online]. Available: https://ionicframework.com/ (Lastet ned: 2021-03-15).

[99] M. Schwarzmüller. React Native vs Flutter vs Ionic vs NativeScript vs PWA. [Online]. Available: https://academind.com/tutorials/react-native-vs-flutter-vs-ionic-vs-nativescript-vs-pwa/ (Lastet ned: 2021-03-15).

[100] Om Geonorge. [Online]. Available: https://www.geonorge.no/aktuelt/om-geonorge/ (Lastet ned: 2021-03-15).

[101] Leaflet - a JavaScript library for interactive maps. [Online]. Available: https://leafletjs.com/ (Lastet ned: 2021-03-15).

[102] Brukerveiledning. [Online]. Available: https://www.geonorge.no/aktuelt/om-geonorge/brukerveiledning/ (Lastet ned: 2021-03-15).

[103] Dart overview | Dart. [Online]. Available: https://dart.dev/overview#platform (Lastet ned: 2021-03-15).

[104] Introduction to the DOM - Web APIs | MDN. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction (Lastet ned: 2021-03-15).

[105] Documentation - Leaflet - a JavaScript library for interactive maps. [Online]. Available: https://leafletjs.com/reference-1.7.1.html#domutil (Lastet ned: 2021-03-15).

[106] Using Libraries · React Native. [Online]. Available: https://reactnative.dev/docs/libraries (Lastet ned: 2021-03-15).

[107] Ionic Angular Overview - Ionic Documentation. [Online]. Available: https://ionicframework.com/docs/angular/overview (Lastet ned: 2021-03-15).

[108] Getting Started - NativeScript Docs. [Online]. Available: https://docs.nativescript.org/angular/start/introduction (Lastet ned: 2021-03-15).

[109] Angular - ElementRef. [Online]. Available: https://angular.io/api/core/ElementRef (Lastet ned: 2021-03-15).

[110] Angular - Using published libraries. [Online]. Available: https://angular.io/guide/using-libraries (Lastet ned: 2021-03-15).

[111] Introducing Ionic 4: Ionic for Everyone - Ionic Blog. [Online]. Available: https://blog.ionicframework.com/introducing-ionic-4-ionic-for-everyone/ (Lastet ned: 2021-03-17).

[112] Capacitor Blog - How Capacitor Works. [Online]. Available: https://capacitorjs.com/blog/how-capacitor-works (Lastet ned: 2021-03-17).

[113] Capacitor: Cross-platform native runtime for web apps. [Online]. Available: https://capacitorjs.com/ (Lastet ned: 2021-03-17).

[114] Apache Cordova. [Online]. Available: https://cordova.apache.org/ (Lastet ned: 2021-03-17).

[115] Cordova vs Capacitor - Ionic - The Cross-Platform App Development Leader. [Online]. Available: https://ionicframework.com/resources/articles/capacitor-vs-cordova-modern-hybrid-app-development?_gl=1*1q8dn6q*_ga*Mzc1ODA5MjguMTU5NzkyMDE5Ng..*_ga_REH9TJF6KF*MTYxNTk4MjcyNy4yNS4xLjE2MTU5ODQ3MTQuMA.. (Lastet ned: 2021-03-17).

[116] Angular. [Online]. Available: https://angular.io/ (Lastet ned: 2021-03-17).

[117] "Vuejs/vue," vuejs. [Online]. Available: https://github.com/vuejs/vue (Lastet ned: 2021-03-17).

[118] T. Mikkonen and A. Taivalsaari, "Web Applications – Spaghetti Code for the 21st Century," in *2008 Sixth International Conference on Software Engineering Research, Management and Applications*, pp. 319–328.

[119] Does your web app need a front-end framework? - Stack Overflow Blog. [Online]. Available: https://stackoverflow.blog/2020/02/03/is-it-time-for-a-front-end-framework/ (Lastet ned: 2021-03-17).

[120] Announcing the (new) Ionic Vue Beta - Ionic Blog. [Online]. Available: https://ionicframework.com/blog/announcing-the-new-ionic-vue-beta/ (Lastet ned: 2021-03-17).

[121] TypeScript: Typed JavaScript at Any Scale. [Online]. Available: https://www.typescriptlang.org/ (Lastet ned: 2021-03-17).

[122] Angular - What is Angular? [Online]. Available: https://angular.io/guide/what-is-angular (Lastet ned: 2021-03-17).

[123] Introduction - NGXS. [Online]. Available: https://www.ngxs.io/ (Lastet ned: 2021-03-17).

[124] Redux - A predictable state container for JavaScript apps. | Redux. [Online]. Available: https://redux.js.org/ (Lastet ned: 2021-05-20).

[125] Introduction - NGXS. [Online]. Available: https://www.ngxs.io/concepts/intro (Lastet ned: 2021-03-17).

[126] TextToSpeech | Android Developers. [Online]. Available: https://developer.android.com/reference/android/speech/tts/TextToSpeech (Lastet ned: 2021-03-17).

[127] AVSpeechSynthesizer | Apple Developer Documentation. [Online]. Available: https://developer.apple.com/documentation/avfaudio/avspeechsynthesizer (Lastet ned: 2021-03-17).

[128] "Capacitor-community/text-to-speech," Capacitor Community. [Online]. Available: https://github.com/capacitor-community/text-to-speech (Lastet ned: 2021-03-17).

[129] Creating Capacitor Plugins - Capacitor. [Online]. Available: https://capacitorjs.com/docs/plugins/creating-plugins (Lastet ned: 2021-03-17).

[130] About npm | npm Docs. [Online]. Available: https://docs.npmjs.com/about-npm (Lastet ned: 2021-03-17).

[131] Firebase. [Online]. Available: https://firebase.google.com/ (Lastet ned: 2021-03-17).

[132] About | Node.js. [Online]. Available: https://nodejs.org/en/about/ (Lastet ned: 2021-03-17).

[133] Managed MongoDB Hosting | Database-as-a-Service | MongoDB. [Online]. Available: https://www.mongodb.com/cloud/atlas (Lastet ned: 2021-05-27).

[134] Firebase - Crunchbase Company Profile & Funding. [Online]. Available: https://www.crunchbase.com/organization/firebase (Lastet ned: 2021-03-17).

[135] Google Acquires Firebase To Help Developers Build Better Real-Time Apps. TechCrunch. [Online]. Available: https://social.techcrunch.com/2014/10/21/google-acquires-firebase-to-help-developers-build-better-realtime-apps/ (Lastet ned: 2021-03-17).

[136] What is BaaS? | Backend-as-a-Service vs. serverless | Cloudflare. [Online]. Available: https://www.cloudflare.com/learning/serverless/glossary/backend-as-a-service-baas/ (Lastet ned: 2021-03-17).

[137] Firebase Products. [Online]. Available: https://firebase.google.com/products-build (Lastet ned: 2021-03-17).

[138] Cloud Firestore | Firebase. [Online]. Available: https://firebase.google.com/docs/firestore (Lastet ned: 2021-03-17).

[139] Firebase Authentication | Simple, free multi-platform sign-in. [Online]. Available: https://firebase.google.com/products/auth (Lastet ned: 2021-03-17).

[140] F. Bianchi. Coolors - The super fast color schemes generator! [Online]. Available: https://coolors.co/ (Lastet ned: 2021-03-11).

[141] Colors HEX. [Online]. Available: https://www.w3schools.com/colors/colors_hexadecimal.asp (Lastet ned: 2021-03-11).

[142] Colors RGB. [Online]. Available: https://www.w3schools.com/colors/colors_rgb.asp (Lastet ned: 2021-03-11).

[143] Affinity designer – professional graphic design software. [Online]. Available: https://affinity.serif.com/en-us/designer/ (Lastet ned: 2021-03-11).

[144] Bransjeledende vektorgrafikkprogramvare | adobe illustrator. [Online]. Available: https://www.adobe.com/no/products/illustrator.html?promoid=PGRQQLFS&mv=other (Lastet ned: 2021-03-11).

[145] Figma: The collaborative interface design tool. [Online]. Available: https://www.figma.com/ (Lastet ned: 2021-03-11).

[146] Adobe XD | fast & powerful UI/UX design & collaboration tool. [Online]. Available: https://www.adobe.com/no/products/xd.html (Lastet ned: 2021-11-03).

[147] OLED introduction and basic OLED information | OLED-Info. [Online]. Available: https://www.oled-info.com/oled-introduction (Lastet ned: 2021-03-11).

[148] What is the impact of Dark Mode on battery drain? – Mobile Enerlytics. [Online]. Available: http://mobileenerlytics.com/dark-mode/ (Lastet ned: 2021-03-11).

[149] L. Cardoso. LeonardoCardoso/Colorblinding: An extension for Google Chrome (and Chromium) that simulates the website as a color vision impaired person would see. [Online]. Available: https://github.com/LeonardoCardoso/Colorblinding (Lastet ned: 2021-03-11).

[150] Øremerker. [Online]. Available: http://lindholtdata.no/lvs/Merker.aspx (Lastet ned: 2021-03-08).

[151] Gol beitelag SA. [Online]. Available: http://golbeitelag.no/ (Lastet ned: 2021-03-01).

[152] Nannestad sau og geit. [Online]. Available: www.nsg.no/akershus/nannestad (Lastet ned: 2021-01-28).

[153] R. Jenni, "RaphaelJenni/FirebaseUI-Angular." [Online]. Available: https://github.com/RaphaelJenni/FirebaseUI-Angular (Lastet ned: 2021-03-22).

[154] Norgeskart bakgrunn cache - Kartkatalogen. [Online]. Available: https://kartkatalog.geonorge.no/metadata/norgeskart-bakgrunn-cache/c0d063aa-59fc-42db-bc5d-a1c88f2bf256 (Lastet ned: 2021-03-23).

[155] Zoom levels - Leaflet - a JavaScript library for interactive maps. [Online]. Available: https://leafletjs.com/examples/zoom-levels/ (Lastet ned: 2021-03-23).

[156] Map Tile API. [Online]. Available: https://xserver2.cloud.ptvgroup.com/dashboard/Content/TechnicalConcepts/Rendering/DSC_Map_Tile_API.htm (Lastet ned: 2021-05-29).

[157] Slippy map tilenames – OpenStreetMap Wiki. [Online]. Available: https://wiki.openstreetmap.org/wiki/Slippy_map_tilenames (Lastet ned: 2021-03-22).

[158] M. Dong, Y.-S. K. Choi, and L. Zhong, "Power-saving color transformation of mobile graphical user interfaces on OLED-based displays," in *Proceedings of the 14th ACM/IEEE International Symposium on Low Power Electronics and Design - ISLPED '09*. ACM Press, p. 339. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1594233.1594317 (Lastet ned: 2021-03-23).

[159] @capacitor-community/background-geolocation - npm. [Online]. Available: https://www.npmjs.com/package/@capacitor-community/background-geolocation (Lastet ned: 2021-05-20).

[160] S. Krug, *Rocket Surgery Made Easy: The Do-It Yourself Guide to Finding and Fixing Usability Problems*. New Riders.

[161] R. Budiu. Quantitative vs. Qualitative Usability Testing - Nielsen Norman Group. [Online]. Available: https://www.nngroup.com/articles/quant-vs-qual/ (Lastet ned: 2021-03-22).

[162] Hva er en Think Aloud-protokoll? - Netinbag. [Online]. Available: https://www.netinbag.com/no/internet/what-is-a-think-aloud-protocol.html (Lastet ned: 2021-03-23).

[163] D. Renwick. How many participants do I need for qualitative research? Optimal Workshop. [Online]. Available: https://blog.optimalworkshop.com/how-many-participants-do-i-need-for-qualitative-research/ (Lastet ned: 2021-04-15).

[164] Avstand og færre kontakter - FHI. [Online]. Available: https://www.fhi.no/nettpub/coronavirus/fakta/avstand-kontakter/?term=&h=1 (Lastet ned: 2021-03-24).

[165] Hotjar: Website Heatmaps & Behavior Analytics Tools. [Online]. Available: https://www.hotjar.com/ (Lastet ned: 2021-04-22).

[166] How to Analyze and Evaluate Usability Tests | Hotjar. hotjar. [Online]. Available: https://www.hotjar.com/usability-testing/evaluation-analysis/ (Lastet ned: 2021-04-19).

[167] Cloud Storage for Firebase. Firebase. [Online]. Available: https://firebase.google.com/docs/storage (Lastet ned: 2021-04-22).

Vedlegg

Vedlegg A

Bruksanvisning for å kjøre applikasjonen lokalt

A.1 Sette opp miljø på maskinen

For å få satt opp miljøet på maskinen hvor applikasjonen skal kjøre følger du denne guiden: `https://capacitorjs.com/docs/v3/getting-started/environment-setup`.

A.2 Klone kildekoden fra Github

Kildekoden til prosjektet kan hentes ned til egen maskin fra Github. Linken til repoet er: `https://github.com/tvgb/sau-client/tree/runnable-build`. Pass på at den valgte branchen er "runnable-build".

For å hente koden til egen maskin kan du enten bruke *git* eller laste ned filene i en zip-mappe. For å installere git på egen maskin følg denne guiden: `https://git-scm.com/book/en/v2/Getting-Started-Installing-Git`.

Hvis du velger å klone koden ved hjelp av git, naviger først inn i mappen hvor du ønsker at prosjektet skal ligge. Åpne et kommandolinjeverktøy som *Terminal* på MacOS eller powershell på windows i mappen. Skriv deretter denne kommandoen etterfulgt av *enter*:

```
git clone https://github.com/tvgb/sau-client.git
```

Naviger deretter inn i mappen ved å skrive denne kommandoen etterfulgt av *enter*:

```
cd sau-client
```

Hvis du velger å laste ned koden som en zip-fil kan du gjør det ved å gå inn på `https://github.com/tvgb/sau-client/tree/runnable-build`, trykke på den grønne "Code"-knappen og velge "Download ZIP". Når mappen er lastet ned pakker du den ut i en mappe du kaller "sau-client". Naviger deretter inn i mappen på samme måte som nevnt over.

A.3 Kjøre koden

Med en terminal åpen i mappen "sau-client" og med Node.js med NPM installert, skriv disse kommandoene etterfulgt av *enter*:

```
npm install
```

```
npm build
```

For å starte opp applikasjonen på android kjør deretter denne kommandoen etterfulgt av enter:

```
ionic capacitor run android
```

Når Android Studio har åpnet seg og og gradle build er ferdig kan du starte applikasjonen ved å trykke på play-knappen opp i høyre hjørne.

Hvis du er på mac og har installert xcode kan koden kjøres på iOS ved å kjøre denne kommandoen etterfulgt av enter:

```
ionic capacitor run ios
```

Når Xcode har åpnet seg og lastet kan du trykke på playknappen oppe i venstre hjørne for å kjøre applikasjonen.

A.4 Logge inn i applikasjonen

For å logge inn i applikasjonen bruker du brukernavnet `testbruker@gmail.com` med passord *123456*.

Vedlegg B

Rapportskjerma

Organisert beitebruk

RAPPORTSKJEMA – TILSYN PÅ UTMARKSBEITE

Dette skjemaet kan nyttast for å sikre beitelaget ein dokumentasjon på utført tilsyn med dyra i utmarka gjennom beitesesongen, slik det er sett krav om i *Forskrift om tilskott til organisert beitebruk.* Det er lagt opp til at kvar tur skal noterast fortløpande på dette skjemaet. Av aktuelle opplysningar kan nemnast daude og skadde dyr, årsak, rovvilt, laushundar, generell uro, søyer som manglar lam, beiteforhold og anna. Kartfesting av observasjonar som tillegg til dette skjemaet kan vera aktuelt. Etter endt beitesesong skal oppsummeringsskjemaet på neste side (baksida) fyllast ut og sendast leiaren i laget.

Beitelag: ……. Beiteår: ……

Tilsynsperson: ……..

Dato:	Rute/område:
Observasjonar:	
Dato:	Rute/område:
Observasjonar:	
Dato:	Rute/område:
Observasjonar:	
Dato:	Rute/område:
Observasjonar:	

Dato:	Rute/område:
Observasjonar:	
Dato:	Rute/område:
Observasjonar:	
Dato:	Rute/område:
Observasjonar:	
Dato:	Rute/område:
Observasjonar:	

Oppsummering av beitesesongen - informasjon om dyretal, tap og eigeninnsats, inkludert km køyring

	Sau	Lam	Sau + lam	Storfe	Geit/kje
Dyr sleppt på utmarksbeite					
Dyr tapt på utmarksbeite					
Tap i % (dyr tapt/dyr sleptx100)					
Eigeninnsats, dagar:	Tilsyn	Sanking	Anna arbeid	Samla	Køyring, km

Vedlegg C

Utfylte rapportskjema fra brukertestene

C.1 Brukertest 1

Selve brukertesten

Brukeren hadde ikke brukt registreringsgrensesnittet før testen.

Oppgave 1: Laste ned kartutsnitt

- **1.1** Endre navnet

Brukeren utfører oppgaven uten stopp. Alt går kjempesmud.

Oppgave 2: Ny oppsynstur

- **2.1** Legge til deltakere, alle som er med på turen, kan legge til beskrivelse hvis de har lyst.

Skjønner hvordan navn skal legges til uten spørsmål.

Oppgave 3: Registrere sauer

- Gå til O.S Bragstads plass 2b.
    - **3.1** 5 sauer totalt
    - **3.2** 3 hvite, 1 svart og 1 brun
    - **3.3** 2 søyer og 3 lam
    - **3.4** 1 grønt slips og 1 gul
    - **3.6** Register et nytt øremerke
    - **3.7** Fullføre oppsynstur

Brukeren skjønner hvordan man plasserer kartet for å registrere sau. Skjønner også at man trykker på pluss og på sauesymbolet.

Brukeren skjønner hvordan man registrerer den forskjellige informasjonen. Skjønner også hvordan tilbakeknappen fungerer.

Skjønner hvordan man legger til et nytt øremerke og velger det etterpå.

Litt vanskelig å skjønne at man kunne sveipe mellom ulike underkategorier under en registrering. Fant relativt fort ut av det på egenhånd og det ble lett etter det.

Registreringen går utmerket.

Oppgave 4: Registrere død sau

- Gå til hovedbygningen
    - **4.1** Registrer 1 dødt dyr
    - **4.2** Skriv kommentar om ønskelig
    - **4.3** Ta et bilde

GPS hopper et stykke bort. Forvirrer brukeren litt.

Skjønner hvordan man skal registrere et dødt dyr. Skjønner også hvordan man tar bilde. Registreringen går utmerket.

Oppgave 5: Registrere skadet sau

- Gå til 7030 Trondheim
    - **5.1** Registrer 2 skadede sau
    - **5.2** Skriv kommentar om ønskelig

Fant ut at man skulle trykke på bandasjen. Registreringen gikk utmerket. Brukeren sier at «Dette var litt gøy.»

Oppgave 6: Registrere rovdyr

- Gå til Kolbjørn Hejes vei 1A
    - **6.1** Registrer observasjon av gaupe
    - **6.2** Skriv kommentar om ønskelig

Brukeren trykker på rett symbol og velger gaupe. Registrering går utmerket.

Oppgave 7: Fullføre oppsynstur

- **7.1** Sjekk om registreringene stemmer
- **7.2** Skriv beskrivelse om ønskelig
- **7.3** Trykk Fullfør.

Noe gikk galt under opplastingen og vi må lage en fake oppsynstur

Oppgave 8: Sjekk liste av oppsynsturer

- **8.1** Trykk på “Oppsynsturer”
- **8.2** Finn og trykk på den oppsynsturen som nettopp ble registrert
- **8.3** Sjekk om alt er riktig
- **8.4** Gå tilbake til hovedssiden.

Finner fram til oppsynsturen men skjønner ikke hvordan man bruker all funksjonaliteten.

Generelle tilbakemeldinger:
Det var enkelt og greit. Var ikke så enkelt å skjønna sveipinga. Kunne vært en pil som viste hva man skulle gjøre første man skal regisrtere.

C.2 Brukertest 2

Selve brukertesten

Brukeren var kjent med systemet for registrering av sau før testen startet.

Oppgave 1: Laste ned kartutsnitt

- Endre navnet på det nedlastede kartutsnittet.

Kartet laster ganske tregt. Brukeren trykker på last ned før kartet har lastet inn.

Brukeren må gå inn og ut av kartet får å få det til å laste.

Brukeren gikk ut av siden og trykte litt feil. Kan være fordi test-leder ba bruker om å skifte navn oppsynsturen og ikke kartutsnitt.

Oppgave 2: Ny oppsynstur

- Legge til deltakere, alle som er med på turen, kan legge til beskrivelse hvis de har lyst.

Registrering av navn gikk strålende.

Oppgave 3: Registrere sauer

- Gå til Kolbjørn Hejes veg 8. Registrer sau på andre siden av plenen
    - 5 sauer totalt
    - 3 hvite, 1 svart og 1 brun
    - 2 søyer og 3 lam
    - 1 grønt og 1 gul slips
    - Register et nytt øremerke
    - Fullføre oppsynstur

Ser på kartet og vet ikke helt hva som kan gjøres. Får litt hjelp til å zoome. Knappen for registrering av sau fungerer ikke første gangen og brukeren blir litt usikker. Det funker på andre forsøk.

Registrering av informasjon går fint.

Bruker går videre fra grensesnittet for registrering av øremerker før det har blitt lagt til. Trykker på nesteknappen framfor å lukke tastaturet og trykke lagre. Skjønner at det ikke ble lagt til og går tilbake. Da fortstå bruker hva som ble feil og fikser det på egenhånd.

Ellers gikk alt fint!

Oppgave 4: Registrere rovdyr

- Gå til Kolbjørn Hejes vei 1A
- Registrer observasjon av gaupe
- Skriv kommentar om ønskelig

Brukere flytter ikke på markøren for registrering før registrering blir gjort.

Ellers blir det registrert en gaupe som ønsket.

Oppgave 5: Registrere død sau

- Gå til hovedbygningen
    - Registrer 1 dødt dyr
    - Skriv kommentar om ønskelig
    - Ta et bilde
    - Fjern bildet som er tatt.

Igjen glemte brukeren å flytte markøren.

Trykker først på bildeknappen for å ta bilde, skjønner fort at det ikke fungerer og trykker deretter på +-knappen. Å fjerne bildet gikk fint.

Oppgave 6: Registrere skadet sau

- Gå til 7030 Trondheim
- Registrer 2 skadede sau
- Skriv kommentar om ønskelig

Glemte igjen å flytte markøren. Ellers alt fint.

Oppgave 7: Fullføre oppsynstur

- Sjekk om registreringene stemmer
- Skriv beskrivelse om ønskelig
- Trykk Fullfør.

Skjønte hvilken knapp som skulle trykkes på. Alt gikk fint .

Oppgave 8: Sjekk liste av oppsynsturer

- Trykk på “Oppsynsturer”
- Finn og trykk på den oppsynsturen som nettopp ble registrert
- Sjekk om alt er riktig
- Gå tilbake til hovedssiden.

Finner fram. «For en fin oversikt».

Generelle tilbakemeldinger:

«Fine ikoner som er lett å forstå uten behov for tekst»

«Vanskelig å se på enkelte dialogbokser om det står ja eller nei»

Forteller til brukeren at brukeren glemte å flytt registreringsmarkøren før det ble gjort en registrering. Kommer da på at det ble glemt. <

C.3 Brukertest 3

Selve brukertesten
Brukeren er kjent med systemet for registrering av sau før brukertesten startet.

Oppgave 1: Laste ned kartutsnitt
- Endre navnet på det nedlastede kartutsnittet.

Brukeren skjønner hva som skal trykkes på. Skjønner hva som skal lastes ned.

Brukeren skjønner med en gang hva som skal trykkes for å endre navnet.

Oppgave 2: Ny oppsynstur
- Legge til deltakere, alle som er med på turen, kan legge til beskrivelse hvis de har lyst.

Brukeren skjønner hva som skal gjøres uten noen spørsmål.

Oppgave 3: Registrere sauer
- Gå til Kolbjørn Hejes veg 8. Registrer sau på andre siden av plenen
    - 5 sauer totalt
    - 3 hvite, 1 svart og 1 brun
    - 2 søyer og 3 lam
    - 1 grønt og 1 gul slips
    - Register et nytt øremerke
    - Fullføre oppsynstur

Skjønner ikke at markøren skal flyttes for å plassere registreringen på rett plass.

Selve registreringen går utmerket.

Klarer fint å registere et nytt øremerke. Velger det også. Alt gikk fint.

Bruker skjønner selv etter registreringen at markøren burde blitt plassert på rett plass før det ble trykt på plussknappen.

Oppgave 4: Registrere rovdyr
- Gå til Kolbjørn Hejes vei 1A
- Registrer observasjon av gaupe
- Skriv kommentar om ønskelig

Skjønner denne gangen at markøren skal plasseres på rett plass.

Velger rett ikon og rett dyr fra lista.

Alt gikk fint.

Oppgave 5: Registrere død sau

- Gå til hovedbygningen
    - Registrer 1 dødt dyr
    - Skriv kommentar om ønskelig
    - Ta et bilde
    - Fjern bildet som er tatt.

Tror først at død sau er plasteret. Skjønner at det er trykt feil når brukeren kommer inn på siden. Går tilbake etterpå og skjønner deretter at det er dødninghodet som er rett.

Alt annet går strøkent.

Oppgave 6: Registrere skadet sau

- Gå til 7030 Trondheim
- Registrer 2 skadede sau
- Skriv kommentar om ønskelig

Alt går fint!

Oppgave 7: Fullføre oppsynstur

- Sjekk om registreringene stemmer
- Skriv beskrivelse om ønskelig
- Trykk Fullfør.

Skjønner med en gang hvilken knapp det er.
alt går fint

Oppgave 8: Sjekk liste av oppsynsturer

- Trykk på “Oppsynsturer”
- Finn og trykk på den oppsynsturen som nettopp ble registrert
- Sjekk om alt er riktig
- Gå tilbake til hovedssiden.

Finner fram men skjønner først ikke at det må trykkes på pila.

Må ha litt mer padding rundt kartet.

Generelle tilbakemeldinger:

Skjønte ikke helt det med flytting rundt av markøren i starten, men tok det etterhvert.

Brukeren påpeker at systemet kanskje burde lagre regelmessig til fil for å ta vare på data om mobilen dør.

«Det var jo veldig lett å forstå da!»

«Symbolene for registrering var greie å forstå, i hvert fall for sau og rovdyr.»

C.4 Brukertest 4

Selve brukertesten

Brukerne hadde tidligere kjennskap til bruk av grensesnittet for registrering av sau.

Oppgave 1: Laste ned kartutsnitt

- Endre navnet på det nedlastede kartutsnittet.

Nedlasting går utmerket. Det samme gjør endring av navn.

Oppgave 2: Ny oppsynstur

- Legge til deltakere, alle som er med på turen, kan legge til beskrivelse hvis de har lyst.

Legger fint til alle medlemmer på turen.

Oppgave 3: Registrere sauer

- Gå til Kolbjørn Hejes veg 8. Registrer sau på andre siden av plenen
    - 5 sauer totalt
    - 3 hvite, 1 svart og 1 brun
    - 2 søyer og 3 lam
    - 1 grønt og 1 gul slips
    - Register et nytt øremerke
    - Fullføre oppsynstu

Brukeren glemmer å flytte rundt på markøren for registrering.

Selve registreringen går pent.

Brukeren lurer på hva vitsen er med pilene når man ikke kan trykke på dem. Hvorfor må man sveipe.

Skjønner registrering av nytt øremerke. Legger det til.

Fullføring går fint.

Brukeren får beskjed om at brukeren glemte å flytte på markøren. Brukeren trodde at registreringen skjedde på plassen hvor gps-lokasjoner vises.

Oppgave 4: Registrere rovdyr

- Gå til Kolbjørn Hejes vei 1A
- Registrer observasjon av gaupe
- Skriv kommentar om ønskelig

Skjønner denne gangen at markøren skal flyttes til rett plass. Velger rett dyr og utfører oppgaven perfekt.

Oppgave 5: Registrere død sau

- Gå til hovedbygningen
    - Registrer 1 dødt dyr
    - Skriv kommentar om ønskelig
    - Ta et bilde
    - Fjern bildet som er tatt.

Brukeren setter markøren på rett plass.

Oppgaven går utmerket!

Oppgave 6: Registrere skadet sau

- Gå til 7030 Trondheim
- Registrer 2 skadede sau
- Skriv kommentar om ønskelig

Brukeren skjønner hva alle ikonene betyr.

Oppgaver utføres utmerket.

Oppgave 7: Fullføre oppsynstur

- Sjekk om registreringene stemmer
- Skriv beskrivelse om ønskelig
- Trykk Fullfør.

Brukeren skjønner hva som skal gjøres og laster opp alt.

Oppgave 8: Sjekk liste av oppsynsturer

- Trykk på “Oppsynsturer”
- Finn og trykk på den oppsynsturen som nettopp ble registrert
- Sjekk om alt er riktig
- Gå tilbake til hovedssiden.

Brukeren finner fram til rett oppsynstur. Skjønner ikke at man må trykke på pilen og at man ikke kan trykke på hele. Kan legge til litt padding på kartet.

Generelle tilbakemeldinger:
Ingen

C.5 Brukertest 5

Selve brukertesten

Brukeren hadde ikke tidligere kjennskap til registreringsgrensesnittet.

Oppgave 1: Laste ned kartutsnitt

- Endre navnet på det nedlastede kartutsnittet.

Skjønner helt fint hvordan kartutsnitt fungerer med zooming og alt. Skjønner også helt fint hvordan man bytter navn.

Oppgave 2: Ny oppsynstur

- Legge til deltakere, alle som er med på turen, kan legge til beskrivelse hvis de har lyst.

Så ikke at seg selv var registrert fra før av. Slettet seg selv til slutt da personen skjønte at personen var automatisk registrert. Kommenterer at «Appen var veldig intuitiv, og jeg skjønte alt jeg skulle gjøre selv om jeg ikke vet hva en oppsynstur er.»

Oppgave 3: Registrere sauer

- Gå til Kolbjørn Hejes veg 8. Registrer sau på andre siden av plenen
    - 5 sauer totalt
    - 3 hvite, 1 svart og 1 brun
    - 2 søyer og 3 lam
    - 1 grønt og 1 gul slips
    - Register et nytt øremerke
    - Fullføre oppsynstur

Skjønner ikke at man skal flytte markøren til plassen hvor sauen skal markeres.

Skjønner ikke at man må sveipe for å endre underkategori. Sliter lenge, må ha hjelp her for å skjønna hva som skal gjøres.

Resten går bra. Velger ikke det nye registrerte øremerket.

Oppgave 4: Registrere rovdyr

- Gå til Kolbjørn Hejes vei 1A
- Registrer observasjon av gaupe
- Skriv kommentar om ønskelig

Skjønner denne gangen at man skal flytte på markøren. Velger også rett ikon og rett dyr fra lista.

«Ikonene er kanskje litt små»

Oppgave 5: Registrere død sau

- Gå til hovedbygningen
    - Registrer 1 dødt dyr
    - Skriv kommentar om ønskelig
    - Ta et bilde
    - Fjern bildet som er tatt.

Trykker først på kameraet og ikke pluss-knappen.
Skjønte at man måtte trykke på bildet for å fjerne det.

Oppgave 6: Registrere skadet sau

- Gå til 7030 Trondheim
- Registrer 2 skadede sau
- Skriv kommentar om ønskelig

Velger rett lokasjoner og ikon. Alt får ypperlig.

Oppgave 7: Fullføre oppsynstur

- Sjekk om registreringene stemmer
- Skriv beskrivelse om ønskelig
- Trykk Fullfør.

Finner fram til rett knapp med en gang.

Oppgave 8: Sjekk liste av oppsynsturer

- Trykk på “Oppsynsturer”
- Finn og trykk på den oppsynsturen som nettopp ble registrert
- Sjekk om alt er riktig
- Gå tilbake til hovedssiden.

Skjønner hvordan alt her fungerer.

Generelle tilbakemeldinger:

- Skriften er kanskje for tynn til at den blir nok lesbar.
- Liker at alt er veldig adskilt.
- Veldig deilig at alt er tilgjengelig via hjem-skjermen.
- Skjønte ikke helt hva strømsparingsmodusen var.
- Knappene er litt for små, ikonene i knappene tar litt for mye plass.
- Rovdyrikonet ser kanskje litt for lite farlig ut, kunne forveksles med en hund.
- Liker at det er et mørkt tema.
- Kunne vært lurt med mer while space på oppsummeringssiden.

Kimia Abtahi og Trym Gjelseth-Borgen

Masteroppgave Datateknologi 2021

NTNU
Kunnskap for en bedre verden