



Norwegian University of
Science and Technology

A Mobile Application and Web System to Monitor Free-Range Grazing Sheep

Frida Meland Schmidt-Hanssen

Master of Science in Computer Science

Submission date: June 2018

Supervisor: Svein-Olaf Hvasshovd, IDI

Norwegian University of Science and Technology
Department of Computer Science

Abstract

This project has explored the opportunity to increase the efficiency of supervising free-range grazing sheep. The government requires that shepherd hikes should be conducted at a minimum of once per week throughout the grazing season. Thus, reports must be submitted on the supervision hikes and its' details.

As of today, the shepherd uses pencil and paper to register observations when he is performing a supervision hike. Thus, where the shepherd has walked, and the positions of observations made are not accurate. Furthermore, the supervision reports must be submitted electronically, and it is time-consuming and redundant work to transfer written notes to electronic ones. So, to raise the accuracy and to avoid redundant work, a mobile application and web system was designed and prototyped.

The mobile application was created for tracking inspection hikes and for registering observations. The hikes can further be synchronized to a web server so the web application can use the hike data. The web application was created to display the hikes in a straightforward manner and for downloading hike summary reports.

A prototype of the system was needed to demonstrate something tangible to a sheep farmer. It is often difficult as a user of a system to understand one's needs before testing something in reality. Thus, the applications were demonstrated to a sheep farmer and the County Governor of Trøndelag, and it provided relevant information regarding supervision and feedback to the prototypes. A user test was also conducted on the prototypes to test the user interface. New functional requirements and user interface changes were discovered from the demonstrations and tests, and added to the future work of the system.

If the new functional requirements and some user interface changes are performed, it is assumed that it can assist sheep farmers with supervision. It can help to avoid redundant work and to save time by an automatic generation of calculations needed in the submitted reports to the government.

Sammendrag

Dette prosjektet har utforsket muligheten for å effektivisere oppsyn av sau på utmarksbeitet gjennom sommeren. Myndighetene krever at tilsynsturer skal utføres minst én gang i uken og dokumentasjon på gjennomførte turer skal sendes inn etter endt beitesesong.

I dag bruker gjeteren penn og papir for å registrere observasjoner mens han er ute på oppsynsturer. Dermed blir ikke turruten og posisjonene til observasjonene gjeteren registrerer helt nøyaktig siden de blir skrevet ned på papir. Videre, så må rapportene etter endt sesong sendes inn elektronisk og det er tidkrevende og dobbelt opp med arbeid å overføre skrevne notater til elektronisk form. For å øke nøyaktigheten og for å unngå unødvendig arbeid, så har en mobil og webapplikasjon blitt prototypet i denne masteroppgaven.

En mobilapplikasjon ble designet for å loggføre oppsynsturer og for å registrere observasjoner. Turene kan videre synkroniseres til en web server slik at webapplikasjonen kan benytte seg av turdataen. Webapplikasjonen ble laget for å kunne se turene man har gått og for å laste ned en enkel oppsummeringsrapport over alle turene man har gått.

En prototype av systemet måtte utvikles for å vise frem noe håndgripelig for en sauebonde. Det er ofte vanskelig som bruker av et system å vite hva man er ute etter før man tester det i virkeligheten. Derfor ble prototypene av applikasjonene fremvist for en sauebonde og Fylkesmannen i Trøndelag, og dette gav relevant informasjon om tilsyn av sau på utmarksbeitet og god tilbakemelding på prototypene. I tillegg ble en brukertest gjennomført for å teste brukergrensesnittet. Nye funksjonelle krav og forbedringer til brukergrensesnittet ble oppdaget fra demonstrasjonene og testene, og dette ble lagt til i fremtidig arbeid av systemet.

Hvis de nye funksjonelle kravene og brukergrensesnittet blir endret på, så vil trolig et system som dette kunne hjelpe sauebønder med oppsyn av sau. Det kan hjelpe ved å unngå dobbelt opp med arbeid og ved å spare tid ved automatiske utregninger som trengs i de innsendte rapportene til myndighetene.

Preface

This master thesis is written by Frida Schmidt-Hanssen as part of the Computer Science education at Norwegian University of Science and Technology (NTNU) in Trondheim. It was conducted during the spring semester of 2018, and it is the final requirement for a degree in Master of Science (MSc) with a specialization in Databases and Search.

Acknowledgements

I would like to thank my supervisor, Svein-Olaf Hvasshovd. Your feedback through our weekly meetings, and on my academic writing, has been valuable throughout the project. Moreover, the meetings with a sheep farmer and the County Governor of Trøndelag would not have been conducted without your efforts.

Secondly, I would like to thank Steingrim Horvli, a sheep farmer located in Oppdal, who took time to answer important questions regarding inspection hikes during the grazing season. He also gave me relevant feedback on the Pecora prototype.

Thirdly, I would like to thank the County Governor of Trøndelag represented by Eva Dybwad Alstad and Arnstein Lyngstad. Alstad and Lyngstad took time to watch a brief demonstration of the Pecora prototype and to answer relevant questions.

Lastly, I would like to thank my aunt, Elisabeth, for borrowing me her old mobile phone so I could quickly test the Pecora application throughout this master thesis, and my friends for taking the time to conduct a user test of the system.

Frida Schmidt-Hanssen
June 7, 2018

Table of Contents

Abstract	ii
Preface	iv
Acknowledgements	iv
List of Figures	vi
List of Tables	viii
List of Listings	ix
1 Introduction	1
1.1 Project Name and Logo	1
1.2 Project Background	1
1.3 Project Description	2
1.4 Stakeholders	2
1.5 Duration	3
1.6 Outline	3
2 Preliminary Study	4
2.1 Understanding the Problem	4
2.2 Existing Solutions	6
3 Methodology	9
3.1 Experiences and Motivation	10
3.2 Research Questions	11
3.3 Design and Creation	12
3.4 Interviews and Observations	16
3.5 Qualitative Analysis	16
3.6 Future Plan	17
3.7 Summary and Discussion	17
4 Requirements Elicitation	19
4.1 Functional Requirements	19
4.2 Non-Functional Requirements	23
5 Technologies	26
5.1 Mobile Application Platform	26
5.2 Cloud Service Platform	27
5.3 Development Tools and Languages	28

5.4	Graphical Design Tool	34
5.5	Map Data and Frameworks	34
5.6	Hardware	37
6	User Interface Design	39
6.1	Mobile Application	39
6.2	Web Application	43
7	System Design and Implementation	48
7.1	System Design	48
7.2	Implementation	56
8	Results and Analysis	72
8.1	Interviews	72
8.2	Observations	78
9	Discussion	86
9.1	Review of the Research Questions	86
9.2	Future Work	90
9.3	Lessons Learned	96
10	Summary and Conclusion	98
10.1	Summary	98
10.2	Conclusion	100
	References	101
	Appendices	
A	Specialization Project	I
A.1	Report and Specification Links	I
A.2	Mobile Application Specification	I
B	Code	III
B.1	Github Repository Links	III
B.2	Server APIs	III
C	Interview 1: Steingrim Horvli	XI
C.1	Roles	XI
C.2	Interview	XI
D	Interview 2: The County Governor	XVII
D.1	Roles	XVII
D.2	Interview	XVII
E	User Test	XXIV

List of Figures

1.1	Project logo for Pecora.	1
2.1	Ewes with different colors on their tie.	5
2.2	Common sheep predators in Norway.	6
2.3	Telespor's radio bell product.	7
2.4	Screenshot from Beitesnap with "Farende Fant" options.	8
3.1	Model of the research process.	9
3.2	The adapted model of the research process.	10
3.3	Screenshot of Trondheim on Skisporet.no's website.	14
3.4	Paper prototype of the web application.	15
3.5	A draft of the system with a cloud service.	15
3.6	The adapted model of the research process: <i>Future work</i>	17
3.7	Gantt diagram for the project period.	18
5.1	Screenshot of the Android Studio IDE.	29
5.2	Screenshot of the Sublime Text user interface.	30
5.3	Screenshot of the XAMPP program for OS X.	31
5.4	Screenshot of the phpMyAdmin user interface.	32
5.5	Screenshot of the Postman user interface with a POST request.	33
5.6	Screenshot of the Trello user interface with a board.	34
5.7	Icons made with the GIMP tool.	34
5.8	Map zoom levels.	35
5.9	Pecora running on an Android emulator.	37
6.1	App screenshots.	39
6.2	App screenshots of downloading an offline map.	40
6.3	App screenshot of a new hike.	41
6.4	App screenshot of walking a hike.	41
6.5	App screenshots of new observation process.	42
6.6	App screenshots of registering an observation of nine white sheep.	42
6.7	App screenshots of hike history.	43
6.8	Web application screenshot of the login display.	44
6.9	Web application screenshot of the main screen.	44
6.10	Web application screenshots of the menu tabs.	45
6.11	Web application screenshots of hike details.	46
6.12	Web application screenshot of three hikes on the map.	46
6.13	Web application screenshot of two hikes within a time interval.	47
6.14	Old and new PDF example of a generated report.	47
7.1	System overview diagram.	48
7.2	Use case diagram for the mobile application.	49
7.3	Use case diagram for the web application.	50
7.4	Android activity flow diagram.	51
7.5	Class diagram of the mobile application.	52

7.6	Layout populated by the <code>HistoryArrayAdapter</code> class.	53
7.7	Web application illustration.	54
7.8	Folder structure for the web application and server code.	54
7.9	The database design in Pecora.	55
7.10	The running EC2 instance in the AWS console.	57
7.11	Details of the RDS instance in the AWS console.	57
7.12	Pecora database in phpMyAdmin.	57
7.13	Web application screenshot of how Leaflet Locate control functions.	71
8.1	Report schema for organized supervision of outfield grazing sheep.	76
8.2	Screenshot of "lamb loss percentage" map layer at NIBIO Kilden.	77
8.3	App screenshots of the new observation registration logic.	85
9.1	Summary table from the supervision document shown in Figure 8.1b.	87
9.2	Possible UI improvements of the mobile application.	91
9.3	Paper prototype for improving the web application.	92
9.4	Suggestion for database design improvement.	94
A.1	Pecora mobile application guideline created by Svein-Olaf Hvasshovd.	I
A.1	Pecora mobile application guideline created by Svein-Olaf Hvasshovd.	II
E.1	The test background and description.	XXIV
E.2	The test steps for the mobile and web application.	XXV

List of Tables

1.1	Summary of stakeholders.	2
3.1	Design and creation plan.	12
4.1	Functional requirements for the Pecora server.	19
4.2	Functional requirements for the Pecora web application.	20
4.3	Functional requirements for the Pecora mobile application.	22
8.1	Overview of the conducted user tests.	80
8.2	Test results from the mobile application tasks.	81
8.3	Test results from the web application tasks.	82
9.1	Future functional requirements for the Pecora system.	92
10.1	Implementation status for all functional requirements from Chapter 4.	99

List of Listings

5.1	Volley dependency in <code>builde.gradle</code>	29
7.1	The database file <code>dbh.inc.php</code> with AWS.	58
7.2	The database file <code>dbh.inc.php</code> with XAMPP.	58
7.3	Server API <code>get-user.inc.php</code> for retrieving user information.	58
7.4	<code>checkLogin()</code> called from <code>onCreate()</code> in <code>MainActivity.java</code>	60
7.5	<code>checkLogin()</code> method from <code>SessionManager.java</code>	60
7.6	Volley POST request for login.	60
7.7	URL for registration API on the server.	61
7.8	Code snippets from <code>MainActivity</code> and its <code>synchronizeData()</code> method.	62
7.9	Camera events from <code>HikeActivity.java</code>	62
7.10	<code>saveImageInExternalCacheDir()</code> method from <code>HikeActivity</code>	63
7.11	PHP session code in <code>index.php</code>	64
7.12	Code snippet from <code>login.php</code>	64
7.13	Code from <code>logout.inc.php</code>	64
7.14	Code snippet from <code>index.php</code>	65
7.15	Map initialization.	65
7.16	Ajax GET request for "most recent hike" in <code>script.js</code>	65
7.17	SQL query in <code>get-most-recent-hike.inc.php</code>	66
7.18	Code extraction from <code>showHikeOnMap()</code> in <code>script.js</code>	66
7.19	Code extraction from <code>getMostRecentHikes()</code> in <code>script.js</code>	68
7.20	Checkbox listener on hike list in <code>script.js</code>	68
7.21	The <code>removeHikeFromMap()</code> method in <code>script.js</code>	68
7.22	Click listener for hike interval button in <code>script.js</code>	69
7.23	Extraction from PDF creation code in <code>script.js</code>	70
7.24	Initialization of sidebar in <code>script.js</code>	70
7.25	Sidebar HTML in <code>index.php</code>	70
7.26	Geo location code.	71
B.1	The server API <code>loginUser.php</code> for login in the app.	III
B.2	The server API <code>registerUser.php</code> for registration in the app.	IV
B.3	The PHP file <code>login.inc.php</code>	V
B.4	The <code>showHikeOnMap()</code> method in <code>script.js</code>	VI
B.5	Click listener for report button in <code>script.js</code>	IX

1 Introduction

This introducing chapter will give the reader some overall information about the project to understand the problem at hand. The contents include the background and a description of the project, as well as information regarding the stakeholders, the duration of the project, and the further outline of the report.

1.1 Project Name and Logo

The project was named *Pecora*, as it is the Italian word for sheep. As a result of this, the project can be referred to as Pecora, as well as the mobile application, web application, app, system, or project. Furthermore, the user of Pecora will be referred to as he or the end user. The project logo is depicted in Figure 1.1.



Figure 1.1: Project logo for Pecora.

1.2 Project Background

Norwegian farmers who have sheep in the outfield pastures must supervise their sheep on a weekly basis during the grazing season. At the end of a grazing season, the farmers are required to submit a report regarding the welfare of their sheep to the government. The reports provide documentation of conducted supervision of the free-range sheep and contain information about loss or injuries and other details from the shepherd's inspection hikes.

The supervision hikes and reports are just one of many tasks the farmer must do to run a farm, and the mentioned tasks take time and effort. Furthermore, the farmers must often apply for compensation for lost sheep at the end of the season due to predators. To be compensated for the loss, the farmer has to fill out an application providing details and proof of death that must be sent to the County Governor. Thus, the details and observations noted during the inspection hikes are highly significant. Regardless of the loss, reports must nevertheless be submitted, and the shepherd must continuously record information.

As of today, the observations during the hike is manually written on paper. After an ended walk, the remarks are typed in on a computer. As the farmer already has enough work on a farm, there is a desire to digitize the supervision and report process to make it more efficient. Hence, it would be less tiresome for the shepherd on a supervision hike, and more efficient for the farmer when reporting to the government at the end of the season.

1.3 Project Description

The purpose of this project was to develop a system with different features that will help to digitize sheep farmer's inspection hikes and reporting to the government. A mobile application to log inspection hikes and a web application for revision and report-generation was planned to fulfill this purpose.

The mobile application should support a complete map of Norway, and the hikes should be logged on the map by GPS. Further, it would feature registration of sheep observations and other elements of interest on the map.

The content from the hikes would later be available in the web application when the shepherd's device connects to a network and synchronizes the data. The user should be able to see several hikes on the map at the same time. At the end of the grazing season, a standardized report should be produced of the hike data and be available for download. Henceforth, the user could submit this summary to the government. This procedure would benefit both the shepherd and the farmer regarding time and effort.

As the mobile application was prototyped in the specialization project Autumn 2017 [1], the web application and cloud service were the primary focus of this project. Nevertheless, the mobile app was further developed to support new and missing features.

1.4 Stakeholders

The stakeholders of the Pecora system is presented in Table 1.1 below.

Table 1.1: Summary of stakeholders.

Stakeholders	Role
Sheep farmers	The end user
County Governor	Receives reports from sheep farmers
Mattilsynet	Receives reports from the County Governor
Svein-Olaf Hvasshovd	The supervisor and adviser

Sheep Farmers The sheep farmers are the target end users for the Pecora system, and its success depends on the sheep farmers' use of the system.

The County Governor The County Governor is a stakeholder who is interested in the detailed supervision reports from the sheep farmers, and thereby the standardized reports Pecora will produce.

Mattilsynet Mattilsynet, the Norwegian Food Safety Authority, is a stakeholder of the Pecora system as they have concerns regarding the welfare of sheep and other animals on the outfield pastures. Hence, Mattilsynet is also interested in the reports and documentation of conducted supervision.

Svein-Olaf Hvasshovd Svein-Olaf Hvasshovd is a professor at the Department of Computer Science at NTNU. He is the project supervisor and adviser. Hvasshovd is a stakeholder because he adds positive value to the project and because he has an interest in the report and the end product.

1.5 Duration

The project started on January 13, 2018. From this date, a total of 21 weeks were given to finalize the master thesis. The report should be delivered no later than June 13, 2018.

1.6 Outline

The further outline of this report is structured in the following way:

- Chapter 2 presents the pre-study, which includes an understanding of the problem at hand and existing solutions in the field of study.
- Chapter 3 presents the methodology, which includes the background and planning of the project.
- Chapter 4 presents the requirements elicitation, which includes both the functional and non-functional requirements.
- Chapter 5 presents the technologies chosen and used to develop the Pecora system.
- Chapter 6 presents the user interface (UI) design of the application and describes how it is used.
- Chapter 7 presents the system design and implementation of the project.
- Chapter 8 presents the results and analysis generated from conducted interviews and user test observations.
- Chapter 9 presents a discussion, future work of the Pecora system, and lessons learned throughout the thesis.
- Chapter 10 presents the summary and conclusion.

2 Preliminary Study

This chapter outlines the preliminary research done at the beginning of the project. Without a proper understanding of the problem to be solved, one can expect an increased degree of failure in addressing the issue. A pre-study is a tool employed to gain familiarity with both the problem space, existing solutions and the potential solution space.

2.1 Understanding the Problem

A sheep farmer has many different and daily tasks to perform on the farm to make everything go around. Looking out for the sheep when they are grazing outfield during the summer is just another task to do, and it is desirable that it becomes simpler and more efficient to conduct.

The government requires that supervision is carried out at least once per week for grazing fields. If there is an emergency situation, such as a predator attack, the surveillance must increase for that period. The farmers in the same area often cooperate and register all sheep that are observed on the supervision hikes and not only their own. It is expected that the typical sheep farmer spends about 10-12 hours on supervision per week throughout the grazing period. Some areas in Norway that are particularly vulnerable to predator attacks do also have organized supervision to cover a more significant grazing area. In organized supervision, some people are selected for a workweek where they must have a minimum of four days per week with supervision. The organized supervision complement the areas the farmers alone are unable to supervise. These hikes come in addition to the one day required by the government.

What the government requires of details and information from the supervision hikes, on the other hand, has no clear guidelines. However, all observations are expected to be registered. The government wants as much information as possible, such as time and place, so they know what is happening throughout the grazing fields. If everything looks normal during the supervision hikes, the farmers also register this.

At the end of the grazing season, all registered information from the supervision hikes is collected from the different farmers in the area and sent to the County Governor within a deadline. The information is sent as a standardized electronic form, which is auto-filled with personal information and other records the County Governor already is in possession of. *Mattilsynet* is also interested in the collected data, which is also a reason for the strict requirement of proper follow-up of the sheep.

The most critical information for the farmers themselves is registration of missing lamb. Additionally, it is vital to register where and how many sheep they observe, if they see predators, where they walk in the terrain, and if they see anything unusual.

The most common observation is missing lamb. If that is the case, the farmer or shepherd starts to suspect that something unusual has happened and must put

in extra resources to cover that area. However, it does not necessarily mean that a predator has taken the lamb. The lamb may have switched to another group of sheep and "replaced" its mother. Thus, it is important to register if there are too many lamb within a group as well. How they can tell if there are missing lamb or too many lamb within a group is by looking at the color encoding of the tie that is attached to the ewe's bell. The color encoding indicates how many lamb the ewe has. There are different rules for the colors used throughout the counties, but the *NSG* (Norsk Sau og Geit) board has recommended that the following encodings should be used:

- Red tie = 0 lamb,
- Blue tie = 1 lamb,
- Yellow tie = 2 lamb,
- Green tie = 3 lamb [2].

Figure 2.1 shows how the tie looks like on the ewes. Usually, the ewe has three lamb or less, but it may also give birth to four or more [3]. However, the national recommendation for the tie color encoding does not include colors for more than three lamb per ewe.



(a) Red tie [4].

(b) Blue tie [5].

(c) Yellow tie [6].

Figure 2.1: Ewes with different colors on their tie.

The main reason for a loss of sheep, and especially the lamb, is due to the predators that live and hunt in the outfield pastures. Thus, it is wanted to register these predators if they are observed. Wolverine, Eurasian lynx, and golden eagle are common sheep hunters to mention some. The animals are depicted in Figure 2.2.

How often, and when, the farmers choose to conduct the inspection hikes depends on the weather conditions. The weather varies a lot in Norway, and if an application is to be used for supervision purpose, it is essential that registration of observations can be done efficiently. If it is pleasant weather conditions, it will not be a problem to spend extra time in the grazing terrain and register complete and detailed observations. If it rains, on the other hand, one would want to complete the hike as fast as possible. Thus, registration with simple clicks and minimal typing is preferred.

It is important to note that Pecora is not supposed to be a complete system for registering all of the grazing sheep that belong to the sheep farmer on each inspection



Figure 2.2: Common sheep predators in Norway.

hike. Firstly, it is a challenge to find all the sheep. Secondly, it is a vast grazing area to cover. Thus, the shepherd tries to get an overview of most of the sheep, and he also registers all other information that is useful and needed. A radio bell with GPS tracking attached to some of the sheep is used as a starting point before the shepherd begins a new inspection hike. Using the radio bell makes it easier to get an idea of where the sheep are currently located.

The sheep are mostly observed from a distance, which also adds more inaccuracy. Binoculars are one of the essential tools for the shepherd, but no matter how advanced the binoculars, it is hard to observe all details, especially reasonably small tags on the sheep. Furthermore, the sheep or lamb could be standing in a position where the tags and tie color is not inspection-visible. Another difficulty and limitation is the sheep's movement, as one may double-register sheep or not observe sheep that are temporarily located behind obstacles. However, these inaccuracies are tolerable for an observation registration system as Pecora. Compared to a bank transaction system where there has to be 100% accuracy, and mistakes are not tolerated, Pecora will not suffer from this lack of precision.

2.2 Existing Solutions

Before developing a new application or system, it is wise to research what systems already exist; which features they have, and what they possibly are missing. If something very similar to an idea already exists, it could be a waste of time and effort to develop it if one is not sure a new solution will be significantly better. For this project, applications that exist for free-range animal tracking and surveillance had to be researched. The following subsections will go into more details about three different products; what they offer, what benefits and disadvantages they have, and what they are missing compared to Pecora.

2.2.1 Telespor

Telespor is a company that develops and sells products and services for electronic surveillance of free-range animals [10]. Their product is a radio bell that can be attached to a sheep (see Figure 2.3). The radio bell sends the sheep's position, which

enables the farmers to see where their animals are located. It also includes a built-in motion sensor that is triggered if something unexpected happens, such as the animal has not moved for the past few hours [10]. SMS and email notifications are supported, and the service can also be integrated with the application Norgeskart¹, making it possible to visualize the individual animals on the map [10].



Figure 2.3: Telespor's radio bell product [10].

A tracking product and service like Telespor is a necessity for farmers to locate the sheep and to get notifications if the sheep may be hurt or dead. However, the product has a relatively high first-purchase price (1118 NOK with a five months subscription [10]). Keeping the bell on all sheep will be an expensive affair relative to the market value of the sheep. Hence, most farmers have it on a certain amount of the sheep and supervision hikes are still needed to keep track of all the sheep.

2.2.2 BeiteSnap

BeiteSnap is an application developed for both free-range animal owners and other people that would like to help with observation registration in the field ("Farende Fant") [11]. The farmer can define the grazing area the animals are likely to be in during the grazing season. Further, the "Farende Fant" can register an observation either by a picture or a location (see Figure 2.4a). It registers the type of animal, and if it is observed, hurt or dead (see Figure 2.4c) [11]. If the observation is within the farmer's defined area, he will get a notification with the registered information. He can also filtrate what kind of information and observations that are desired [11]. Supervision hikes can also be done with this application, and it will be registered as a track log in the application [11]. There is also a service for emergencies, such as predator attacks. This contributes to a faster overview of the situation and may reduce the animal's suffering [11]. The pricing of the service is 1200 NOK per year for an individual user [11].

An app like this is beneficial for the farmers, as opposed to getting phone calls every time people locate sheep in the grazing area. With the application, the observations from people are orderly managed and summarized. However, the farmer may

¹<https://play.google.com/store/apps/details?id=no.avinet.norgeskart>

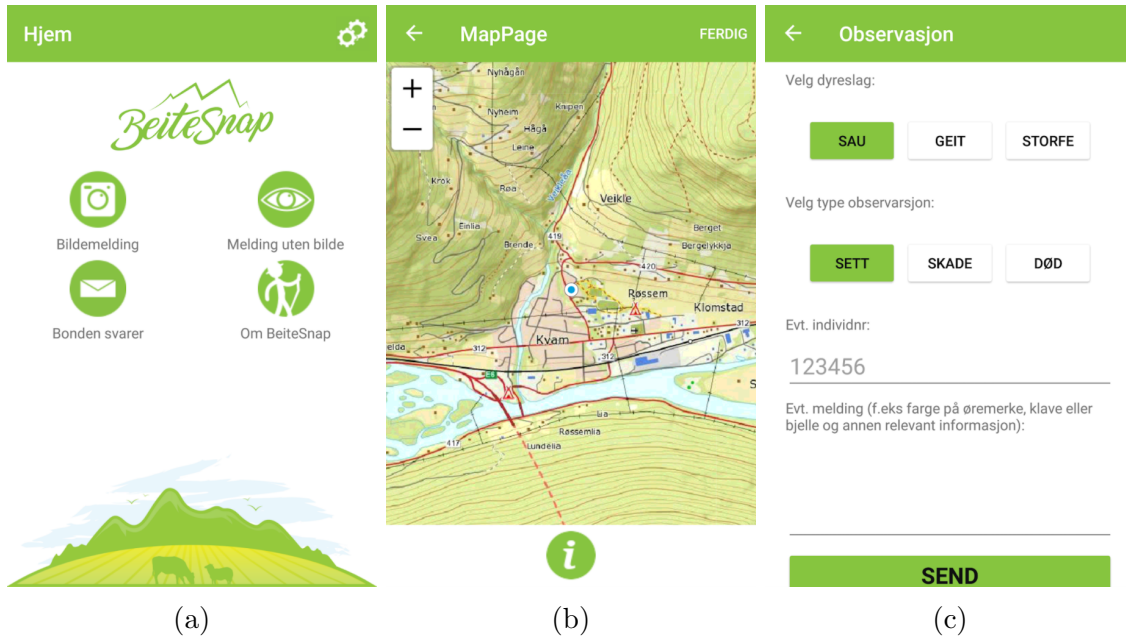


Figure 2.4: Screenshot from Beitesnap with "Farende Fant" options [12].

be interested in more observations than just their animals and their condition, such as if there are predators nearby, hunters, and so forth.

2.2.3 Find My

Find My is a company that provides products and services with satellite tracking technology [13]. It can track animals, humans and objects [13]. Thus, Find My is closer to Telespor than Beitesnap. For animals, one attaches the electronic bells (satellite based) to the animals, which will make them trackable from a smartphone or a computer. If anything happens to the animals, such as tiny movements or moving toward the grazing area boundary, notifications will be sent to the user, and one can easily navigate to the bell that sent the notification [13].

The same things as for Telespor also applies here, that a product like this is a necessity for the farmer to track the sheep, but it is a too expensive investment to monitor the whole sheep herd. Find My recommends that 25% of the pack should be controlled to get a good overview of the animals, although the most ideal of course is to trace 100% [13]. If the farmer has 200 grown sheep, the price for a purchase of 50 bells (25%) plus a charging unit will be 99 000 NOK (1890 NOK per bell and 4500 NOK for a 5-bell charging unit) [13]. So, Find My is a perfect solution for GPS tracking, but it is not an application for supervision hikes to register information about the sheep and surroundings that the County Governor needs at the end of the season.

3 Methodology

This chapter elaborates the chosen methodology for this master thesis. First, the methodology is presented. Second, a more detailed explanation of the six different steps extracted from the methodology follows.

Briony J. Oates (2005) developed a model of the research process, which is depicted in Figure 3.1 [14]. Some components of this model were used to create the methodology for this master thesis.

Research is often based on personal experiences, and people research for a variety of reasons. In advance, it is vital to understand and think about why the research is needed and the motivation behind it. Literature review and studying related artifacts previously produced are also essential. After this, the next step is to derive research questions where one thinks about the wanted achievements and outcome, and where further research or experiments might be needed. To answers the research questions, a strategy must be approached, and the figure below shows six different strategies. Usually, the relationship between the research questions and strategy is a $1:1$ -relation. Further, to enable analysis and evaluation of the research, data generation methods are needed. Several generation methods are often used, which the figure shows with a $1:N$ -relation. The data generated can be either *quantitative* or *qualitative* [14].

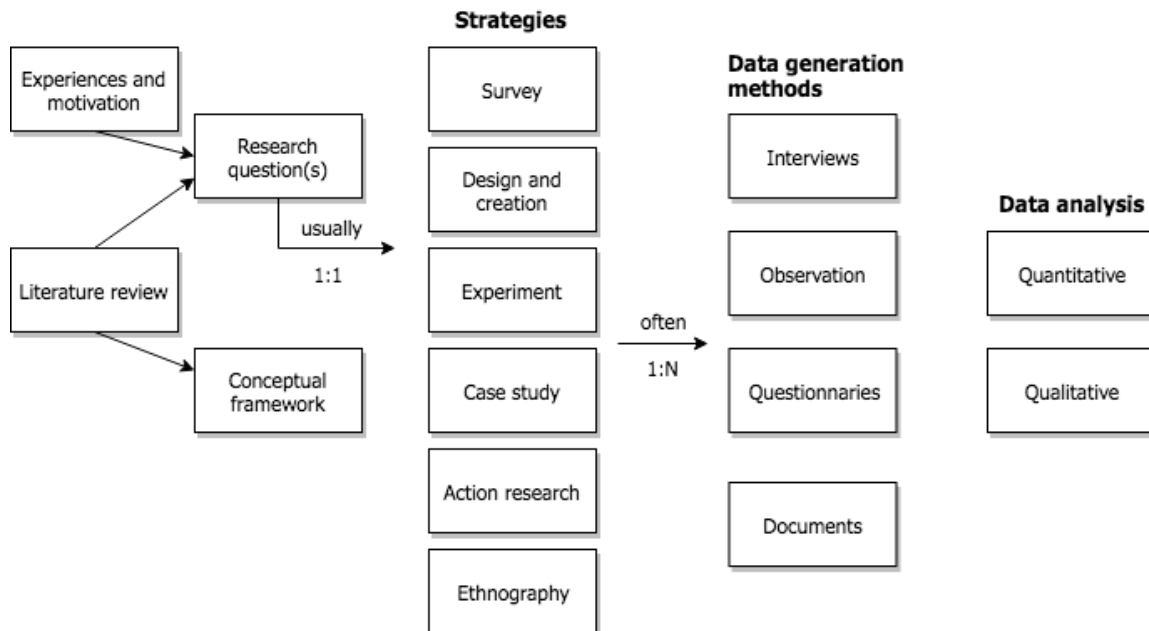


Figure 3.1: Model of the research process [14].

The components used and conducted from Oates' model is shown in Figure 3.2. The *Experiences and motivation* component and *System specification* was performed

and created by Hvasshovd (the supervisor) in advance of the specialization project and thesis (see Appendix A.2). The components within the red, dotted line, on the other hand, was conducted throughout this thesis. The relevant strategy for the project was *Design and Creation*, as the strategy focuses on developing new information technology products, and is often computer-based systems [14]. Further, the *Interviews* and *Observations* were chosen as relevant data generation methods, which resulted in *Qualitative* data analysis. The last four boxes of Figure 3.2 shows the further plan for the Pecora project and what must be done to release a final product.

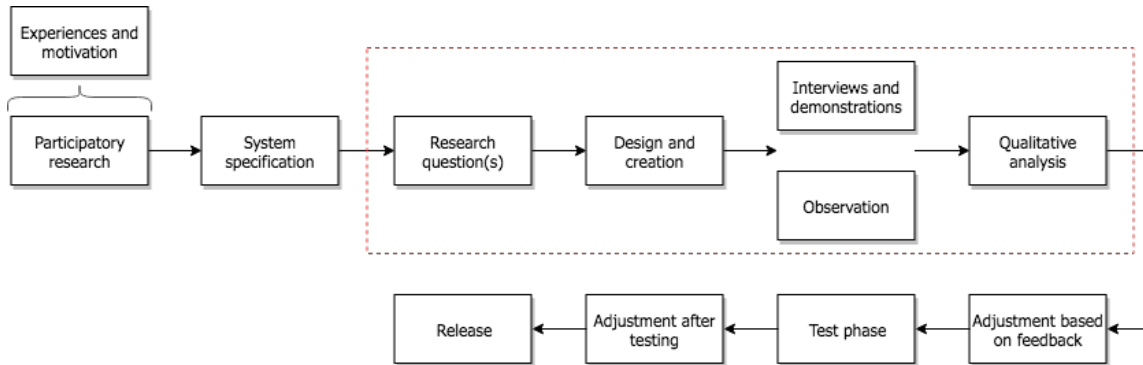


Figure 3.2: The adapted model of the research process.

The following sections will elaborate the steps more in detail. A summary is found at the end of the chapter, discussing the methodology and showing the actual timeline for the project.

3.1 Experiences and Motivation

As mentioned above, the *Experiences and motivation* component was completed in advance of the thesis by the supervisor, and the project was based on experiences from *Participatory research*. Participatory research methods involve research processes where people’s life-world and meaningful actions are under study [15]. Participatory research is often used within qualitative social research [15], and for the Pecora project, a sheep farmer’s supervision routines were studied by participating as a shepherd. Hvasshovd assisted a sheep farmer with supervision in the Oppdal area through two to three grazing seasons and experienced and gained knowledge about how proper supervision is conducted.

Hvasshovd mentioned that rendering the hike route and accurate geographical positions of observations were difficult with only written information in a notebook. In the case of acute events where more supervision would be needed, it could be helpful with better position accuracy. Furthermore, the sheep farmer had experienced a significant loss of sheep due to wolverines and was interested in something that would help reduce the loss.

With this as a foundation, the idea of the mobile application that would track the supervision hikes evolved. As most people carry a phone with them today, utilizing

it and tracking its position could be beneficial. The positions of acute events would be more exact and maybe help to reduce injuries and loss of sheep. Further, when the information registered already would be saved digitally, a summary and report could easily be generated.

Hvasshovd wrote the system specification for the mobile application based on this idea, and the specification was given at the beginning of the specialization project. The specification included sketched prototypes of the application and how it should function (see Figure A.1 in Appendix A), but nothing was set in stone, and there were opportunities for changing and improving the design.

Henceforth, the motivation was to replace the pencil and notebook to register more detailed information, make registration of observations faster, and to generate a summary of supervision hikes more straightforward, making the process more time efficient than today.

3.2 Research Questions

Based on the experiences and motivation, four research questions were derived. The questions will be presented with a small description. For simplicity, the term *research question* has been abbreviated to *RQ*.

- **RQ1: "Will Pecora provide better and more detailed information to the government than how it is today?"**

With GPS tracking, it is expected that the supervision hikes will be more detailed with more accurate observation positions. Thus, resulting in better information to the government.

- **RQ2: "Will Pecora provide better communication between farmers who cooperate with supervision than how it is today?"**

With GPS tracking, it is expected that it will be easier to communicate hike routes between farmers that cooperate in an area as it will be rendered on a map. Further, rendering positions of observations is expected to be easier than explaining with written notes from a notebook.

- **RQ3: "Will Pecora provide better and more detailed information for the farmer himself than how it is today?"**

It is expected that it will be more simple to remember supervision details if the route and observations are rendered on a map instead of by written notes.

- **RQ4: "Will Pecora increase the efficiency of supervision and reporting information to the government?"**

Digital tracking of hikes, an automatic summary of the digital hike information, and calculations of numbers based on the hike information can be expected to increase the efficiency of reporting information to the government.

3.3 Design and Creation

The *Design and creation* of the methodology was the most crucial step for the project. A Pecora prototype was necessary to show something tangible to the relevant users. This way, they could provide constructive criticism to the prototype and essential feedback on what they need in a potential, final product. Thus, it was necessary to plan the design and creation of the prototype properly. Planning is essential when developing information technology systems and a project plan was created early at the beginning of the thesis.

The mobile application originated from a guideline with sketched prototypes given at the beginning of the specialization project (see Appendix A). The guidance made it easy to start with the functional requirements for the implementation phase. A web application was also part of the Pecora system from the beginning, but prototypes and suggestions for that application were not part of the guideline. Thus, the planning of the website began with this thesis.

Early on, a rough creation plan was made. The plan included five different phases; planning, design planning, technology research, and implementation. The steps were not isolated from each other and strictly set, as with the known waterfall method [16]. Thus, tasks could be done in parallel at times, or one could go back and redo a job if needed. However, each phase had a central purpose, and it was important not to get distracted with other future tasks that were necessary to be done in the project. Table 3.1 shows the planned time span of all phases.

Table 3.1: Design and creation plan.

Task	Week
Planning	2-4
Design planning	2-3
Technology research	2-6
Implementation	6-18

The next paragraphs will elaborate the phases a bit more in detail.

Planning The planning of the system began early in the project, and the most important was to structure the system and set boundaries. As the project had a limited amount of time, the limitations were especially significant to consider. With a software system, there are always more tasks that can be implemented to optimize and improve it. Hence, it was crucial to conduct a requirements elicitation so the implementation would not affect all other phases of the project, like documentation, or testing to mention some. So, in this phase, a rough project plan was made, requirements were set, design planning began, and research of technology. The requirements of Pecora can be found in Chapter 4.

Design Planning Design and user interface planning was necessary for the web application as it was not part of the guideline mentioned earlier. The most straightforward and least expensive form of prototyping is paper prototyping, so it was easy to choose that method. Details that would be interesting for a sheep farmer were thought through, and these were tried to be presented in an orderly manner drawn on paper. Results from the design planning can be found in Section 3.3.1. Design planning of the cloud service and further preparation of the mobile application were needed as well, and the details can be found in Sections 3.3.2, and 3.3.3, respectively. The final user interface of both the mobile and web application is found in Chapter 6.

Technology research As experience with cloud services and website development were lacking, the research phase became quite relevant. Firstly, it was wise to get an overview of what facilities existed, and which of them offered proper documentation. What people mostly prefer to use was also an aspect to consider. It was quickly done to get an overview of the services, but the hard part was actually to choose one of them. So, the research phase got a bit longer than expected, but it was essential to select the right service and find proper tools for the implementation phase. Searching for and doing tutorials was also a big part of the research phase. More details of the services and technology chosen will be elaborated in Chapter 5.

Implementation The system could be divided into three parts from the beginning; the app, the website, and the server. Thus, the implementation phase got split into three as well. As a prototype of the app was already developed, it was wise beginning with the server logic and web development to get started with those parts of the system as well. However, to get started with the hike tasks of the web development, the hike data from the mobile application was essential. Thus, the server logic and synchronization from the application needed to be dealt with before web tasks could be started. More details of the implementation are found in Chapter 7.

3.3.1 Web Application Design

The usage of the web application versus the mobile application is entirely different in Pecora, as described in Section 1.2. Thus, developing the web application based on the app was pointless. As showing the hike trails with information regarding the sheep's whereabouts were considered the essential feature of the application, this should be displayed smoothly and cover most of the screen.

Inspiration was found in the website of Skisporet², which shows the real-time information regarding snow grooming for the ski trails in Norway (see Figure 3.3). The ski trails' colors are decided by how recent the trail is groomed and the time range used is defined by the box shown to the right of the figure.

After having a look at Skisporet's website, ideas were flourishing. The map should be full screen with small control buttons, such as a menu and zoom controls, and the

²<https://skisporet.no/>

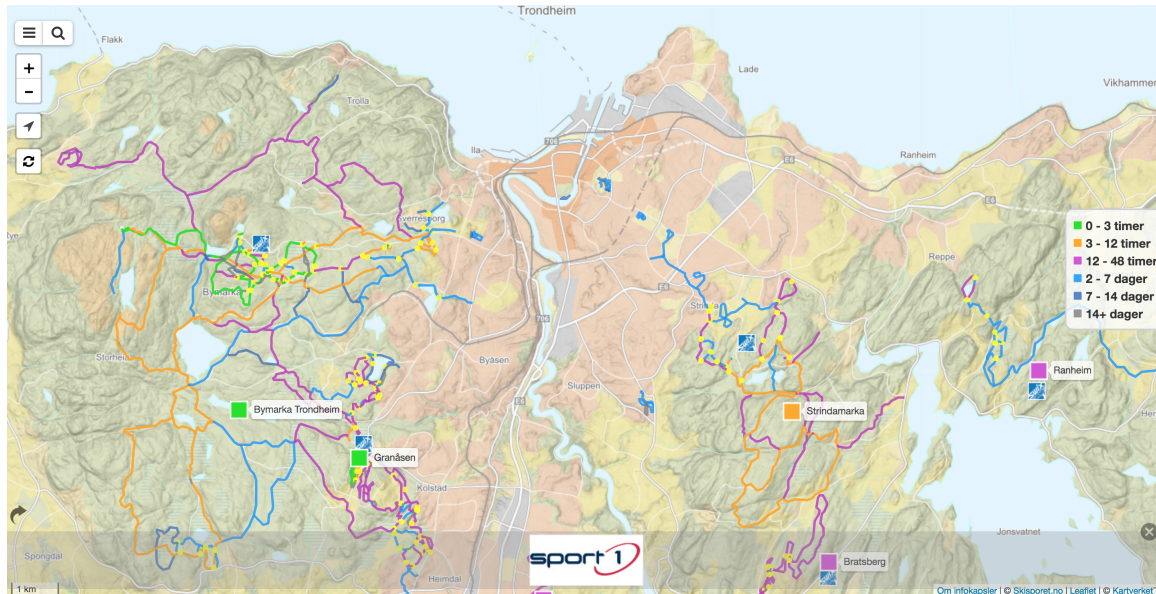


Figure 3.3: Screen shot of Trondheim on Skisporet.no's website [17].

hike trails should have different colors. These ideas were further put into action by drawing a paper prototype, and they can be found in Figure 3.4.

Figure 3.4a shows how one hike was planned to look like on a full-screen map. The trail and observations should be displayed with a color chosen for that hike. In the top-left part of the screen, there should be a burger menu, which is commonly used in graphical user interfaces [18]. Below the menu, a list of the three most recent hikes should be shown with different colors where the hike already shown on the map is checked. Figure 3.4b shows how the hike details should be displayed in pop-up windows when they are clicked on. Figure 3.4c shows how it should look when the three most recent hikes are checked in the list and shown on the map with different colors.

It was also planned that options for viewing hikes within a time interval and generating a report of the hike data should be available in the menu.

3.3.2 Cloud Service Design

To better understand how the cloud service should function in Pecora, a simple figure was made (see Figure 3.5). The figure shows that the mobile application should be able to synchronize hike data to a database in the cloud and that the web application should be able to retrieve and show hike data from the database. Additionally, the cloud system should validate user login and be able to register new users. Further planning of the cloud included testing of different technology and watching tutorials to see what service would fit Pecora the best. More details of the technology will be elaborated in Section 5.2.

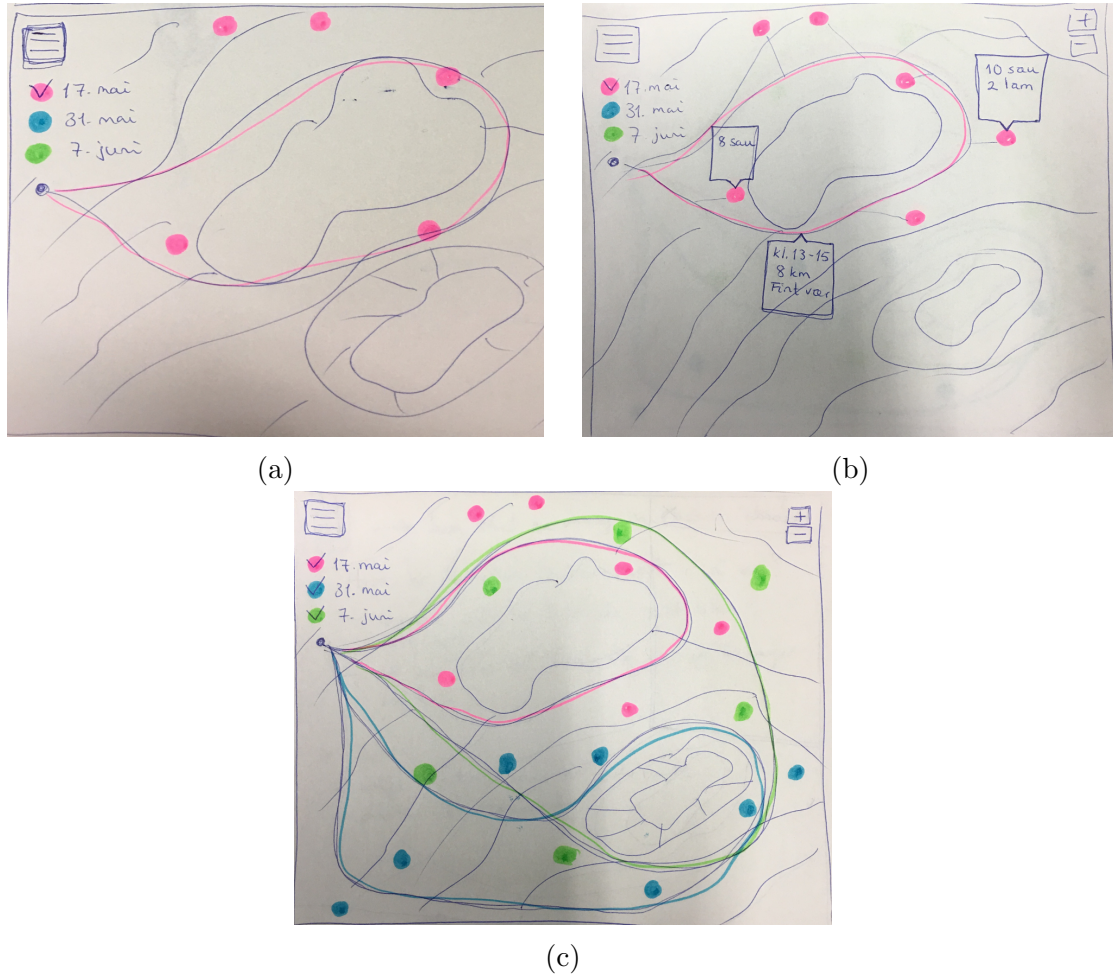


Figure 3.4: Paper prototype of the web application.

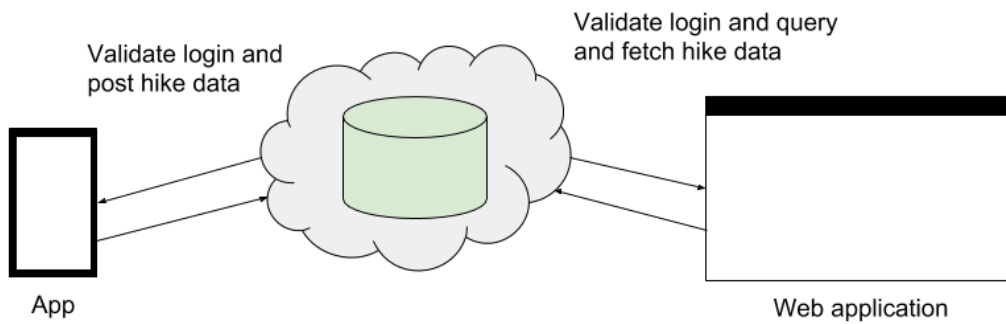


Figure 3.5: A draft of the system with a cloud service.

3.3.3 Mobile Application Design

When the first Pecora mobile application prototype was created, some essential features were not implemented due to time restrictions (see Table 4.3). User profiles were not included in the first version, and this was vital to implement for further

development of Pecora.

As mentioned in the *Implementation* paragraph above, the hike data from the app was important to synchronize to a database so the main tasks of the website development could be started as soon as possible. Hence, the plan for the mobile application became as follows:

1. Set up a connection to the database.
2. Develop logic for insertion of hikes into the database.
3. Add user logic and add user ID to each hike.
4. Implement remaining requirements when other prioritized requirements for the other systems are completed.

3.4 Interviews and Observations

As mentioned earlier in the chapter, the data generation methods found to be the most relevant for the project included *Interviews* and *Observations*.

The interview method is a particular kind of conversation where, at least in the beginning of the interview if not all the way through, the researcher controls the agenda and will ask most of the questions [14]. Interviews with relevant users as a sheep farmer and the County Governor of Trøndelag were conducted with other groups working on similar thesis projects. A question list was produced in advance of each interview, and the researchers asked most of the questions throughout the meetings. More details of the interviews can be found in Section 8.1.

The observation method is done by watching and paying attention to what people do, rather than what they report they do [14]. The method was used through user testing on a selected number of people, where their actions were observed. The user test was especially interesting and useful as only one person performed Pecora's implementation. Every task implemented is very logical for the implementer, and it is not easy to see possible improvements or to find mistakes as she knows precisely how the system functions. As making errors often reveal bugs and difficulties in the user interface, testing with other people was crucial and beneficial. More details of the observations can be found in Section 8.2.

The two methods above resulted in *qualitative data*, which will be elaborated and discussed in the next section.

3.5 Qualitative Analysis

Qualitative data includes all non-numeric data, for example, found in interview tapes [14]. The interviews and observation sessions were audio recorded and later transcribed for better analysis.

Qualitative analysis is not always a straightforward task as there are no established procedures for it. *Quantitative* analysis, on the other hand, has mathematical and statistical procedures. Hence, qualitative analysis depends more on the skill of the

researcher to find patterns and themes within the data [14]. However, for this project, it was not massive amounts of data to analyze, which made it quite effortless to extract the useful information from both the interviews and observations. What was interesting to figure out through the analysis was if Pecora was a helpful tool, what could be improved, changed, or removed, and what was missing. The results from the data generation methods are found in Chapter 8.

3.6 Future Plan

The last four boxes of the adapted model of the research process were mentioned earlier as the future work needed to release a final version of Pecora (see Figure 3.6). As a time limit and boundaries were set for the project, there was not enough time beginning with these components. The testing and feedback process of software development is one of the most crucial phases, and it is essential for releasing a finished product.

Thus, the future work would be to implement adjustments to the system based on the qualitative analysis from the interviews and observations, before the relevant users would perform a proper test of the prototype over a set period. After testing, an adjustment would be performed once again based on the feedback from the relevant users, before a release of the final product.

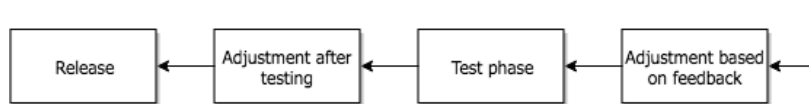


Figure 3.6: The adapted model of the research process: *Future work*.

More details of future work is found in Section 9.2.

3.7 Summary and Discussion

The methodology chosen for the project worked well in practice, and the Gantt diagram³ in Figure 3.7 shows the actual time span of each step. The time span is displayed by months and weeks. The experience and motivation and research questions are not part of the diagram as the steps were already conducted before and through the specialization project. Using the research process model gave more structure to the project, and made the goal and wanted achievements more clear. The other steps are listed in the diagram, in addition to a *Documentation* step.

Documentation was added as it is one of the most vital tasks in software development. Perhaps somebody is continuing one's work someday and need to understand how it is structured and implemented, and proper documentation will make this achievable. The documentation phase did not start from the beginning, but a log was written each working-day so documenting would be simple later on when initiated. As seen in the diagram, documentation began in week nine of the thesis.

³<http://www.gantt.com/no/>

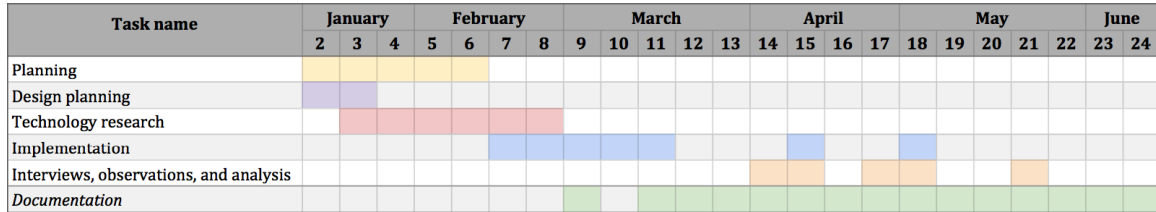


Figure 3.7: Gantt diagram for the project period.

Even though the methodology fitted the project well, some disadvantages and advantages of the strategy and analysis could be discussed.

The design and creation strategy had the purpose of creating a prototype to demonstrate to relevant users of Pecora, which meant it was developed without communication with the users in advance. If a meeting with the sheep farmer were conducted before development began, the prototype would most likely have turned out better and more optimal, seen in retrospect. On the other hand, if only an idea had been presented to the relevant user and not a tangible solution, he may not have known what he wanted and would not have given the appropriate feedback that was gained through the prototype.

Further, there are both disadvantages and advantages of qualitative analysis. The cons are that one can not make robust conclusions because of unrepresentative selection [19]. There exist more disadvantages of qualitative analysis, but the one mentioned was the most relevant for this project. Only one sheep farmer was interviewed, which qualifies for an unrepresentative selection. The advantages, on the other hand, is that one can reconcile an idea with few people early in a process. For Pecora, interviewing one relevant user resulted in essential information for further development and outfield grazing sheep in general. Furthermore, it gives the interview subjects the opportunity to elaborate their opinions and allows for follow-up questions from both interviewers and its interview subjects [19].

4 Requirements Elicitation

This section presents the solution that was planned with the understanding of the problem in mind, and what could be useful for the sheep farmer during and after inspection hikes.

4.1 Functional Requirements

Functional requirements define what the system does and what it should be able to do [20]. The requirements list is typically made before the development begins, which makes planning and delegation of tasks and responsibility easier to conduct.

The functional requirements of the Pecora server and the web system are shown in Table 4.1 and 4.2. The features that were not implemented in the specialization project is shown in Table 4.3, along with some new functionality for adapting to the server and web system. Each of the requirements has an ID, a priority spanning from low to high, and a description. The priority was important for ordering and prioritizing the tasks in the implementation phase.

Table 4.1: Functional requirements for the Pecora server.

ID	Priority	Description
FR1	High	Host Pecora’s website and database
FR2	High	Authentication of users
FR2.1	High	Validation of input
FR2.2	High	Check login credentials in both the mobile and web application
FR3	High	Registration of a new user
FR3.1	High	Validation of input
FR3.2	High	Hash the user’s password
FR4	High	Synchronize hikes from the mobile application
FR4.1	High	Add hikes from the app to the database if they do not already exist

The server system plays a vital role in Pecora as it is responsible for hosting the website and for handling database requests (FR1). Authentication of users is an example of this (FR2), and the server must check login credentials for both the app and web application, in addition to validating the input. A second responsibility is registration and validation of new users (FR3). It is crucial that the server verify user input so no malicious code can be injected and harm the database (FR2.1 and 3.1). Another security and safety measure is to hash the user passwords and not store it in plain text (FR3.2). Synchronization of new hikes must also be supported, and

a check for existence is required before a hike is inserted into the database (F4 and FR4.1).

Table 4.2: Functional requirements for the Pecora web application.

ID	Priority	Description
FR5	High	Use map data of Norway from Kartverket
FR6	High	Navigation and zooming options on the map
FR7	High	Have a menu bar that is expandable
FR8	High	Show hikes on the map
FR8.1	High	Retrieve and parse hikes data from the database
FR8.2	High	Show the hike route and observation markers
FR8.3	High	If the user clicks on the route, show hike details in pop-up
FR8.4	High	If the user clicks on observation markers, show information belonging to the specific observation
FR8.5	Medium	Show the hikes with different colors
FR8.6	High	Show different icons for observation points and observations
FR8.7	Low	Show the most recent hike when the user logs in
FR8.8	High	If the user does not have any hikes yet, the map should show a specified point on the map
FR9	Low	Have a button for showing the user's current location
FR10	High	Show 5 of the most recent hikes in a list with checkboxes
FR10.1	Medium	The user should be able to choose a time interval and get a list of hikes within that interval
FR10.2	Medium	The checkbox belonging to the most recent hike should be checked
FR10.3	High	If a checkbox is checked, show the belonging hike on the map
FR10.4	High	If a checkbox is unchecked, remove the hike from the map
FR10.5	High	The user should be able to see several hikes on the map at the same time
FR10.6	Low	If the user does not have any hikes yet, a label should say so
FR11	Medium	Show user profile information and a logout button in menu
FR11.1	High	If the logout button is clicked the user is logged out and redirected to the login screen
FR12	High	A generate report option be in the menu
FR13	High	If the user is not logged in, the login page should be displayed
FR14	High	A user can only log in if a user is created through the app

Table 4.2 contains 10 requirements, including some sub-requirements.

The first two requirements concern the map data, which is to be retrieved from Kartverket, and that the map needs navigation and zooming options (FR5 and 6). These requirements are required for an acceptable detail level of the Norwegian terrain, which resulted in high priorities.

An expandable menu (FR7) is suitable on Pecora's website as it will not occupy map space before the user clicks it. A menu, in general, is essential for user options, but it is unnecessary that it fills the display at all times. Henceforth, a high priority was given to this requirement. A standard sidebar menu would have been the alternative.

FR8 is one of the primary functional requirements of Pecora and concerns showing hikes on the map. This requirement has eight sub-requirements, and naturally, they got a high or medium priority. The first is retrieving the hike data from the database and parsing the objects, especially those in JSON⁴ format, as will be elaborated later in Section 7.2. After the parsing is complete, the hike route and observation markers must be put on the map (FR8.2). If the user clicks them, further details must be shown in a pop-up window (FR8.3 and 4). FR8.5 and FR8.6 state that the hikes and markers must be different. This requirement will make the user interface more clearly. Lastly, when the user logs in to the website it should show the last walked hike (FR8.7), but if the user does not have any belonging hikes yet the map should show some specified point on the map (FR8.8).

Functional requirement 9 says to have a button for showing the user's current location. This function can be user-friendly as the user may scroll around on the map, and maybe he needs to be animated back to a known area to get an overview once more. As this feature is not vital, it got a low priority.

When planning how to organize hikes and show hikes to the user in the best possible way, a list of the hikes seemed like a good idea. Most of these sub-requirements got a high or medium priority as they were crucial for the system. Sorting hikes on dates are wise as we often associate events with time. So, the five most recent hikes should be shown to the user in a list with checkboxes (given the user has 5 or more hikes) (FR10). FR10.1 concerns if the user should be able to choose a time interval to display hikes within it. This can be useful for previous hikes and to find a hike where something significant happened. FR10.2 states that the most recent hike should be checked in the list by default. If one of the checkboxes is checked, the selected hike with all information should be shown on the map (FR10.3). If it is unchecked, remove the hike from the map (FR10.4). Since the hike list is displayed with checkboxes and not radio buttons (where only one element can be active), this means that more hikes can be shown on the map at the same time (FR10.5). This could potentially be an exciting feature when a shepherd has walked many inspection hikes; perhaps a pattern of where the sheep stay can be observed. FR10.6 concerns feedback to the user. If the user does not have any hikes yet, feedback with "No hikes" should be given to the user so he will remember to synchronize hikes from the mobile application.

Functional requirement 11 and 11.1 concern seeing profile information in the menu.

⁴<https://www.json.org/>

The user will also be able to log out of the website from this page and be redirected to the login page. FR11 got a medium priority as it is essential to show which user is signed in. If more critical options or information were included, such as changing passwords or other details about the farm, it would have gotten a higher priority. FR11.1 got a top priority as the user should be logged out appropriately. By showing the login screen, the user can be reassured that he is logged out.

FR12 embraces the requirement to generate a standardized report based on the data from all hikes conducted during the grazing season. If this report were something similar to what the farmers must send to the County Governor at the end of the season, it would save them time and effort. The report could easily summarize how many sheep were observed at each hike, and possibly how many lambs found to be missing.

Lastly of the website requirements are FR13 and 14, which concerns authentication of users. If the user is not logged in, he should be redirected to the login page. If a user has not registered in the mobile application, he should not be able to sign in to the Pecora web application.

Table 4.3: Functional requirements for the Pecora mobile application.

ID	Priority	Description
FR15	Medium	The user should be able to register movement and area of an observation
FR16	Medium	The user should be able to register an existing observation with a new location
FR17	High	The user should be able to take a picture of an observation
FR18	High	If the user logs in/is logged in, the main activity should be displayed
FR18.1	High	If the user is not logged in, the login activity should be displayed
FR18.2	High	If the user has not registered, he should not be able to log in
FR18.3	High	The user should be able to register a new user profile
FR19	High	The app should synchronize new hikes to the database server

Table 4.3 above contains requirements for the mobile application that were not implemented in the specialization project. As they are essential for Pecora, they were further included in this project. Additionally, some new requirements were added to match with the website, for example, user session logic (F18).

FR15 and 16 concern registering movement of observations and registering a new location for an already existing observation. FR15 can be useful information for the shepherd when walking new supervision hikes as he may get a better clue of the sheep's location. FR16 could be useful for a better precision of the observation's position if one sees the same one a second time. Hence, both requirements have gotten a medium priority.

The functional requirement FR17 is about taking a picture of an observation. This requirement could be crucial for the sheep farmer regarding insurance and compensation if a predator kills a sheep. Images taken of the observation could serve as proof and perhaps help to get compensation paid. Naturally, this requirement got a high priority.

The last requirement, FR19, involves synchronizing the user's new hikes to the server. The application should be able to sync the hikes automatically when it detects a network connection.

4.2 Non-Functional Requirements

In addition to the functional requirements, there are some non-functional requirements or quality attributes wanted for Pecora. Non-functional requirements guide and constrain the architecture [20], and it is a measurable or testable property of a system that is used to indicate how well the system satisfies the needs of its stakeholders [21].

Five non-functional requirements were found relevant for Pecora, that is; availability, usability, interoperability, security, and performance. Each of these non-functional requirements will be explained in the next paragraphs in no specific order.

Availability Availability refers to a property of software that it is there and ready to carry out tasks when one needs it to be [21]. This term could be mistaken for reliability, but rather than having the notion of recovery, availability is about minimizing service outage time by mitigating faults [21].

Availability is crucial for online systems, such as Pecora. The users will expect the mobile application or the website to carry out all requested tasks when they need it to be. Should this requirement not be met, Pecora will lose confidence and credibility. One way to prevent availability issues could be to use established and experienced companies, such as Amazon⁵, Google⁶ or Azure⁷, to run and operate the servers.

Usability Usability is a quality attribute which is concerned with how easy it is for the user to accomplish the desired task and the kind of user support the system provides [21]. Usability has shown to be one of the easiest and cheapest ways to improve a system's quality. Learning system features, using a system efficiently, and minimizing the impact of errors are areas usability comprises [21]. Thus, usability seems to be a critical quality attribute for Pecora.

The mobile application and website have to be easy to learn for new users as already existing applications and websites today are well-developed and have come a long way with user-friendliness in mind. If Pecora is difficult to use or learn, the risk of losing end-users will rise. As a sheep supervision system like this is relatively unknown for most people, some time might be expected to go before the users are

⁵<https://aws.amazon.com/>

⁶<https://cloud.google.com/>

⁷<https://azure.microsoft.com/>

100% sure with the full functionality of the applications. However, the learning curve should be steep. As the mobile app will mostly be used in the field, it is essential that observations will be easy to register as the weather conditions may be varying. Furthermore, it could be useful to have a support system where end users can easily contact and ask questions, and also have a frequently asked questions (FAQ) page on the website with the most common issues and problems.

Interoperability Interoperability is about the degree to which two or more systems can usefully exchange meaningful information via interfaces in a particular context [21].

As the Pecora system has been extended from an app to a system with a website and server logic, interoperability is an important quality attribute. The three different components have to communicate and exchange meaningful information for the system to function correctly. The mobile application needs to communicate with the server to authenticate and register users, and to synchronize hikes. Further, the website needs to communicate with the server to authenticate users as well and to retrieve and show the hike data from the app to the user.

Security The security quality attribute is a measure of the system's ability to protect data and information from unauthorized access while still providing access to people and systems that are authorized [21]. Three terms often characterize security;

- Confidentiality: The property that data or services are protected from unauthorized access,
- Integrity: The property that data or services are not subject to unauthorized manipulation, and
- Availability: The property that the system will be available for legitimate users [21].

As Pecora stores sensitive information, such as user information and location data, security is an important non-functional requirement. Security had to be considered in the implementation from the beginning of, and the three terms mentioned above was a good guideline. To prevent security risks, it is a must to include security from the beginning and to go through the possible attacks the system could face. A simple example could be an attacker who gets access to the database, and the user passwords are stored in plain text. Thus, the user passwords should be hashed to mitigate damage if an attacker got access to the database.

Performance Performance is a non-functional requirement which concerns time and the software system's ability to meet timing requirements. When events occur, as requests from users or other systems, the system must respond to them in time [21].

As of today, most systems respond fast to its end-users. People are becoming more and more impatient if a system is slow and does not react in time, so performance

is an important quality attribute to consider. An example of increasing performance could be to avoid unnecessary calls to the database, as this is often a bottleneck in systems and will introduce latency.

Performance is often linked with scalability too. If Pecora is to be used by many users someday, the system will have to tolerate multiple requests from different users at the same time and respond to them in time. As earlier mentioned with availability, using the big companies for cloud services could be wise in this case too. The cloud services are made for automatic scalability and will maintain the network-connected hardware [22].

5 Technologies

This technology chapter declares the background for technology decisions made for the project. It includes the mobile application platform and cloud service chosen for the system, development environments, development tools, and the map frameworks used for both the mobile and web application. Common for all technology choices is that they are offered as free (or free tier) and open source software.

5.1 Mobile Application Platform

When making mobile applications it is common to develop a solution that is running on both Apple's iOS⁸ and Google's Android⁹ platforms. Other operating systems, such as Windows¹⁰, are also used, but as 99.6% of new smartphones bought run either iOS or Android [23], it most common and profitable to develop for those two.

iOS development requires that one use the Xcode IDE with iOS SDK. Xcode supports different programming languages, but Swift is the most used by developers [24]. For Android development, the Android Studio IDE is widely used. With Android Studio one uses the Android SDK and Java¹¹ as the default programming language [25].

Many developers are of the opinion that development for iOS is more natural due to the fact that Xcode and iOS SDK are specially made for iPhones, as opposed to the fragmentation in Android that refers to the vast number of devices that come in all shapes and sizes, as well as with considerable differences in performance and screen sizes. Another problem with the Android fragmentation is the many active versions at the same time. To build an app that is compatible with all versions is challenging [26].

The Pecora mobile application was developed for Android, and the decision was made early in the specialization project due to limitations such as time and experience. In retrospect, it has been discovered that web technology, such as Apache Cordova¹², could have been the most optimal platform for developing a solution for both iOS and Android. It would also have been more beneficial to the map technology and web application in mind as the code could have been reused.

However, the earlier experience with programming an Android app affected the decision to develop for that operating system. It was also discovered that it exists a library with example projects for working on map data and download of offline maps, which was further motivating for the decision. The library used in the application

⁸<https://developer.apple.com/library/content/referencelibrary/GettingStarted/DevelopiOSAppsSwift/>

⁹<https://www.android.com/>

¹⁰<https://www.microsoft.com/en-us/windows/phones>

¹¹<https://www.java.com/>

¹²<https://cordova.apache.org/>

will be presented in Section 5.5.

5.2 Cloud Service Platform

One of the critical technology decisions that were made included which cloud service platform to choose. Cloud computing has made a profound impact on innovation and the economics of business overall, as it introduces more flexibility and reduces costs [22]. It is a model that enables ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., servers, and storage) that can be rapidly provisioned and released with minimal management effort or service provider interaction [27]. Thus, cloud computing services allow one to focus more on the applications one are making and innovation rather than to worry about the expenses of hardware and infrastructure management [22].

If Pecora is to be a popular application for inspection hikes, cloud computing could be beneficial as it dynamically assigns and reassigns resources according to consumer demand, so-called *resource pooling* [27]. Many of the services also have a pay-per-use or charge-per-use plan as the cloud systems control and optimize the resource use appropriate to the type of service (e.g., storage, or active user accounts) [27].

There are three different cloud computing service models, Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [27]. SaaS provides the consumer with applications running on a cloud infrastructure, such as web-based email [27]. With PaaS, the consumer can deploy consumer-created applications or similar to the cloud infrastructure [27]. IaaS provides the consumer with processing, storage, networks and other fundamental resources where the consumer can deploy and run the software, which can include operating systems and applications [27]. Characteristic of the three models is that the consumer does not manage or control the underlying cloud infrastructure [27], but the degree of control increases from SaaS to IaaS. The PaaS service model is the model fitting for this project as it is a consumer-created application deployed onto the cloud infrastructure.

The cloud service platforms considered were Amazon Web Services¹³ (AWS), Microsoft Azure¹⁴, and Google Cloud Platform¹⁵ as these are some of the most popular cloud platforms according to Clutch.co [28]. Of course, there are many other alternatives out there today, but as they are the services most heard of, talked about, and used the most, it was a good way to limit the options.

As Google's Android technology already had been used in the Pecora mobile application, it became natural to look at the cloud platform they also provide. Firebase¹⁶ seemed like a good option as it would offer a serverless service for both mobile applications and web clients. However, no matter how good the service appeared, it was hard to know exactly how to get started when one already had a functioning app. Further on, Azure was looked at and researched a bit, but it was quickly discarded as Google Cloud and AWS seemed like better options in this case.

¹³<https://aws.amazon.com/>

¹⁴<https://azure.microsoft.com/>

¹⁵<https://cloud.google.com/>

¹⁶<https://firebase.google.com/>

In the end, AWS was chosen as they provide a good tutorial for setting up a cloud server and connecting it to a relational database. Additionally, it is free of charge up to a specific amount of use. The Amazon Elastic Compute Cloud¹⁷ (Amazon EC2) and Amazon Relational Database Service¹⁸ (Amazon RDS) was investigated and used until the free-tier period was up. How EC2 and RDS was used will be further elaborated in Section 7.1.4.

5.3 Development Tools and Languages

Android Studio Android Studio¹⁹ was chosen as the integrated development environment (IDE) as it is the official IDE for Android application development. It is a free IDE and is based on JetBrains's IntelliJ IDEA²⁰. Android Studio offers many good features for developers as:

- Instant run.
- Intelligent code editor.
- Fast emulator.
- Robust and flexible build system.
- Development for all Android devices.
- Code templates [25].

A screenshot of Android Studio's user interface can be seen in Figure 5.1.

Java Java is the standard programming language in Android Studio, so naturally, this became the programming language for the mobile application. Java is a general-purpose language that is concurrent, class-based, object-oriented, and designed to have as few implementation dependencies as possible [29]. It is a widely used language with a large user base, which made it an appropriate language to use for the app when research and help might be needed. Experience with Java from earlier projects was also beneficial.

XML XML²¹ (Extensible Markup Language) is the standard language for the graphical design in Android Studio. Hence, this language was used for the different layouts of the mobile application. XML is a markup language that defines a set of rules for encoding documents in a format that is both human- and machine-readable [30]. XML is easy to read and well-structured, so with the code editor in Android Studio, it was an efficient language to use.

¹⁷<https://aws.amazon.com/ec2/>

¹⁸<https://aws.amazon.com/rds/>

¹⁹<https://developer.android.com/studio/index.html>

²⁰<https://www.jetbrains.com/idea/>

²¹<https://www.w3.org/XML/>

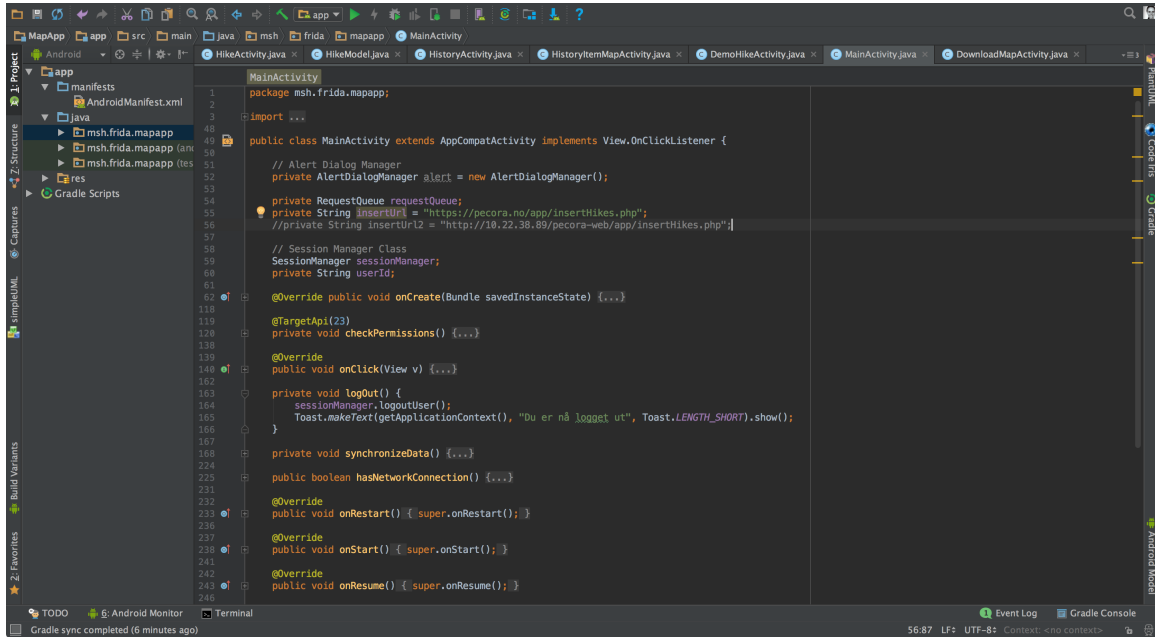


Figure 5.1: Screenshot of the Android Studio IDE.

SQLite SQLite²² is an in-process library that implements a self-contained, serverless, zero-configuration, and transactional SQL database engine [31]. A serverless database was needed for saving the hike data locally as the application may be without WiFi or cellular network connection, for example in the mountains. Since SQLite is a part of the Android database library and it was recommended for saving data locally, it was easy to choose this for the local application database. SQLite is widely used, and there are many tutorials on how to get started with it, which was also beneficial. As of today, Android strongly recommends using Room instead of SQLite [32].

Volley Volley²³ is an HTTP library that makes networking for Android applications easier and, most importantly, faster [33]. When researching how to perform network operations in the application, the Volley library was mentioned on the Android Docs²⁴ website. Additionally, it is used in many good tutorials online. So, the library was used to connect to the Pecora server and database. This enabled registration of new users, login, and synchronization of hike data. To use Volley in the project, a dependency in the app's `build.gradle` file needed to be included (see Listing 5.1).

Listing 5.1: Volley dependency in `build.gradle`.

```
dependencies {
    ...
    compile 'com.android.volley:volley:1.1.0'
```

²²<https://www.sqlite.org/>²³<https://github.com/google/volley>²⁴<https://developer.android.com/training/basics/network-ops/>

}

Sublime Text Sublime Text²⁵ was used for the web development of the project as it is a lightweight editor and due to earlier positive experience with it. Furthermore, all the features of a big IDE were not needed, so Sublime Text was a suitable choice. Sublime Text supports a large variety of language syntaxes, which made it effortless to write in the languages needed for the project, such as HTML²⁶, CSS²⁷, Javascript²⁸ and PHP²⁹. A screenshot of the user interface can be seen in Figure 5.2.

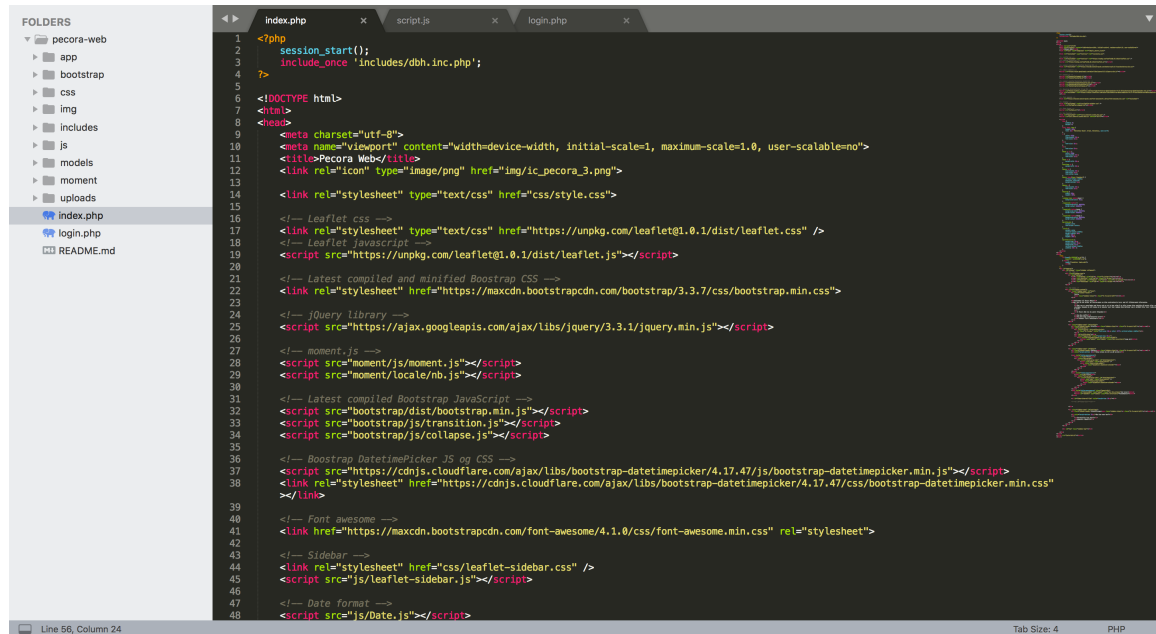


Figure 5.2: Screenshot of the Sublime Text user interface.

HTML & CSS The Pecora website was built using HTML (Hypertext Markup Language 5) and CSS (Cascading Style Sheets). HTML is the coding language that one surrounds the content with, to tell browsers about headers, lists, tables, and so forth. CSS is the language that is used to design the web page, asking the browser to change colors, font, layout, and so on.

JavaScript JavaScript (JS) is a lightweight and dynamic programming language with first-class functions [34]. It is mostly known for the scripting language of web pages, but many non-browser environments also use it, such as node.js³⁰ and Apache CouchDB³¹ [34]. Together with HTML and CSS, JavaScript is the foundation for

²⁵<https://www.sublimetext.com/>

²⁶<https://www.w3.org/html/>

²⁷<https://www.w3.org/Style/CSS/Overview.en.html>

²⁸<https://www.javascript.com/>

²⁹<https://secure.php.net/>

³⁰<https://nodejs.org/en/>

³¹<https://couchdb.apache.org/>

modern web development, and every browser can run JavaScript programs without extensions. Hence, JavaScript was used to develop the Pecora website. As it has many similarities with Java, it was a simple language to use, and solutions to problems were easy to find as it is widely used.

jQuery³², which is a JavaScript library, was used for DOM-traversal (Document Object Model) and manipulation, event handling, and Ajax calls. With the HTML DOM, JavaScript can access and change all the elements of an HTML document. When a web page is loaded, the browser creates the DOM of the page, and then jQuery can access the different objects of the HTML and manipulate it. The event handling was used for, e.g., click events for a button. The Ajax method is beneficial to dynamically update parts of the user interface without having to reload the page, e.g., a request to the server to retrieve some data, and then update the UI with it.

XAMPP A server was needed to test and deploy the web application. XAMPP was used in many tutorials when researching databases with web applications, so XAMPP was downloaded and used. XAMPP³³ is an Apache³⁴ distribution containing MySQL, PHP and Perl³⁵. By running the Apache server (see Figure 5.3), databases can easily be managed by the phpMyAdmin³⁶ application. XAMPP was a useful tool as it was effortless to host a local server for both the mobile application and website.

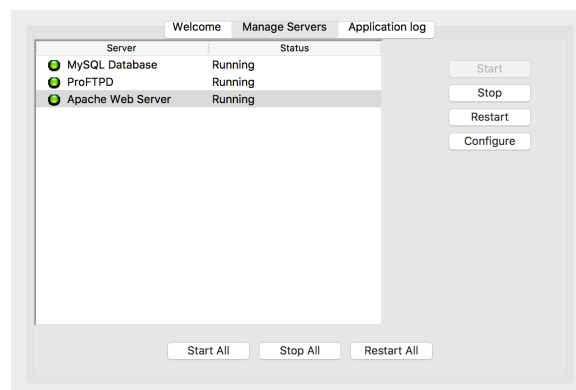


Figure 5.3: Screenshot of the XAMPP program for OS X.

phpMyAdmin phpMyAdmin is a software tool written in PHP, intended to handle the administration of MySQL³⁷ over the web [35]. It supports a wide range of MySQL operations, and these can be performed via the user interface. As phpMyAdmin could easily be used with XAMPP, it was used to manage and create Pecora's databases, such as the *hike* and *user* table. As experience with setting up and managing databases was lacking, phpMyAdmin was a great tool to use as many of the

³²<https://jquery.com/>

³³<https://www.apachefriends.org/>

³⁴<https://www.apache.org/>

³⁵<https://www.perl.org/>

³⁶<https://www.phpmyadmin.net/>

³⁷<https://www.mysql.com/>

MySQL operations could be executed by clicking buttons, such as "Create table". Additionally, there exist many tutorials on how to use the application, which made it even more easy to get started. A screenshot of the user interface can be seen in Figure 5.4.

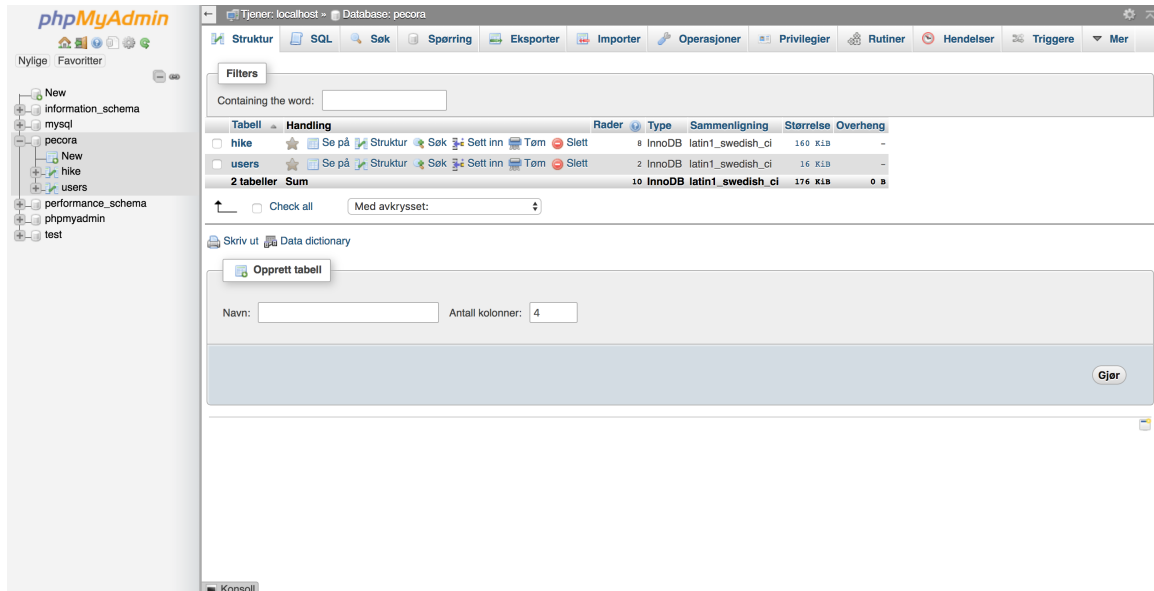


Figure 5.4: Screenshot of the phpMyAdmin user interface.

PHP As XAMPP and phpMyAdmin were used, PHP also became a natural part of the web development. PHP is a widely used open source general-purpose scripting language that is especially suited for web development and can be embedded into HTML [36]. PHP pages can contain HTML with embedded code, and can be put straight in the body by enclosing it with `<?php` and `?>`, which allows one to jump in and out of "PHP mode" [36]. Thus, PHP was used in the web development, but it was mostly used to program the server side of Pecora and to handle user sessions on the website.

JSON The communication between the database server, the mobile, and web application uses the JSON³⁸ format. JSON is a text format that is entirely language independent, but uses conventions that are familiar to programmers of Java and many others. These properties make JSON an ideal data-interchange language [37]. An alternative to JSON could have been XML, which was designed to store and transport data, and plays an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere [38]. However, JSON is a lightweight language to use, and it was easy to convert Java objects to JSON format in the mobile application. Further, it was easy to handle JSON objects with both PHP and JavaScript as well, so XML was not considered a second time.

³⁸<https://www.json.org/>

Postman A RESTful (Representational State Transfer) API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data [39]. Such APIs was made for handling authentication, retrieving and posting hike data in Pecora. Postman³⁹ is an API development environment, and it was diligently used to test the REST APIs created. The Postman application was discovered through a tutorial, and it was easy to use for testing both GET and POST requests. In Figure 5.5, one can see an example of a "register user" test. By adding the fields necessary for registering a user with valid input, it was easy to test what response one would get from the code written, or the server.

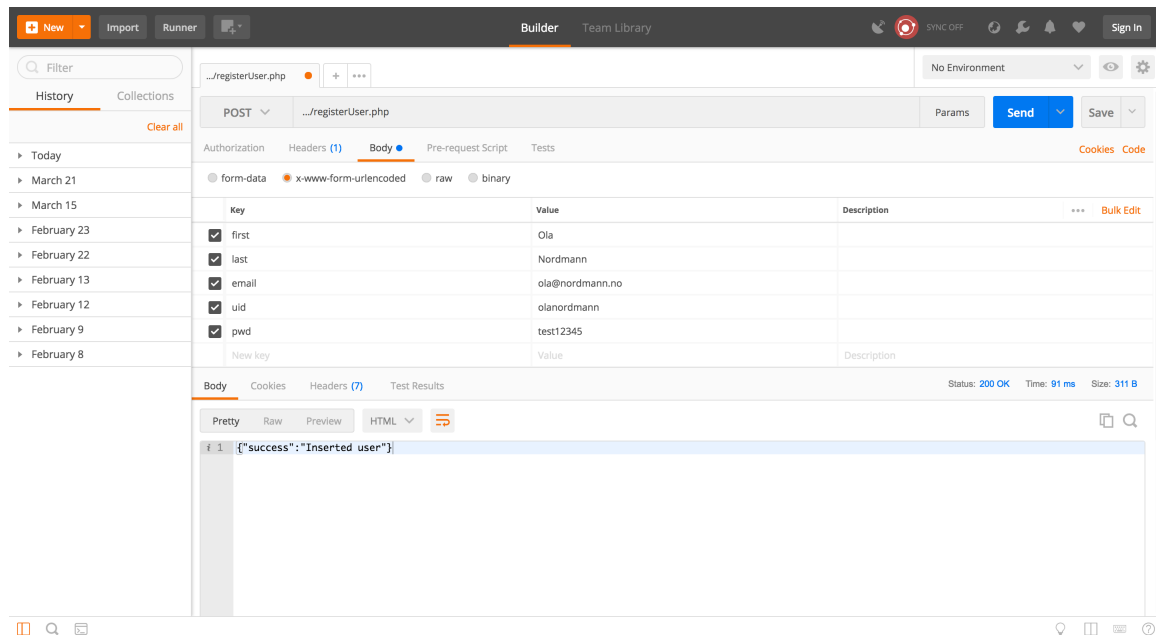


Figure 5.5: Screenshot of the Postman user interface with a POST request.

Trello The Trello⁴⁰ application is a flexible and visual way to organize anything with anyone. Past positive experiences with the tool made it part of this project as well. In this project, it was used to organize things to do, both development and documenting tasks. With Trello one can create different boards, which can contain many lists, which again can contain cards. The cards are typically to-do tasks.

A Pecora board was made (see Figure 5.6) with different "To do" lists, but also state lists, such as "Doing" and "Done". Using Trello made organizing easier, and it was simple to see what needed to be done by putting the most important cards at the top. Labels (see the colors on the cards) could also be used to organize and categorize, such as pink for "Important" and red for "Bug".

³⁹<https://www.getpostman.com/>

⁴⁰<https://trello.com/>

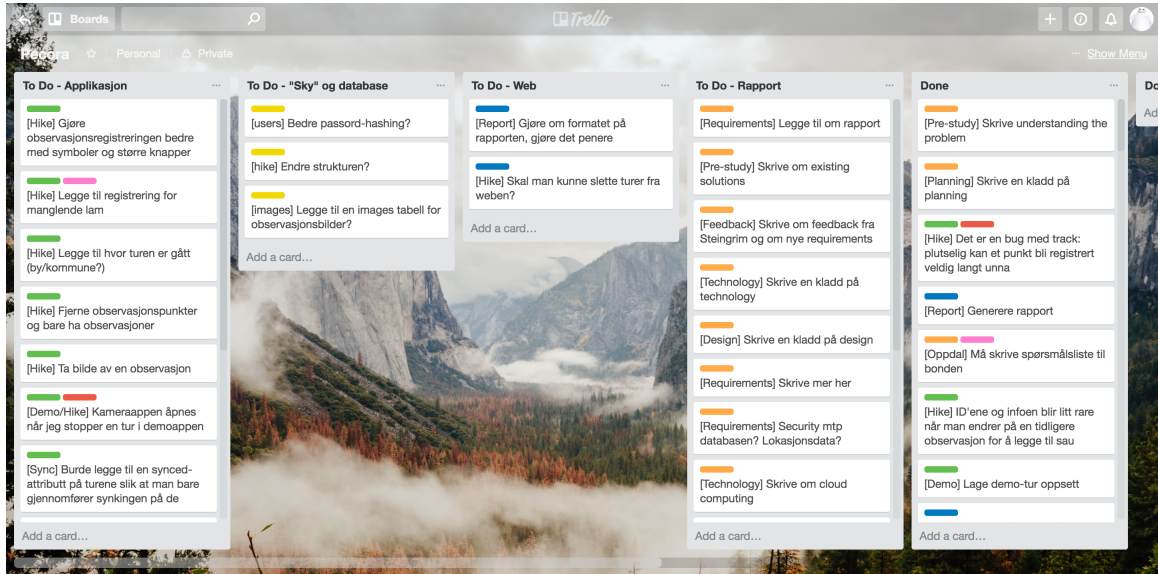


Figure 5.6: Screenshot of the Trello user interface with a board.

5.4 Graphical Design Tool

A graphical design tool was needed to design some of the graphical elements in the Pecora mobile app. XML could be used for designing such elements, but as graphical design tools were familiar, this was easier than spending time on learning how to create elements with XML. The tool used was GIMP⁴¹ as it is a free cross-platform and open source image editor. Some of the icons made with GIMP can be seen in Figure 5.7. It was noticed that some of the images became a bit *pixely* around the edges, but this was considered a small problem as the application was a prototype. The Pecora logo was also designed with GIMP.

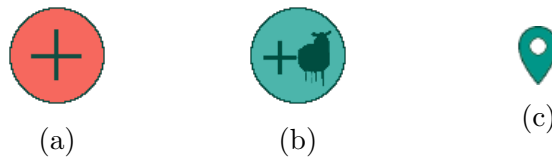


Figure 5.7: Icons made with the GIMP tool.

5.5 Map Data and Frameworks

Map Data The map provider chosen for Pecora was Kartverket⁴², and this decision was made early on in the specialization project as the application is meant for use in Norway. Kartverket offers cache services that support the Google Maps API protocol⁴³.

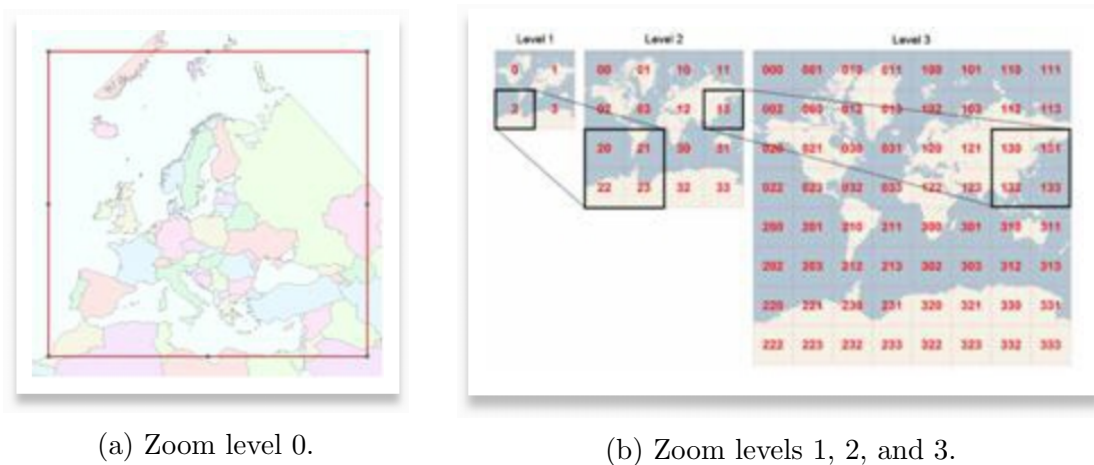
⁴¹<https://www.gimp.org/>

⁴²<https://kartverket.no/>

⁴³<https://www.kartverket.no/data/api-og-wms/>

It was discovered that when being involved with maps and cache services in software development, one has to know about the mapping of map images or so-called *map tiles*. The base for the division of map tiles is the geographical extent of zoom level 0 (see Figure 5.8a). Each tile is further divided into four new tiles, which again are divided into four tiles, and so on, as shown in Figure 5.8b [40].

The zoom levels are used when downloading maps in the mobile application, and the user can choose how many zoom levels to include in the map of a wanted area. The number of tiles to download will vary depending on the number of zoom levels and how big the area is. If the area of the map view is big and the user wants to download it with many zoom levels, it will be many tiles to download. Thus, the information of how many map tiles to download is given to the users before they download a map.



(a) Zoom level 0.

(b) Zoom levels 1, 2, and 3.

Figure 5.8: Map zoom levels [40].

Map Frameworks The main functionality of Pecora is displaying a map with different layers upon it, also called overlays, which shows tracks and markers. So naturally, the map library decision was a critical one.

The map library decision for the mobile application had to be done in the specialization project. Map frameworks as Leaflet⁴⁴, Google Maps API⁴⁵ and Mapbox⁴⁶ were investigated for this purpose. As earlier experience with maps in software development was lacking, it was hard to tell if any of the APIs would meet the desired requirements for the Pecora app. After a lot of confusion and frustration with how the cache service from Kartverket would be combined with a map framework, help was received from another group working on a similar project, who recommended Osmroid⁴⁷.

⁴⁴<https://leafletjs.com/>

⁴⁵<https://developers.google.com/maps/>

⁴⁶<https://www.mapbox.com/>

⁴⁷<https://github.com/osmroid/osmroid>

As for the map library for the website, this decision had to be made at the beginning of this project. Leaflet and Mapbox were once again investigated and tested before the final decision landed on Leaflet, as Skisporet.no uses it. Since Skisporet.no's website contained many of the necessary elements, such as trails, pop-ups, and a menu, it made it a more straightforward decision, as opposed to the app library decision.

The Osmdroid and Leaflet library will be elaborated more in details in the next paragraphs.

Osmdroid Osmdroid is an almost full and free replacement of Android's MapView class, and it also includes a modular tile provider system with support for numerous online and offline tile sources and overlay support with built-in overlays for plotting icons, tracking location, and drawing shapes [41].

This description of the Osmdroid library fitted the functional requirements for the map in the app entirely, as both online and offline tile sources would be needed to cache maps and then to use them offline during a hike. Overlays were also an essential part of drawing hike tracks and placing out icons for observation points and observations.

After choosing Osmdroid, the implementation of the app got much smoother, and the work efficiency increased a lot. The main reason for the faster progress on the application was thanks to that Osmdroid has example projects on their home page and applications on Google's Play Store⁴⁸. The examples were helpful to get started with the map implementation, and they also included examples for features as "my location", cache archival and track recording, which was highly relevant to the Pecora app.

How Osmdroid was used and implemented can be viewed in the specialization report (see Appendix A.1 for report URL).

Leaflet Leaflet is one of the leading open-source JavaScript libraries for interactive maps, and it has most of the map features developers need. It works efficiently through desktop platforms, and it can be extended with useful plugins, in addition to being a well-documented API. As discovered early, Leaflet has many helpful UI and vector layers, as markers, pop-ups, and polylines. Kartverket does also have a tutorial⁴⁹ of setting up Leaflet on a website with its map data, which made it simple to get started with a map on Pecora's website.

Overall, Leaflet was perceived as a helpful map framework, and the learning curve was steep. Leaflet is a widely used framework, so it was easy to find solutions to various issues that appeared along the way. An example was to join polylines and markers into one element, or a so-called *layer group*, so they could be displayed as one hike.

How Leaflet was used and implemented will be explained in Section 7.2.3.

⁴⁸<https://play.google.com/store/apps/details?id=org.osmdroid>

⁴⁹<https://www.kartverket.no/data/lage-kart-pa-nett/>

5.6 Hardware

The target devices for the Pecora app are Android mobile devices that support GPS. Thus, the device range is quite significant. To support this further, Google announced in 2017 that there were 2 billion monthly active devices on Android [42].

Android Studio offers emulators to simulate physical devices, which is useful in many cases of app development. In this case, it was not practical as location and GPS were needed functions to test the app. The GPS location could be simulated on the virtual device, but running and testing the application was more natural with a physical device. Figure 5.9 shows Pecora on an Android emulator.

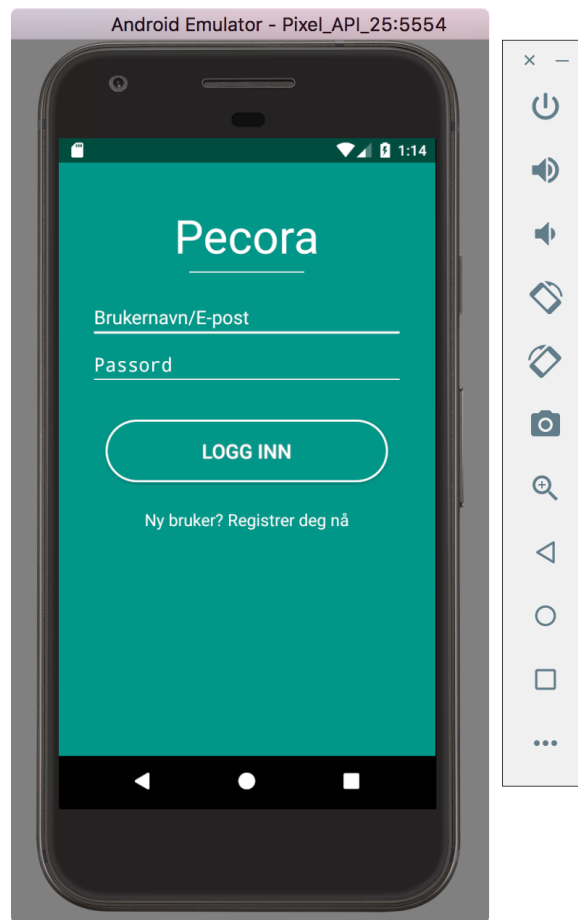


Figure 5.9: Pecora running on an Android emulator.

So, the physical device used to test the app was a Sony Xperia Z5 Compact with Android version 7.0 (Nougat, API 24). Since an older device was used for testing in the specialization project, the minimum version for the Android project was set to Jelly Bean, API 18. Setting the minimum version to a lower version than the newly released version makes the application suitable for a more substantial percentage of the active devices that run Android. This information is given when starting a new project in Android Studio. As of today, if a project targets API 24 and later, the app will run on approximately 8.1% of the devices that are active on the Google Play

Store. Targeting an older API, as 18, will run on about 91.4% of the devices active.

The hardware targets of the Pecora web application are mainly desktop browsers on computers, and not on mobile devices. However, it can be used on mobile phones if desired, but the website will not work optimally on such smaller devices.

Chapter 6 elaborate the design of Pecora, and all the figures in the chapter are screenshots taken with the Sony device or of a desktop browser.

6 User Interface Design

The final user interface design of Pecora will be presented in this chapter. The chapter is divided into two sections; the mobile application and the web application. The sections include appropriate screenshots and descriptions of how the user interface works when the applications are in use.

6.1 Mobile Application

Registration Figure 6.1a shows a screenshot of the mobile application’s registration activity. If the user is new to Pecora, he must register through this form. The user must register with a name, last name, email, username, and password before he can log in. The registration page is accessed by clicking on the label displayed below the login-button when starting the application (see Figure 6.1b).

Login The login screen is displayed in Figure 6.1b. If registered, the user can quickly log in by typing his login credentials and then tap the ”Logg inn”-button.

Menu When the user has logged in, the menu will be displayed. Figure 6.1c is a screenshot of the menu. The menu has four user options; start new supervision hike, download map, see history and synchronize data. Each of the activities mentioned will be explained in more in detail separately. Lastly, a logout-icon is placed in the top-right corner of the menu screen.

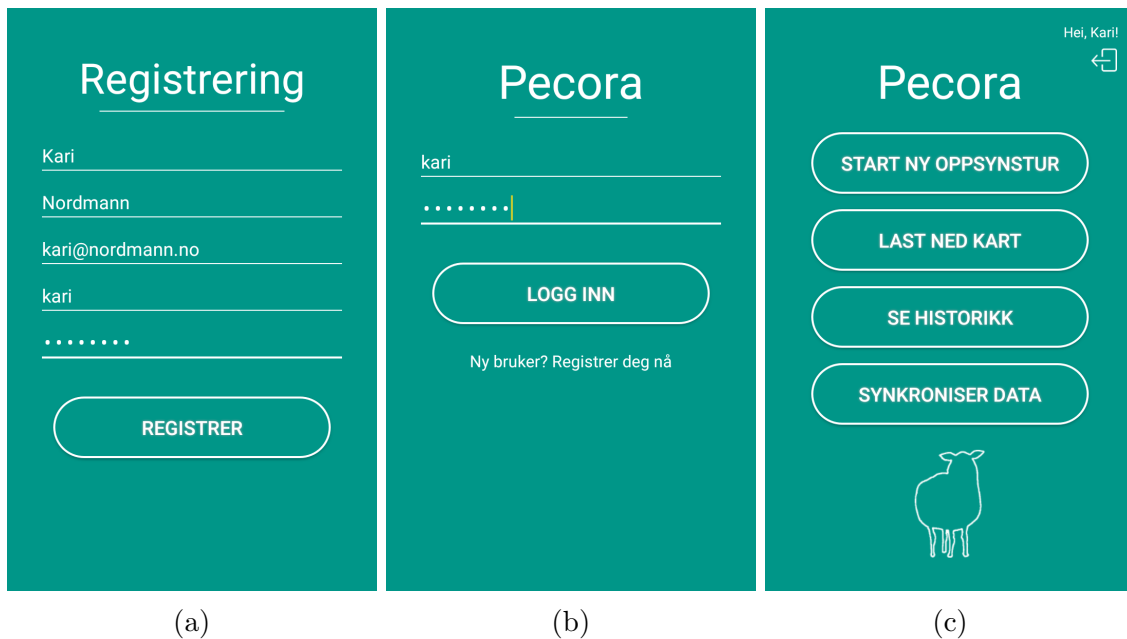


Figure 6.1: App screenshots.

Download Map Before the user can begin a new supervision hike, he must download map data of the area he shall walk in. The application is built for using offline maps due to the reduced cellular network connection in many areas where free-range sheep roam.

When the user chooses to download maps, the screen of Figure 6.2a will be displayed. The screen shows the user's current location by a green arrow. Furthermore, the screen includes two round buttons in the bottom-right corner. The top-button animates the screen to the user's position. The button below is for downloading the map area that is currently showing on the screen. The label in the top-right corner indicates the current zoom level of the map, and this information can be used when downloading a map.

When the user has navigated to the wanted map area, he can tap the download button. A pop-up will appear for choosing an appropriate map name and zoom-levels (see Figure 6.2b). The zoom-levels are selected using the *seek bars*, and a higher zoom-level provides a more detailed map. The label showing the current zoom-level can help the user decide which zoom-levels to choose. A label at the bottom of the pop-up explains how many map tiles the user must download for the selected zoom level range.

When the user is satisfied with the settings, he can press done, and a download progress bar will show (see Figure 6.2c). Information is given to the user if the download completed successfully.

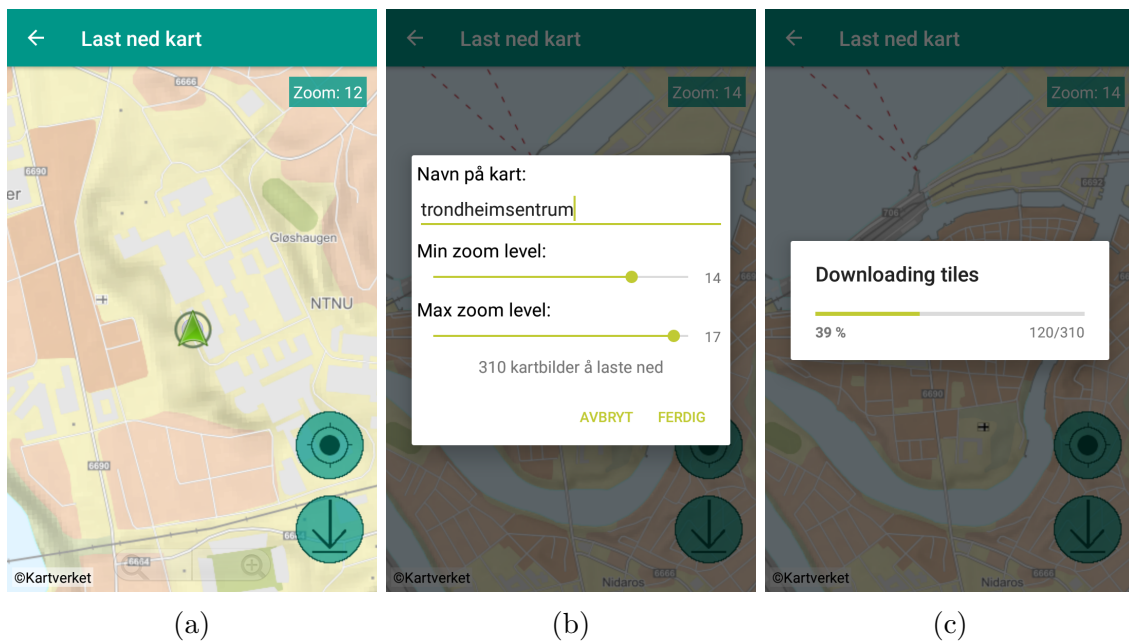


Figure 6.2: App screenshots of downloading an offline map.

Supervision Hike After a map is downloaded, the user can begin a new hike. When the user starts a new hike, he must fill out information about the hike, such as shepherd name, number of participants, weather conditions, description, and lastly,

a downloaded map (see Figure 6.3). The user starts the hike by tapping the "Start turen"-button.

Immediately, the application will start tracking the user's position, which is shown by a blue line. The screen will look like Figure 6.4 when the user is on a hike. The hike screen has three round buttons for "New observation point" (top-right corner), "Stop hike" (bottom-left corner), and "Current location" (bottom-right corner).

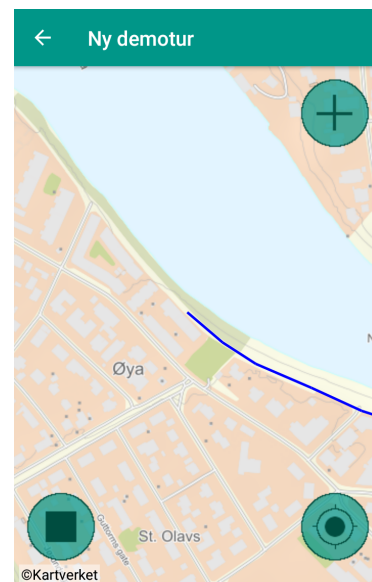
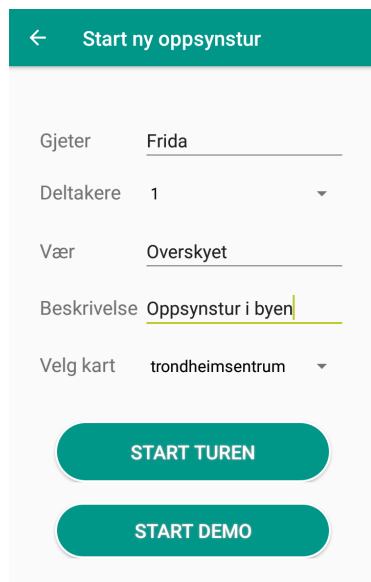


Figure 6.3: App screenshot of a new hike. Figure 6.4: App screenshot of walking a hike.

Figure 6.5 shows the process of registering a new observation point and observation. If the plus-button in Figure 6.4 is tapped, an alert dialog will appear (see Figure 6.5a). If the user wants to register a new observation point, he presses "Ja".

The screen will accordingly change to the screen of Figure 6.5b. A blue circle displays the new observation point, and the screen shows three new buttons. The first is for saving the observation point, the second is for a registering a new observation, and the third is for taking a picture.

Figure 6.5c shows the screen for when the user wants to register a new observation. Firstly, the user must choose a location for the observation. A green marker along with a black cross is centered in the map, and the user can pan and zoom the map to choose a quite exact location. When satisfied, the "Ok"-button can be tapped.

After this, the pop-up layout in Figure 6.6a will appear. The user can choose a category for the observation, and give further details after picking one. In the screenshot provided, the type is *sheep*, and the count is *9*. Further details are their color, which is *white*. When the user is satisfied, the "Ok"-button can be tapped. The new observation will be shown on the map as a yellow circle, with a red line to its belonging observation point (see Figure 6.6b). The observation details can be seen in a pop-up if the yellow circle is tapped.

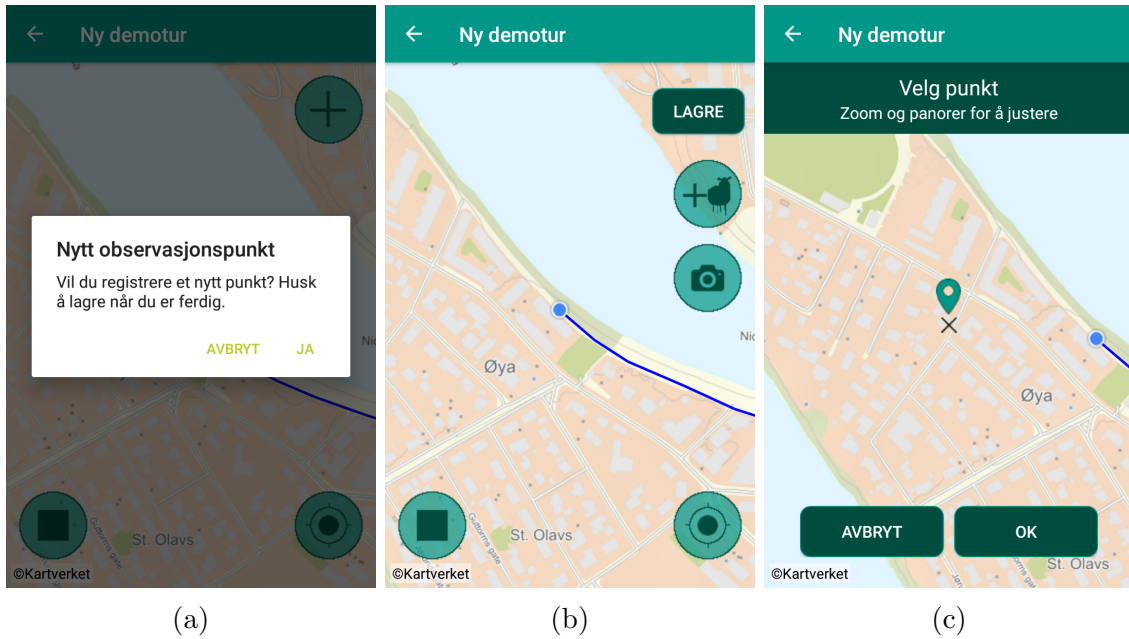


Figure 6.5: App screenshots of new observation process.

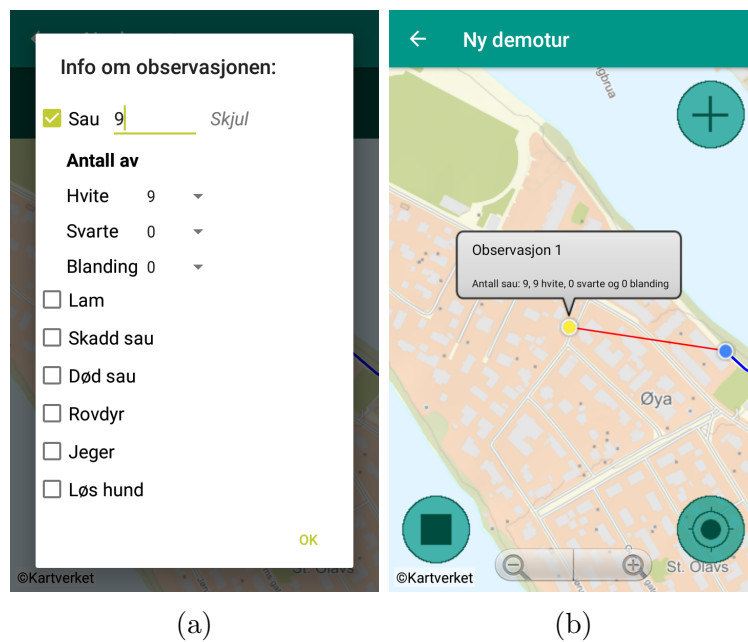


Figure 6.6: App screenshots of registering an observation of nine white sheep.

History The user can easily see the past supervision hikes he has saved. The history activity is shown in the screen of Figure 6.7a. All hikes are displayed in a list with its date, title, start time, and shepherd. A hike can be deleted by a *long-press*, and by clicking "Yes" in an alert dialog that will appear. When a hike is tapped, the screen of Figure 6.7b below illustrates how the application will look. All details are listed, and it is possible to see the map of the hike and further details of the observations.

The last screen, Figure 6.7c, shows the history map. The green circle is the start point of the hike, and the red circle is the endpoint. If the circles are tapped, pop-ups with details are shown.

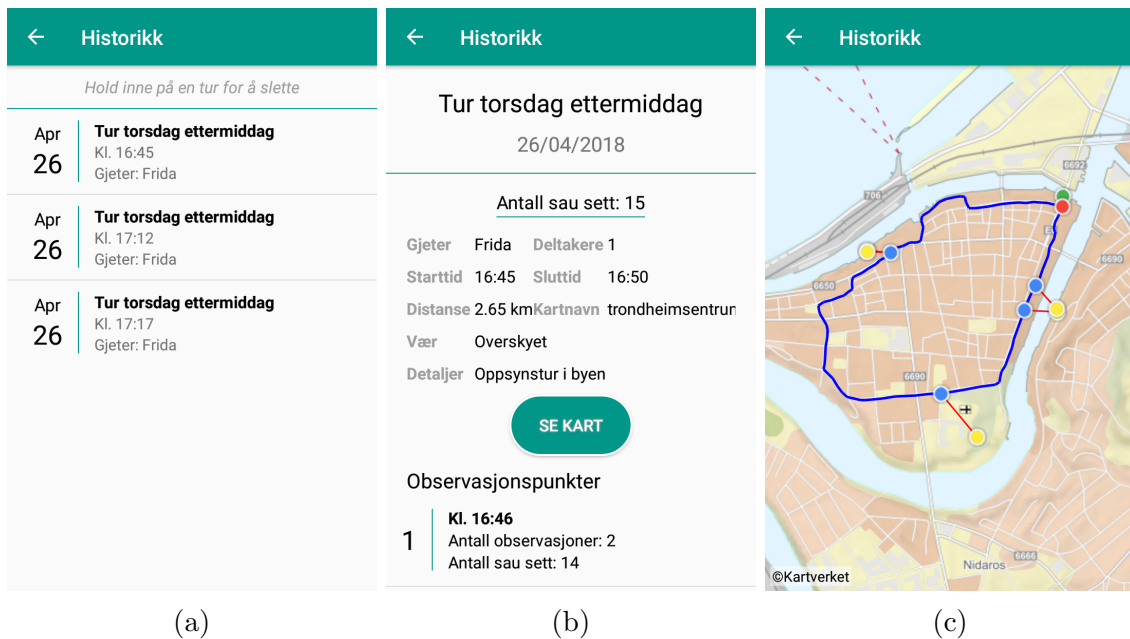


Figure 6.7: App screenshots of hike history.

Synchronizing Hikes For the time being, the user has to tap the "Synkroniser data"-button in the menu (Figure 6.1c) to synchronize with the database. After the button is tapped, the user will get a confirmation if the synchronization was successful or not. It is desirable that this happen automatically when the mobile device detects a network connection in a future version of the app.

6.2 Web Application

Login The web application's login screen is depicted in Figure 6.8. In the current system, the user must be registered through the mobile application with a user account to log in to the web application.

Main Screen When the user has logged in, the application will display a map with the most recent shepherd hike, given the user has any. An expandable menu is positioned at the top-left corner. Zooming controls and a button for showing the user's location is positioned to the right of the menu. The details can be seen Figure 6.9.

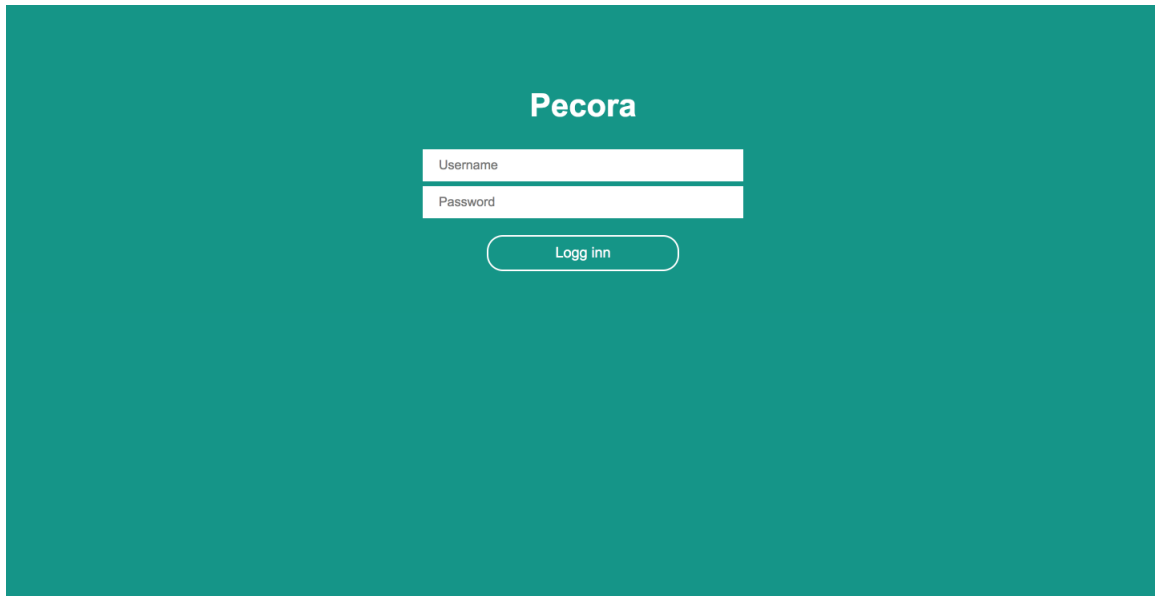


Figure 6.8: Web application screenshot of the login display.



Figure 6.9: Web application screenshot of the main screen.

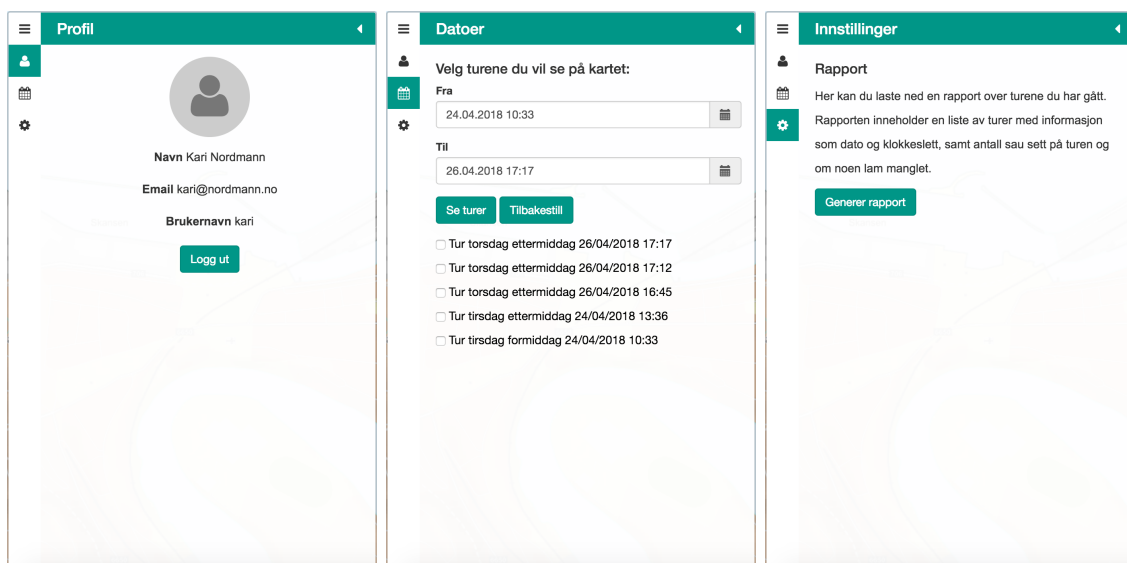
Menu The menu has four different icons that can be clicked; *home*, *profile*, *date*, and *settings*. The home-tab is not essential in the prototype of Pecora, but it was included for illustration purpose. The other tabs, on the other hand, are vital.

The *profile* tab shows user account information along with a log out-button (Figure 6.10a).

Further, the purpose of the *dates* tab is displaying a list of supervision hikes, and for finding hikes within a specific time range. In the tab, two datepickers are positioned at the top where the user can pick a *from*- and *to*-date for a time interval.

Below the datepickers, there are two buttons. The first is for seeing the hikes within the time interval chosen, and the second is for resetting the list located below. At default, the list will display a maximum of the five most recent hikes the user has walked (given the user has five hikes). If the user chooses to see hikes within a picked time interval, this list will change accordingly. To retrieve the default list again, the reset button can be clicked. Each list item has a belonging checkbox, and if it is checked, the hike will be displayed on the map. Several checkboxes can be checked simultaneously, indicating that several hike routes can be displayed simultaneously. See Figure 6.10b for reference.

Lastly, the *settings* tab include the opportunity to download a report of the hike data (see Figure 6.10c). If the "Generer rapport"-button is clicked, a PDF is generated and downloaded to the user's computer.



(a) Profile tab.

(b) Dates tab.

(c) Settings tab.

Figure 6.10: Web application screenshots of the menu tabs.

Hike Details The details of each hike are accessed by clicking on the trail and markers. The information is displayed in pop-up windows, as seen in Figure 6.11. Figure 6.11a shows how the general information regarding a hike looks. The information includes the user input and summary of the hike generated in the mobile application. Further, Figure 6.11b shows an observation point and its details, such as time and the sum of sheep observed. The last two figures are examples of registered observations. The observation in Figure 6.11c is equivalent to the observation of 9 *white sheep* seen in Figure 6.6. The observation in Figure 6.11d is an example of a *dead sheep*-observation along with some further details.

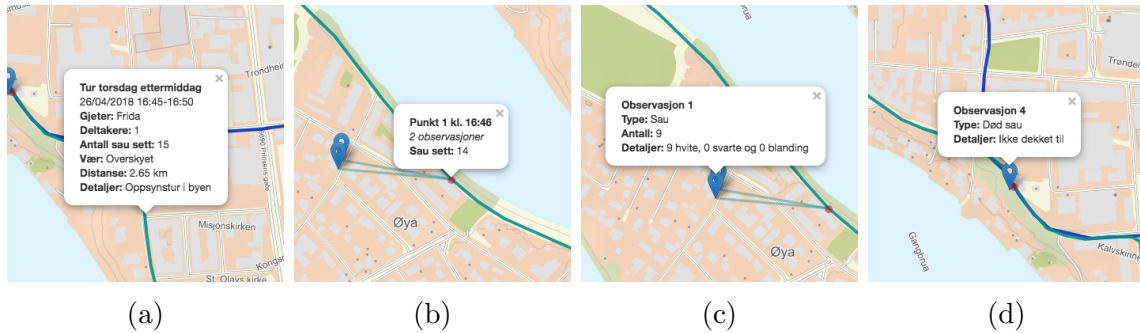


Figure 6.11: Web application screenshots of hike details.

Displaying Several Hikes As mentioned in context with the date menu, several hikes can be displayed simultaneously on the map in the web application. Figure 6.12 illustrates the feature with three hikes displayed. The hike trails are shown with different colors for easy distinction. The color of a hike is randomly picked each time it is displayed.

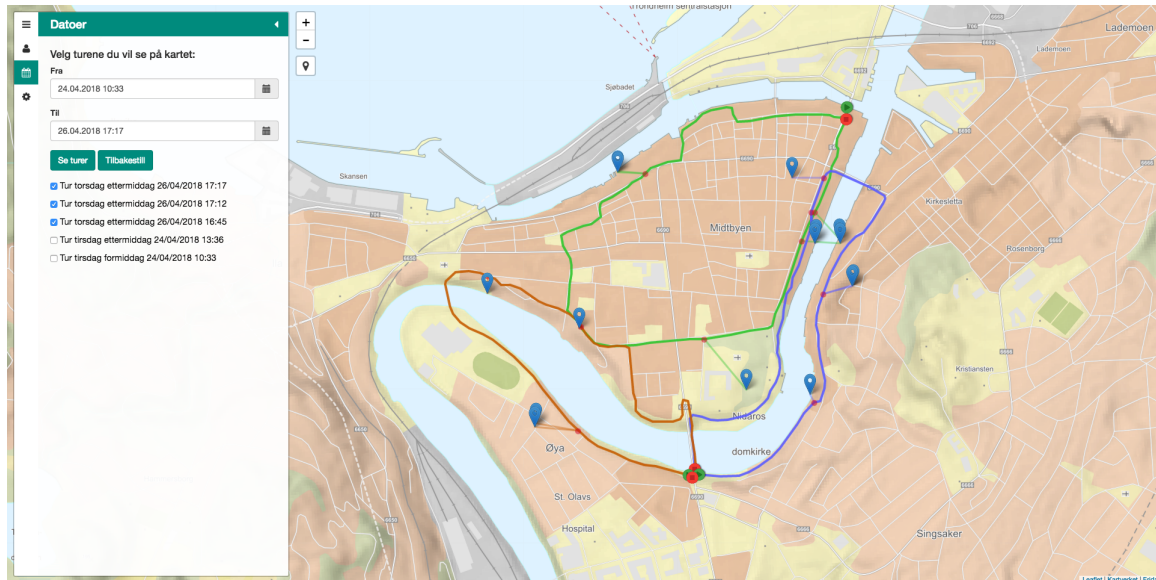


Figure 6.12: Web application screenshot of three hikes on the map.

Time Interval Search Figure 6.13 shows the feature of searching for hikes within a specific time interval. The default list shows five hikes (see Figure 6.12), but after a search between *26.04.2018 12:30 - 26.04.2018 17:20*, the list shows two hikes. The two hikes are displayed on the map.

Downloaded Report An example report of the hike data that can be downloaded from the settings-tab can be seen in Figure 6.14a. The PDF is downloaded with a file name as follows; *report-01-05-2018.pdf*, where the numbers indicate the current date. The first and old version created of the report is displayed in Figure 6.14b.

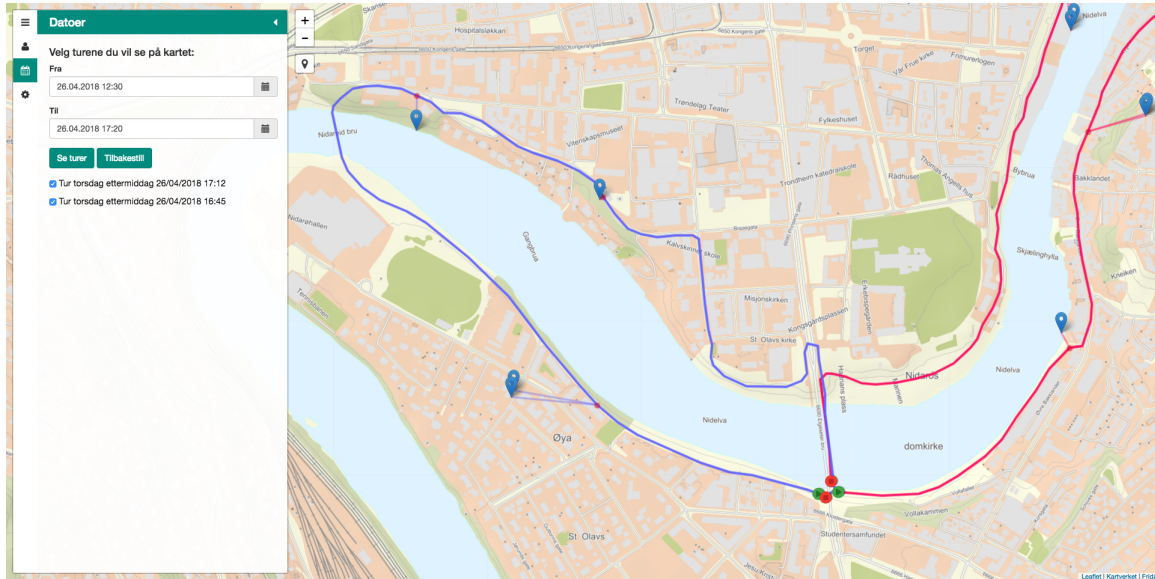




Figure 6.13: Web application screenshot of two hikes within a time interval.

Pecora Generert Rapport		01/05/2018
		
Beitelag: Ukjent		
Beiteår: 2018		
Tilsynsperson: Kari Nordmann		
Dato: 08/04/2018	Beskrivelse: På tur	
Antall sau sett: 26		
Dato: 26/04/2018	Beskrivelse: Oppsynstur i byen	
Antall sau sett: 15		
Dato: 26/04/2018	Beskrivelse: Sentrum	
Antall sau sett: 23		
Dato: 26/04/2018	Beskrivelse: Trondheim sentrum	
Antall sau sett: 29		

Pecora Rapport		01/05/2018
		
Kari Nordmann		
<i>Dato og tid - Antall sau sett - Manglende sau</i>		
08/04/2018 20:05-20:07 - 26 sau - Ingen data		
26/04/2018 16:45-16:50 - 15 sau - Ingen data		
26/04/2018 17:12-17:16 - 23 sau - Ingen data		
26/04/2018 17:17-17:21 - 29 sau - Ingen data		

(a) New version.

(b) Old version.

Figure 6.14: Old and new PDF example of a generated report.

7 System Design and Implementation

This chapter presents the Pecora system design and implementation. Hence, the chapter is divided into two sections. The first section gives a system design overview along with relevant diagrams. The second section provides insight into the implementation of all three parts of the system; the mobile application, the server, and the website.

7.1 System Design

7.1.1 System Overview

An illustration of the Pecora system was created to give a more straightforward overview, and it can be seen in Figure 7.1. The three components of the system are shown within the dashed lines; a mobile device, a server, and a website.

As illustrated, the Pecora app uses the GPS and storage of the mobile device to track inspection hikes and to save data locally. The app does further communicate with the server when the user logs in, registers, or synchronizes hike data. The communication from the server to the app is authentication and feedback regarding hike synchronization.

The server and website communication is a bit different. The server does authenticate users of the website, but it also provides hike data from the database so it can be displayed. The web application communicates with the server when a user wants to log in, and when the user clicks on something, e.g., an earlier hike, that requires data from the database.

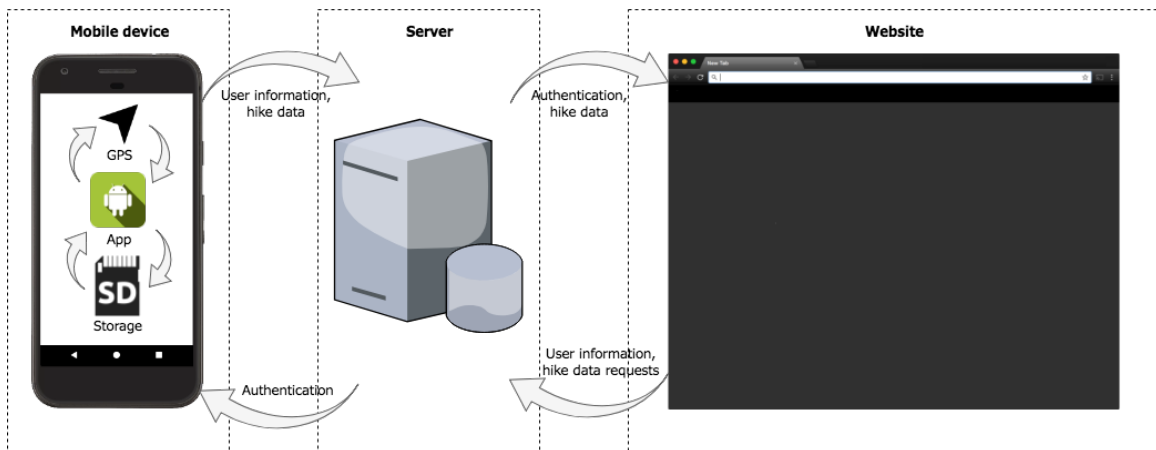


Figure 7.1: System overview diagram.

7.1.2 Use Cases

Use cases are a popular requirements modeling technique as they are both easy to create and understand. Use cases focus on how the user will use the system, and in many ways, they can be better than traditional requirements because they emphasize user-oriented context [43]. The use case technique was not applied directly to find the requirements for Pecora, but it was thought of when writing them. Use cases do also give a better understanding of the system and do clearly show the different actors and actions of the system.

Two use case diagrams were designed for the system; one for the mobile application, and a second for the web application. The user interface and how the applications work when they are in use has already been presented in Chapter 6, but the use case diagrams provide a better overview of the different user actions.

The use case diagram for the mobile application can be seen in Figure 7.2. As displayed, there are three different actors of the application; the user, Kartverket, and the server. Kartverket provides the application with map data, and the server authenticates the user and receives hike data. When the user opens the application, he can either register a new user or sign in. If signed in, the user can begin a new hike, download a map, see hike history, synchronize hike data, or sign out. The further details of the user actions can be seen in the diagram.

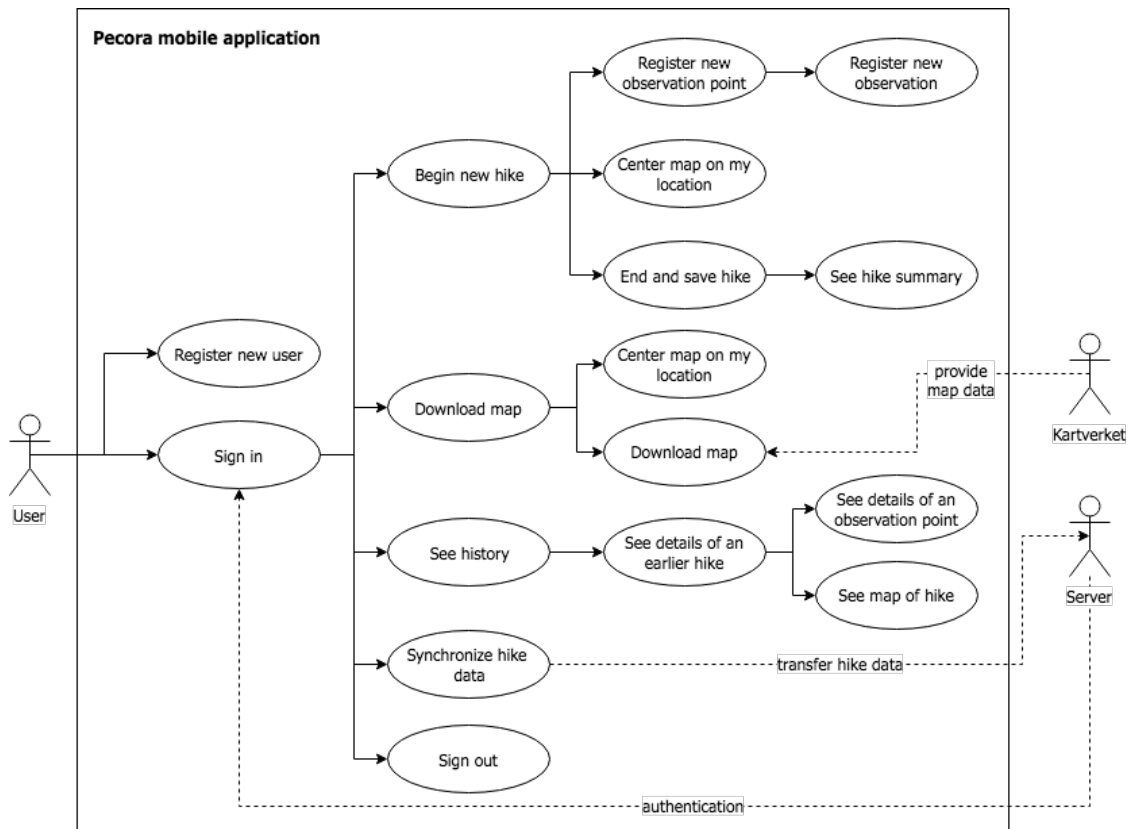


Figure 7.2: Use case diagram for the mobile application.

The use case for the web application can be seen in Figure 7.3 below. The web application has five different actors; the user, Kartverket, the server, the County Governor, and Mattilsynet. The County Governor and Mattilsynet do not provide the application with anything, but they have an interest in the reports the application can generate from the hike data. Kartverket provides the application with map data, and the server authenticates users, and provide hike data. When the web application is opened, the only option for the user is to sign in. When signed in, the user can see the details of the most recent hike on the map, show his location on the map, and click on the different tabs of the menu. More details can be seen in the diagram.

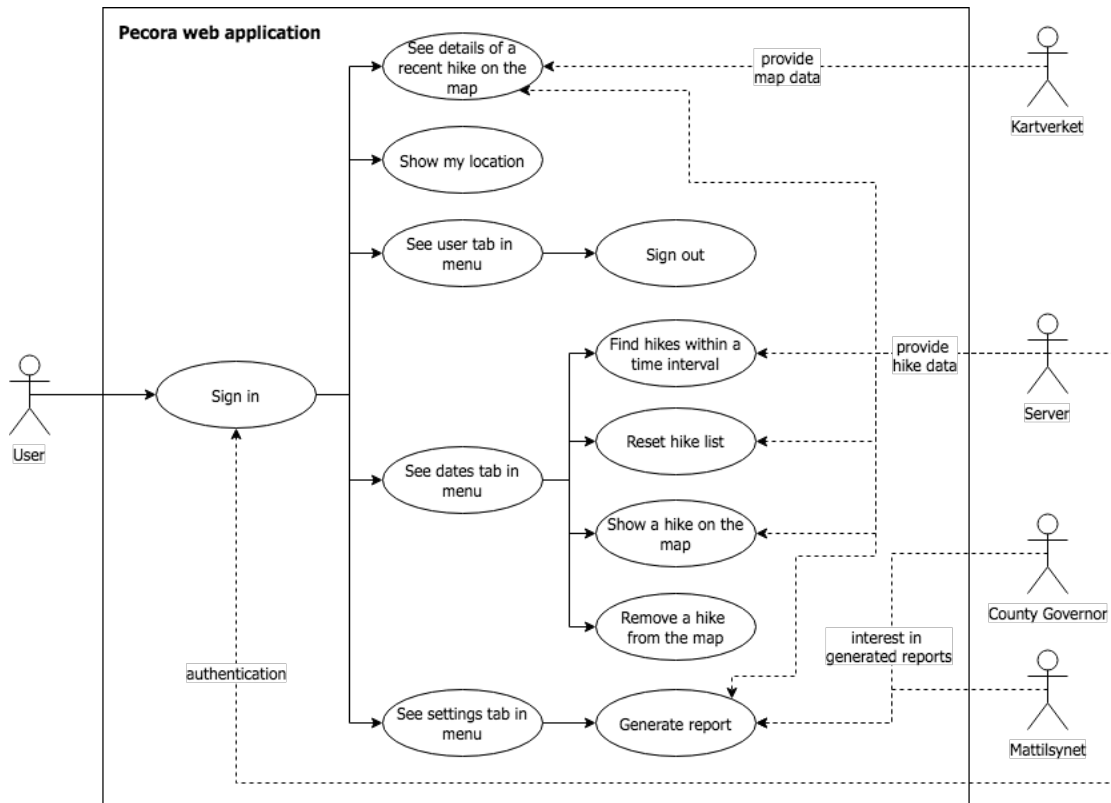


Figure 7.3: Use case diagram for the web application.

7.1.3 Mobile Application Details

A general overview of the system and the use case for the mobile application has been presented, but in this section, an Android Activity⁵⁰ diagram and class diagram introduce further details.

Android Activity Diagram The Android activity diagram was created to show the application flow and structure easily. The application is constructed by Android activities, which are crucial components of an Android app. The activities serve as

⁵⁰<https://developer.android.com/guide/components/activities/intro-activities>

entry points for user interaction as each activity class in this application has a belonging layout, which serves as the user interface. The Android activity flow diagram can be seen in Figure 7.4.

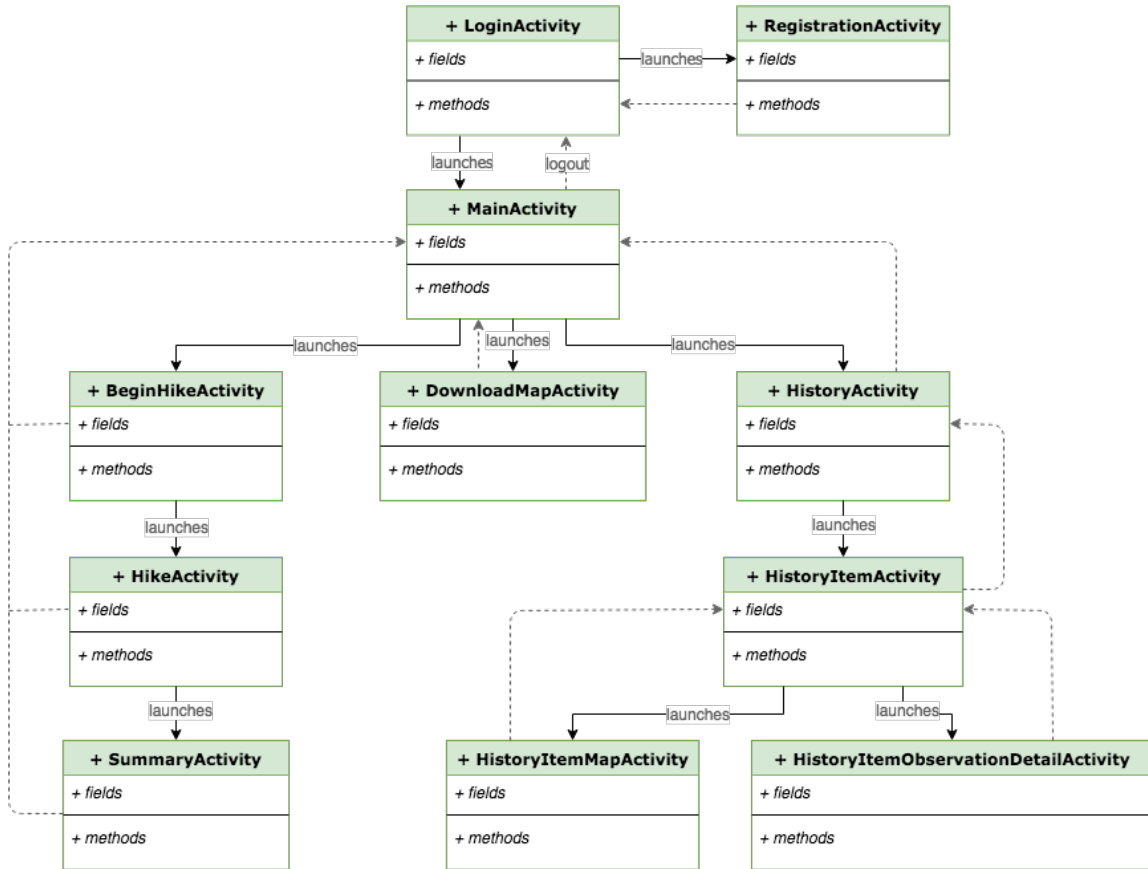


Figure 7.4: Android activity flow diagram.

The diagram shows the Activity hierarchy, where `LoginActivity` is at the top. The `LoginActivity` can launch `MainActivity` or `RegistrationActivity`. The `MainActivity` reflects the menu screen and it can launch either `BeginHikeActivity`, `DownloadMapActivity`, or `HistoryActivity`. The three different activities are further nested with other activity classes to make the application more simple for user input, code structure, and decoupling. This can be seen in the diagram. The black solid lines of the diagram reflect which activities can be launched by which activities. Further, the grey dotted lines show the activities' *parent* relationship, thus, where the "Back"-navigation will lead. For `HistoryActivity`, the back navigation moves up the hierarchy as seen in the diagram. However, `HikeActivity` and `SummaryActivity` will navigate the user back to the menu, as this was the logical navigation for these activities.

Class Diagram All the Android Activities just introduced are classes, so naturally, they were also included in the class diagram. The diagram can be seen in Figure 7.5. The class diagram was created without fields, methods, and constructors as they would have made the diagram too complex and disorganized.

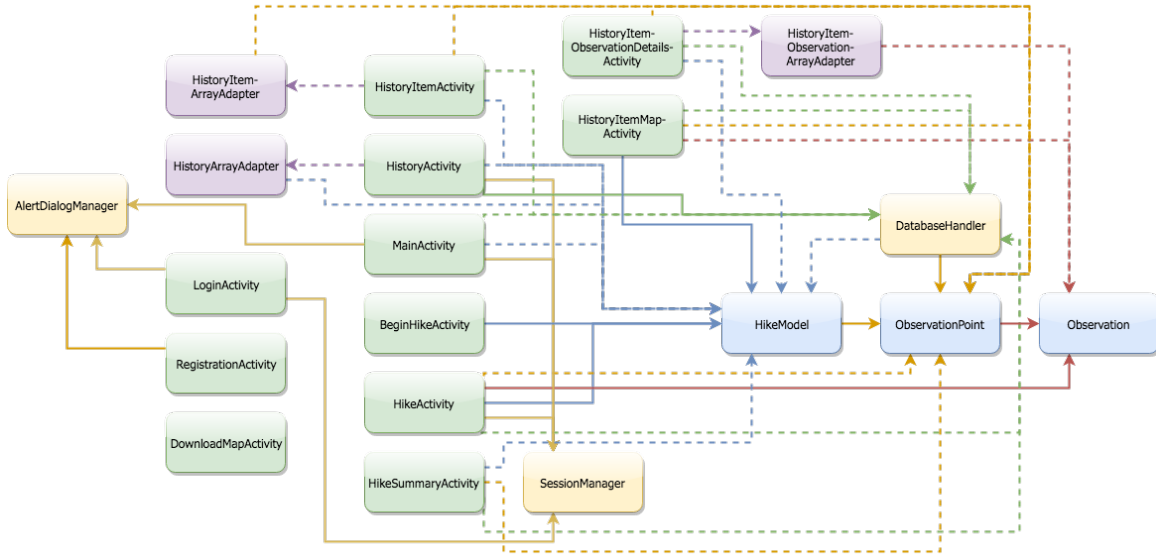


Figure 7.5: Class diagram of the mobile application.

To make the diagram easier to comprehend the classes were assigned different box colors:

- Green boxes: Present Activity classes,
- Purple boxes: Present *ArrayAdapter*⁵¹ classes,
- Yellow boxes: Present *manager* or *handler* classes, and
- Blue boxes: Present *model* classes.

An outgoing arrow from a box is a *reference* to another class, and an incoming arrow is a *usage* from another class. A solid line represent an *association*, and a dotted line is a *dependency*.

The Activity classes were explained in the previous paragraph, and further details regarding references and usages they have can be seen in the class diagram.

The ArrayAdapter classes (purple color) were not necessary classes for Pecora's functionality as an inspection hike application, but they were important for the user interface of the history activities. The classes were used to make the lists of hikes, observation points, and observations look better in the application, resulting in a more user-friendly interface. Figure 7.6 displays an example of the function of HistoryArrayAdapter and its layout, which is used by HistoryActivity.

⁵¹<https://developer.android.com/reference/android/widget/ArrayAdapter>

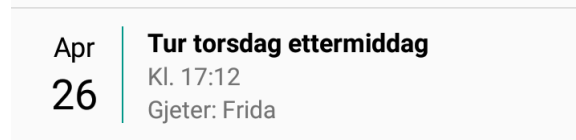


Figure 7.6: Layout populated by the `HistoryArrayAdapter` class.

The yellow boxes represent the manager classes, which include `AlertDialogManager`, `DatabaseHandler`, and `SessionManager`. The alert dialog manager was useful for managing different alert dialogs in the application, e.g., when login fails, or the user has not completed all required fields in the registration form. The database handler class was crucial for storing hikes locally, to recreate hikes in the history activities, and for synchronization towards the global database. Lastly, the session manager class was vital for login sessions in the application. If the user is not signed in, the session manager makes sure the user is directed to the login activity. If signed in, the application shows the main activity immediately and preserve the session until the user logs out.

The last blue boxes represent the model classes for a hike, observation points during a hike, and observations. The `HikeModel` was designed with fields interesting for an inspection hike. The fields can be found in the database design shown in Figure 7.9 in Section 7.1.4. One of the hike fields is an `ObservationPoint`-list, which means the relationship is $1:N$ between a hike model and observation points. The fields of an observation point include a location point, `Observation`-list, id, sheep count, and time. Hence, the relationship between an observation point and observation is $1:N$ as well. Observations have fields such as details, location point, id, sheep count, and type of observation.

See the class diagram for more details on references and usages between the classes.

7.1.4 Web Application and Server Details

The web application system design is displayed in Figure 7.7. The application has a simple structure, as can be seen in the figure. The blue boxes represent the code files written, and the orange boxes represent the libraries and plugins used in the application.

Of the blue boxes there are two HTML pages; the login page `login.php` and the main page `index.php`, which has most of the functionality. The files are coded in HTML, but since some PHP code was required in the files for user session handling, the ending needed to be `.php` and not `.html`. The other two blue boxes are the CSS file `style.css`, which is used by both pages, and the JavaScript file `script.js`, which is only used by the index page.

Screenshots of the folder structure are shown in Figure 7.8 to better illustrate the application construction and the communication with the server.

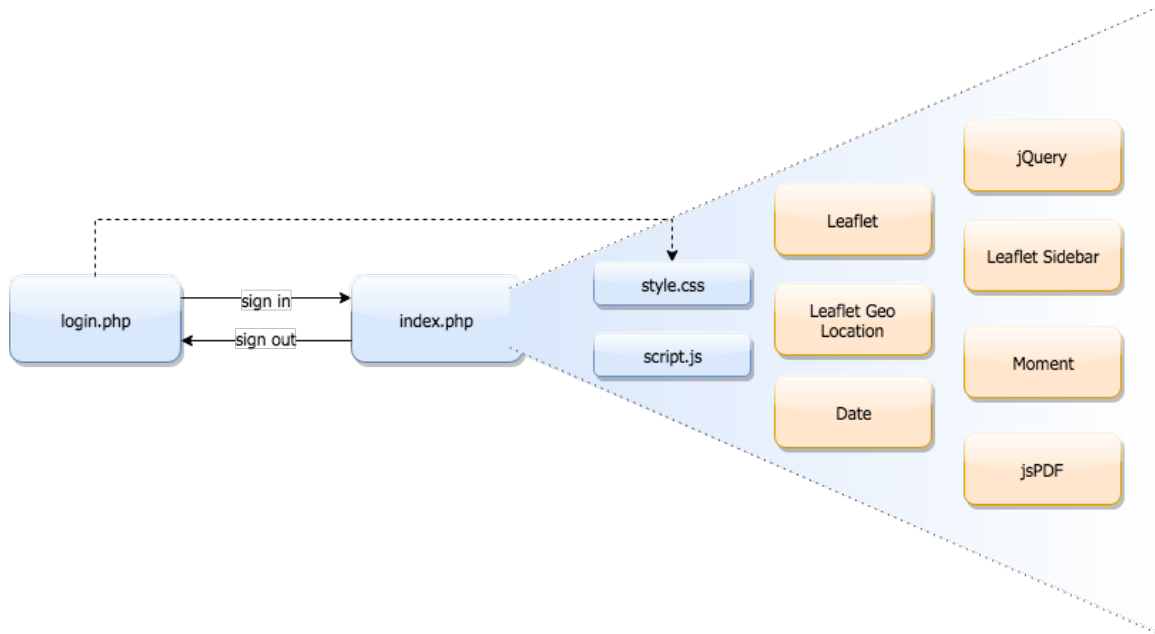


Figure 7.7: Web application illustration.

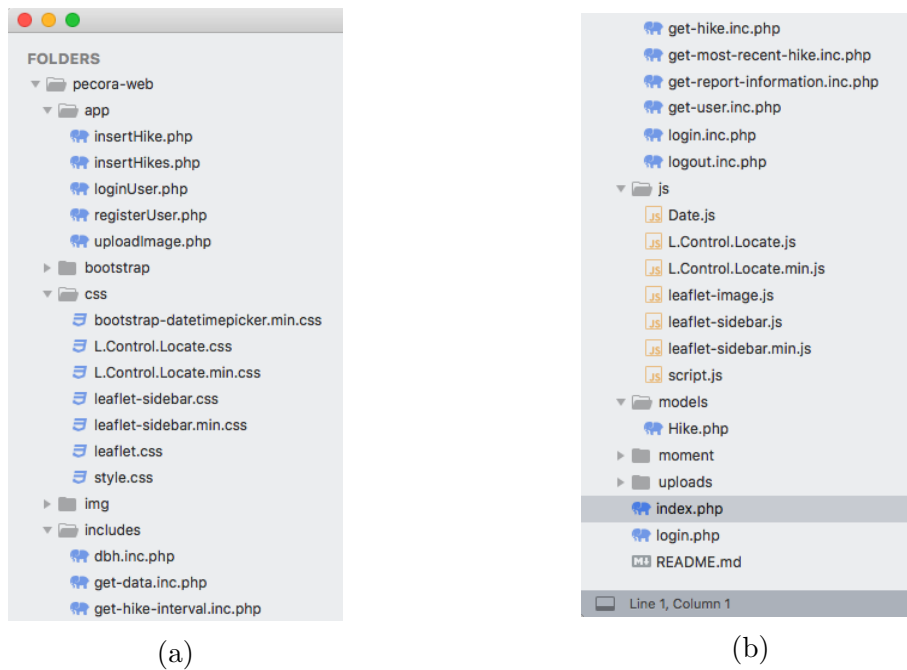


Figure 7.8: Folder structure for the web application and server code.

The most important folders are:

- *app*: Contains the PHP files the mobile application uses for server communication.
- *css*: Contains the style sheet files.

- *img*: Contains the images used in the application, e.g., marker icons.
- *includes*: Contains the PHP files the web application uses for server communication, and the database information file.
- *js*: Contains the JavaScript files.
- *models*: Contains a PHP hike model class to recreate hikes in the web application from the database content.

The index and login PHP files are placed in the *root* project folder. Further implementation details can be found in Section 7.2.

7.1.5 Database Design

The design of the server's database is depicted in Figure 7.9. The Pecora database has two tables, *users* and *hikes*.

The user table holds user information, such as user ID, first name, last name, email, username, and password. The user table is connected with the hikes table through the user ID. The hikes table holds hike information details, such as hike ID, title, name, participants, weather, description, start date, end date, map file, distance, local ID, track points, and observation points.

The database design seems to comply with the first normal form (1NF) at first sight, which is a property of a relation in a relational database [44]. However, the *track* and *observationPoints* fields in the hikes table do not meet the requirement of *atomicity*, as their values can be divided further. The design was created to correspond with the SQLite database in the mobile application, which was designed in the specialization project. Improvements to the design are discussed later in Section 9.2.

How the database content is retrieved in the Pecora system is elaborated in the implementation section.

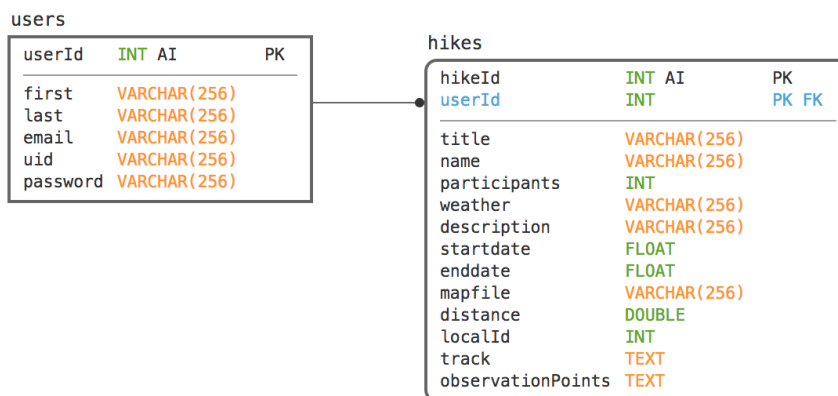


Figure 7.9: The database design in Pecora.

7.2 Implementation

The implementation section is divided into three parts. The three parts elaborate the server setup, the further mobile application implementation, and the web application implementation, respectively.

7.2.1 Server Setup and Communication

Setup In the first period of the thesis, the Amazon Web Services was used for setting up the cloud server mentioned in Section 5.2. This subsection elaborates the setup and usage of the AWS Elastic Compute Cloud 2 (EC2) and Relational Database Server (RDS) and phpMyAdmin.

The following procedure was found in the tutorial by AWS Tutorial Series on Youtube⁵² [45], which lets one connect to an RDS using phpMyAdmin running on an EC2. The first steps included setting up the EC2 server and RDS MySQL server through the AWS console.

The EC2 instance was created with an Ubuntu 14.04 Amazon Machine Image⁵³, a t2.micro instance type⁵⁴, and a security group with all inbound/outbound traffic wide open. Before the instance was launched, a Secure Shell (SSH) key pair was created and downloaded to connect securely with the server.

Before creating the RDS instance, a DB Subnet Group had to be created with a Virtual Private Cloud⁵⁵ (VPC) with all subnets. Further, the RDS instance was created with a MySQL engine, a t1.micro instance type, with the VPC and security group as EC2, and with no backup as it was only a demonstration. The EC2 instance is displayed in Figure 7.10, while the details of the RDS is displayed in Figure 7.11.

After creating the instances, a login by SSH to the EC2 server was needed to update the server and for installing phpMyAdmin. Apache 2 was chosen as the server for phpMyAdmin, and passwords were set for login. After the installation, one could start Apache, and visit the IP address of the EC2 server in a browser and sign in to phpMyAdmin. A localhost database server is default in phpMyAdmin, but to connect with the AWS RDS, the phpMyAdmin `config.inc.php` file needed to be edited. With the right *endpoint address* (see Figure 7.11 below *Connect*), username, and password for the RDS, the instance could be chosen as the database server in phpMyAdmin. Hence, a database could be created on the RDS instance with the tables mentioned in Section 7.1.5. Figure 7.12 shows the Pecora database in phpMyAdmin.

⁵²<https://www.youtube.com/>

⁵³<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AMIs.html>

⁵⁴<https://aws.amazon.com/ec2/instance-types/>

⁵⁵<https://aws.amazon.com/vpc/>

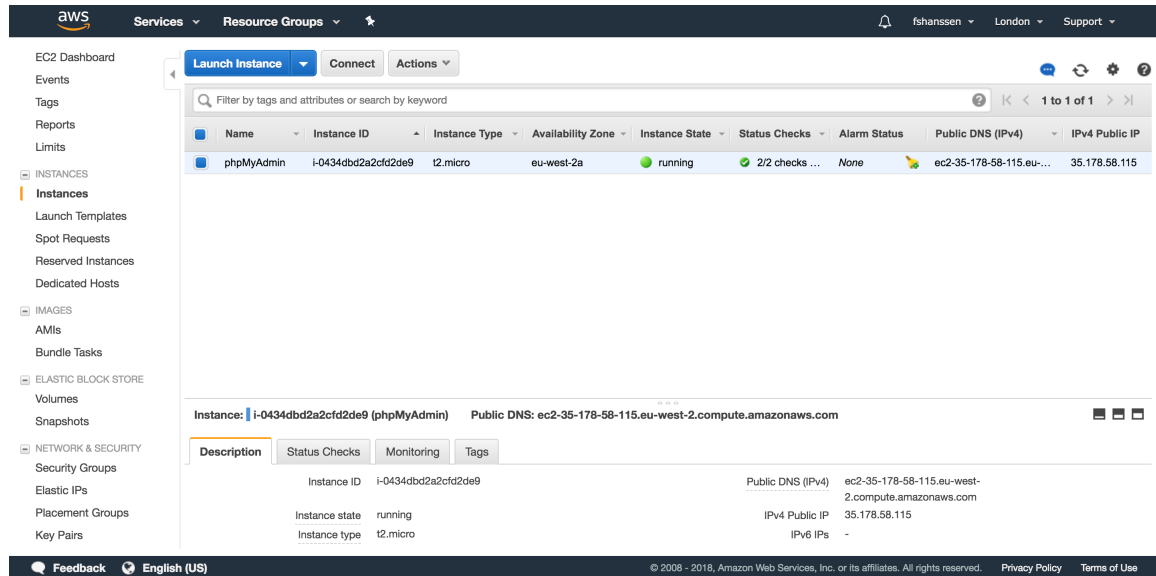


Figure 7.10: The running EC2 instance in the AWS console.

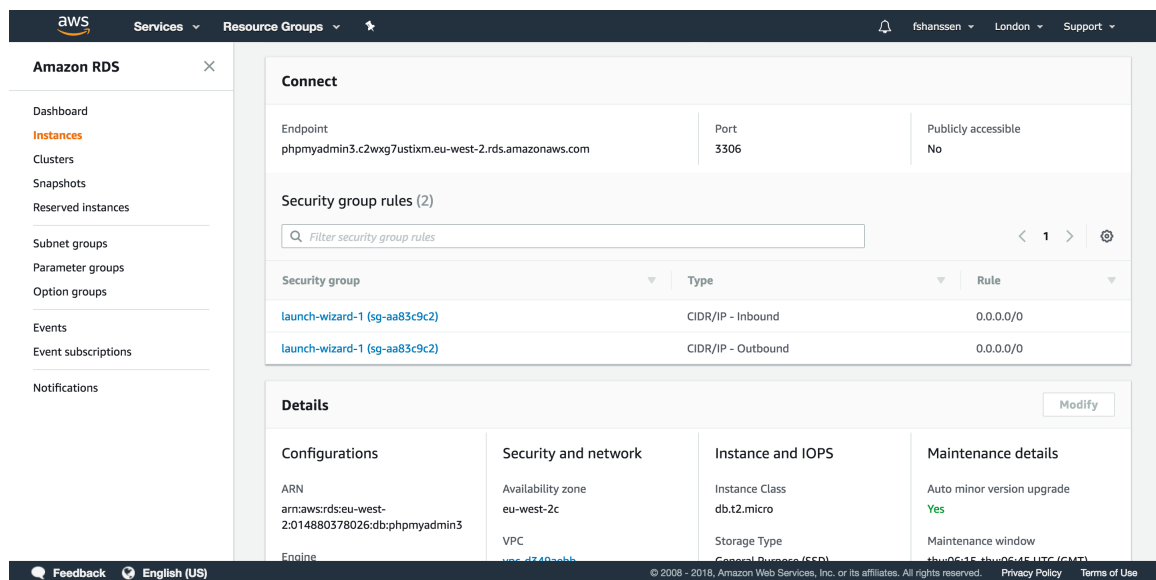


Figure 7.11: Details of the RDS instance in the AWS console.

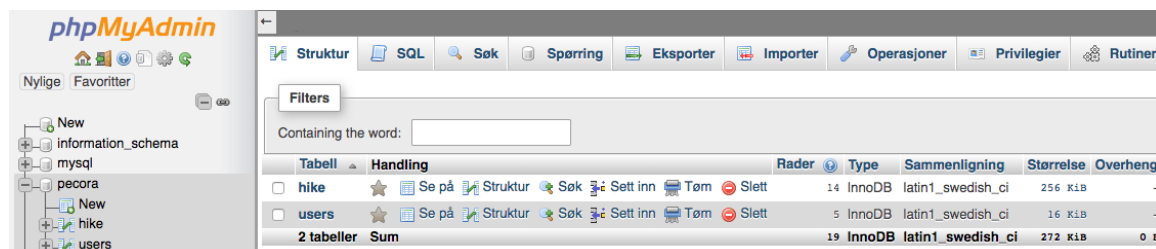


Figure 7.12: Pecora database in phpMyAdmin.

After the AWS free tier period was over, XAMPP was used for local server setup. XAMPP was far less complicated to set up, and fortunately, the communication between the app, website, and database was similar for both solutions which meant no significant changes to the implementation.

To use XAMPP one only had to download the program and then start the Apache web server. To get the website up on the local server the project had to be placed within the `htdocs` folder of XAMPP. `phpMyAdmin` was used the same way with XAMPP as with AWS and were accessed by logging in at `localhost/phpMyAdmin`. Hence, the only changes necessary were the database connection file, and URLs to the server APIs created. The database content already uploaded to the AWS RDS were copied to the local database server before the instance was stopped and deleted.

Communication A database information file had to be created to enable connection to the database server for the mobile and web application. Listings 7.1 and 7.2 displays the database connection file for AWS and XAMPP setup, respectively. All the server APIs were implemented in PHP.

Listing 7.1: The database file `dbh.inc.php` with AWS.

```
<?php
$dbServername = "phpmyadmin3.c2wxg7ustixm.eu-west-2.rds.amazonaws.com";
$dbUsername = "phpMyAdmin3";
$dbPassword = "*****"; //Password in string format and clear text
$dbName = "pecora";
$conn = mysqli_connect($dbServername, $dbUsername, $dbPassword, $dbName);
mysqli_set_charset($conn, 'utf8');
```

Listing 7.2: The database file `dbh.inc.php` with XAMPP.

```
<?php
$dbServername = "localhost";
$dbUsername = "root";
$dbPassword = "*****"; //Password in string format and clear text
$dbName = "pecora";
$conn = mysqli_connect($dbServername, $dbUsername, $dbPassword, $dbName);
mysqli_set_charset($conn, 'utf8');
```

Each of the server APIs, for both the app and the website, has a reference to the database file to enable queries on the database data. The file inclusion can be seen on line 3 in Listing 7.3 below.

Listing 7.3: Server API `get-user.inc.php` for retrieving user information.

```
<?php
session_start();
include 'dbh.inc.php';

$userId = $_SESSION['u_id'];
$sql = 'SELECT * FROM users WHERE user_id=$userId;';
$result = mysqli_query($conn, $sql);
```

```
$resultCheck = mysqli_num_rows($result);

if ($resultCheck == 1) {
    $row = mysqli_fetch_assoc($result);
    //Create array with user information
    $json['name'] = $row['user_first'];
    $json['lastname'] = $row['user_last'];
    $json['email'] = $row['user_email'];
    $json['username'] = $row['user_uid'];
    echo json_encode($json);
} else {
    $json['error'] = 'error';
    echo json_encode($json);
}
```

The listing is also an example of how the server APIs are implemented. The PHP session is used to retrieve the user ID, which is used in all of the MySQL queries to retrieve the correct user and hike information from the database. The queries are written in a string format, as displayed in the code snippet, and all MySQL operations can be used. The query is executed by the PHP MySQL operation `mysqli_query`, which needs the database connection variable and the MySQL string as arguments. To check if the query got any results from the database the function `mysqli_num_rows` is used to count the retrieved rows. In the example below, only one row should be retrieved, so the `if`-statement checks if the number of rows is equal to one. If *true*, the user information is extracted from the row and added to an array. Further, the array is JSON encoded so the server response follows the standard response format that can be easily read by both JavaScript and Java. If the retrieved rows are different from one, an error response is JSON encoded instead. The other APIs follow the same setup, but as they are implemented for different information needs, they are different.

The further sections elaborate how the mobile and web application were implemented, and how the applications use the server APIs to add and retrieve database content.

7.2.2 Mobile Application

The specialization project report contains implementation details of the mobile application such as design, map implementation, map tiles archiving, location implementation, and local database management. A URL link to the report can be found in Appendix A.1. The newly implemented requirements, however, are included in this report.

User Handling User session logic was introduced in the functional requirements (Table 4.3) of the mobile application. To fulfill the requirements FR18-18.3, a `SessionManager` class was implemented, along with a `LoginActivity`, `Registration-Activity`, and Volley POST requests to the server.

The `SessionManager` class allows the user to remain logged in until he wants to log out from the application. Additionally, it makes sure that the login screen will be displayed if the user is not logged in. Listing 7.4 shows a code snippet from the `onCreate()` method in `MainActivity.java`, where the `SessionManager` is initialized and `checkLogin()` is called. The next listing, 7.5, shows the `checkLogin()` method from `SessionManager.java`. The method checks a boolean variable *login*, which is stored in the `SharedPreferences`⁵⁶ of the application. If the boolean is *true*, nothing will happen and `MainActivity` will continue to run. If *false*, the method will interrupt `MainActivity` and start `LoginActivity`.

Listing 7.4: `checkLogin()` called from `onCreate()` in `MainActivity.java`.

```
sessionManager = new SessionManager(getApplicationContext());
sessionManager.checkLogin();
```

Listing 7.5: `checkLogin()` method from `SessionManager.java`.

```
public void checkLogin() {
    if(!this.isLoggedin()){
        Intent i = new Intent(_context, LoginActivity.class);
        i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK|Intent.FLAG_ACTIVITY_NEW_TASK
        );
        _context.startActivity(i);
    }
}
```

In `LoginActivity`, the user can sign in to Pecora. When the sign in button is tapped, a Volley POST request is sent to the server. The request can be seen in Listing 7.6 below. The request is done by adding a `StringRequest`, with POST as method and the URL to the server API, to a `RequestQueue`. The username and password input is posted with the request in a `HashMap`. The `onResponse()` method takes the response from the server and converts it into a `JSONObject`. If the response equals *success*, the user profile information is retrieved and used to create a `SessionManager` login session before starting the `MainActivity`. If the response equals *error*, an alert dialog will show with the relevant error, e.g., "Both username and password must be filled in". The server API for login can be found in Listing B.1 in the appendix.

Listing 7.6: Volley POST request for login.

```
private static final String URL = "https://pecora.no/app/loginUser.php";
private RequestQueue requestQueue = Volley.newRequestQueue(this);
...
final String usernameInput = username.getText().toString();
final String passwordInput = password.getText().toString();
private StringRequest request = new StringRequest(Request.Method.POST, URL, new
    Response.Listener<String>() {
    @Override
    public void onResponse(String response) {
```

⁵⁶<https://developer.android.com/reference/android/content/SharedPreferences>

```

try {
    JSONObject jsonObject = new JSONObject(response);
    if (jsonObject.names().get(0).equals("success")) {
        String id = jsonObject.getString("id");
        ...
        sessionManager.createLoginSession(id, firstname, lastname, email
            , username);
        startActivity(new Intent(getApplicationContext(), MainActivity.
            class));
        finish();
    } else {
        String errorMessage = jsonObject.getString("error");
        alert.showAlertDialog(LoginActivity.this, "Login failed",
            errorMessage, false);
    }
} catch (JSONException e) {
    e.printStackTrace();
}
}
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) { }
}) {
    @Override
    protected Map<String, String> getParams() throws AuthFailureError {
        HashMap<String, String> hashMap = new HashMap<>();
        hashMap.put("username", usernameInput);
        hashMap.put("password", passwordInput);
        return hashMap;
    }
};
requestQueue.add(request);

```

The `RegistrationActivity` is available from the login screen. If the user registers a new user profile, the Volley request sent to the server is almost identical to the login request above. The slight difference is the number of fields that are sent to the server, in addition to receiving more detailed error responses on the different fields. Of course, it does also have a different URL to a different server API. The URL is seen in Listing 7.7 and the API can be found in Listing B.2 in the appendix. If the server response equals *success*, the `LoginActivity` will be started again.

Listing 7.7: URL for registration API on the server.

```
private static final String URL = "https://pecora.no/app/registerUser.php";
```

Hike Synchronization To fulfill the functional requirement FR19, involving synchronization of hikes, an `insertHikes.php` API was implemented along with another Volley POST request. Firstly, the belonging hikes to the logged in user have to be

retrieved from the local database. It was done by comparing the user ID against the hikes'. Secondly, the hikes array needed to be converted to Gson⁵⁷, which is a Java library that can be used to convert Java Objects into their JSON representation in a straightforward manner. After this conversion, the hikes could be sent along with the request to the server. See Listing 7.8 for some details, and otherwise, the request is quite similar to the one in Listing 7.6.

Listing 7.8: Code snippets from MainActivity and its synchronizeData() method.

```
private static final String URL = "https://pecora.no/app/insertHikes.php";
...
Gson gsonHikes = new Gson();
final String hikesString = gsonHikes.toJson(userHikes);
...
@Override
protected Map<String, String> getParams() throws AuthFailureError {
    Map<String, String> parameters = new HashMap<>();
    parameters.put("hikes", hikesString);
    return parameters;
}
```

Take Picture To fulfill the functional requirement of taking a picture of an observation (FR17) a camera activity can be started from HikeActivity. If the user clicks the camera button, the dispatchTakePictureIntent() method fires off and begin the native Android image capture activity. When a picture is captured, the activity finishes and return to the Pecora app and the onActivityResult() method is run. The methods can be found in Listing 7.9 below.

Listing 7.9: Camera events from HikeActivity.java.

```
static final int REQUEST_IMAGE_CAPTURE = 1;

private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
    }
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
        Bundle extras = data.getExtras();
        Bitmap imageBitmap = (Bitmap) extras.get("data");
        String fileName = "img_" + Calendar.getInstance().getTimeInMillis() + ".png";
    }
}
```

⁵⁷<https://github.com/google/gson>

```
        saveImageInExternalCacheDir(getApplicationContext(), imageBitmap,
            fileName);
    }
}
```

The further idea was to save the pictures locally in a Pecora folder and later synchronize the images to the server so they could be shown with the observations in the web application. Unfortunately, synchronization with the server was not implemented, but the images are saved locally in a Pecora folder. The `saveImageInExternalCacheDir()` method compresses the Bitmap argument from `onActivityResult()` before it saves it to a file in the local Pecora folder (see Listing 7.10).

Listing 7.10: `saveImageInExternalCacheDir()` method from `HikeActivity`.

```
public static String saveImageInExternalCacheDir(Context context, Bitmap bitmap
, String fileName) {
    File direct = new File(Environment.getExternalStorageDirectory() + File.
        separator + "Pecora");
    if (!direct.exists()) {
        File directory = new File(Environment.getExternalStorageDirectory().
            getAbsolutePath() + File.separator + "Pecora" + File.separator);
        directory.mkdirs();
    }
    File file = new File(new File(Environment.getExternalStorageDirectory().
        getAbsolutePath() + File.separator + "Pecora" + File.separator),
        fileName);
    if (file.exists()) {
        file.delete();
    }
    try {
        OutputStream fos = new FileOutputStream(file);
        bitmap.compress(Bitmap.CompressFormat.PNG, 100, fos);
        fos.flush();
        fos.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return file.getPath();
}
```

Further details of the mobile application implementation can be viewed in the Github repository (see Appendix B.1 for URL).

7.2.3 Web Application

The functional requirements of the web application were introduced in Table 4.2.

User Sessions When the user opens the web application, the main page `index.php` tries to show. However, the first lines of code in the file is PHP code for user ses-

sion handling. The web application is implemented mostly with HTML, CSS, and JavaScript, but also PHP for user session handling and communication with the server. Listing 7.11 shows the PHP code snippet, which is located above the `html` tags. `session_start()` creates a session or resumes the current one, and the code checks if the user ID has been set in the session. If not, the user will be redirected to `login.php`.

Listing 7.11: PHP session code in `index.php`.

```
<?php
    session_start();
    if (isset($_SESSION['u_id'])) {
        $userId = $_SESSION['u_id'];
    } else {
        header('Location: login.php');
        exit();
    }
?>
<!DOCTYPE html>
<html> .. </html>
```

When the user clicks the sign in button in `login.php`, the username and password input are sent as a POST request through an HTML form with the file reference to the server API; `login.inc.php`. Listing 7.12 displays a code snippet of the HTML form with its two inputs and a button. The full code of `login.inc.php` can be seen in Listing B.3 in Appendix B.

Listing 7.12: Code snippet from `login.php`.

```
<form class="login-form" action="includes/login.inc.php" method="POST">
    <input type="text" name="uid" placeholder="Username">
    <input type="password" name="pwd" placeholder="Password">
    <button type="submit" name="submit">Logg inn</button>
</form>
```

When the user wants to log out, the user session is destroyed, and the user is redirected to the login page. Listing 7.13 shows the little code snippet of `logout.inc.php`, which is run when the logout button is clicked by the user.

Listing 7.13: Code from `logout.inc.php`.

```
<?php
if (isset($_POST['submit'])) {
    session_start();
    session_unset();
    session_destroy();
    header("Location: ../login.php?logout=success");
    exit();
}
```

Map Implementation The map in the web application is implemented by using Leaflet, as mentioned in Chapter 5. Listing 7.14 shows how simply the map is included in a div in the HTML body. At the bottom of the listing, one can see a reference to the JavaScript file `script.js` and Listing 7.15 shows a code snippet from the file. The snippet shows how the map is initialized with the cache service from Kartverket and that `norges_grunnkart` is chosen as the tile layer.

Listing 7.14: Code snippet from `index.php`.

```
<!DOCTYPE html>
<html>
  <body>
    <div id="mapwrap">
      <div id="map" class="sidebar-map"></div>
    </div>
  </body>
  <script src="js/script.js"></script>
</html>
```

Listing 7.15: Map initialization.

```
var mymap = L.map('map').setView([63.416957, 10.402937], 13);
L.tileLayer('http://opencache.statkart.no/gatekeeper/gk/gk.open_gmaps?layers=
norges_grunnkart&zoom={z}&x={x}&y={y}', {
  attribution: '<a href="https://www.kartverket.no/">Kartverket</a>'
}).addTo(mymap);
```

Hike Data Retrieval Before a hike can be shown on the map, the hike data must be retrieved from the database. The data retrieval is implemented by Ajax GET or POST requests to the database server.

The Ajax request for getting the most recent hike when the user logs into the application is shown in Listing 7.16 below. The URL of the server API, type of request, success and error function, and a timeout in milliseconds must be specified in an Ajax request, which can be seen in the listing. The SQL query for the hike in the server API `get-most-recent-hike.inc.php` is shown in Listing 7.17, and the request type is "GET". If the Ajax request is successful and a hike is retrieved from the database, the checkbox of the hike is checked in the hike list (see Figure 6.10b), and the `showHikeOnMap()` method is executed. This method is introduced in the next paragraph.

Listing 7.16: Ajax GET request for "most recent hike" in `script.js`.

```
$.ajax({
  url: "includes/get-most-recent-hike.inc.php",
  type: "GET",
  success: function (data) {
    var hike = JSON.parse(data);
    if (hike.error == 'error') {
      alert("No recent hike to show");
    }
  }
});
```

```

    } else {
        var checkBoxId = '#' + hike.id;
        $(checkBoxId).prop('checked', true);
        showHikeOnMap(hike);
    }
},
error: function(xhr, ajaxOptions, thrownError) {
    alert("AJAX Error - Kunne ikke laste inn turdata");
},
timeout: 15000
});

```

Listing 7.17: SQL query in get-most-recent-hike.inc.php.

```

$sql = "SELECT * FROM hike WHERE userId=$userId AND startdate=(SELECT MAX(
startdate) FROM hike WHERE userId=$userId);";

```

Hike Representation The hikes in the web application are displayed on the map by the method `showHikeOnMap()` in the script file. The full method implementation can be found in Listing B.4 in the appendix as it is an extended method due to JSON parsing of observations and the track of a hike object. However, some parts of the method are presented in this paragraph.

A hike is represented by the the Leaflet layers `Polyline`, `Marker`, `Popup`, and `LayerGroup`. The polylines are *vector layers* and present the hike route coordinates and the links between observation points and observations. The markers and popups are *UI layers* used to mark the observation points and observations on the map with its information if one clicks them. Lastly, the layer group is used for binding the polylines and markers together so that they represent a united hike element.

A superior `LayerGroup` is initialized when the web application is loaded, and further *hike layer groups* can be added to this layer. Such a hike layer is created in the `showHikeOnMap()` method. Firstly, a `mapItems` array is initialized. Secondly, polylines and markers are added to the array. Thirdly, a layer is created based on the map items array and with the hike ID. Lastly, the layer is added to the superior layer group. Listing 7.18 displays a code extraction from the method where the described process can be seen.

Listing 7.18: Code extraction from `showHikeOnMap()` in `script.js`.

```

var layer = L.layerGroup().addTo(mymap);
function showHikeOnMap(hike) {
    var id = hike.id;
    ...
    mapItems = [];
    var polylineColor = createRandomColor();
    ...
    trackPointList = [];
    var jsonTrack = JSON.parse(track);
    for (var i = 0; i < jsonTrack.length; i++) {

```

```

    var latitude = Number(jsonTrack[i].mLatitude);
    var longitude = Number(jsonTrack[i].mLongitude);
    var point = new L.LatLng(latitude, longitude);
    trackPointList.push(point);
  }
  var markerStart = L.marker(trackPointList[0],{icon: startIcon});
  markerStart.bindPopup("<b>Start</b> "+dateStart.format("HH:MM"));
  mapItems.push(markerStart);
  var markerEnd = L.marker(trackPointList[trackPointList.length-1],{icon:
    stopIcon});
  markerEnd.bindPopup("<b>Slutt</b> "+dateEnd.format("HH:MM"));
  mapItems.push(markerEnd);

  var trackPolyline = new L.Polyline(trackPointList, {
    color: polylineColor,
    weight: 4,
    opacity: 0.9,
    smoothFactor: 1
  }).bindPopup('<b>'+title+'</b><br>'+dateStart.format("dd/mm/yyyy HH:MM")+
    ...);
  mapItems.push(trackPolyline);
  mymap.fitBounds(trackPolyline.getBounds());

  var layer2 = L.layerGroup(mapItems);
  layer2.id = id;
  layer.addLayer(layer2);
}

```

The listing does also show how a Polyline for the hike route is created and two Marker objects for the start and end point. The track point list must first be parsed from its JSON format before it can be looped through to create a list with Leaflet LatLng points, which present latitude and longitude points. The points for start and end point can then be retrieved from the list, and the markers can be created. The track polyline is created of the list, and properties as *color*, *weight*, *opacity*, and *smooth factor* can be set. The color of the polyline is decided by a `createRandomColor()` method so the hikes are presented with different colors on the map for easy distinction, as earlier mentioned in Chapter 6. A list with carefully selected colors in a *RGB* format were created, and the `Math.random()` function was used to return a random color from the list.

Hikes Checkbox List How the checkbox list with hikes from the dates tab in Figure 6.10b is implemented is explained in this paragraph. When the website is loaded, the five most recent hikes are loaded from the database and populate the list with their titles and start dates (see Listing 7.19).

Listing 7.19: Code extraction from `getMostRecentHikes()` in `script.js`.

```
var newHTML = '<li><input type="checkbox" id="'+id+'"/> '+title+' '+dateStart.  
    format("dd/mm/yyyy HH:MM")+ '</li>';  
$("#hikes-interval-list").append(newHTML);
```

The checkbox added to the list gets an ID identical to the hike ID so that it is simple to display the correct hike on the map when the user checks it. Listing 7.20 shows the checkbox listener function of the hikes list.

If a checkbox is checked, an Ajax POST request is sent to the database server with the hike ID. The `showHikeOnMap()` method is executed if a hike was retrieved. If a checkbox is unchecked, the `removeHikeFromMap()` method is executed. The method loops through the superior layer and removes the hike layer that matches with the hike ID. The method is displayed in Listing 7.21.

Listing 7.20: Checkbox listener on hike list in `script.js`.

```
$('#hikes-interval-list').on('change', 'input[type="checkbox"]', function() {  
    var hikeId = $(this).attr('id'); //Get the id of the checkbox and hike id  
    if (this.checked) { //If checked; retrieve the hike from database  
        var json = {'hikeId': hikeId}; //Create POST data  
        $.ajax({  
            url: 'includes/get-hike.inc.php',  
            type: 'POST',  
            data: json,  
            success: function (data) {  
                var hike = JSON.parse(data);  
                if (hike.error == 'error') {  
                    alert("Ingen tur aa vise");  
                } else { //Show the checked hike on the map  
                    showHikeOnMap(hike);  
                }  
            },  
            error: function(xhr, ajaxOptions, thrownError){  
                alert('AJAX Error');  
            },  
            timeout: 15000  
        });  
    } else { //If unchecked; remove the hike  
        removeHikeFromMap(hikeId);  
    }  
});
```

Listing 7.21: The `removeHikeFromMap()` method in `script.js`.

```
function removeHikeFromMap(hikeId) {  
    layer.eachLayer(function (layerInGroup) {  
        if (layerInGroup.id === hikeId) {  
            layer.removeLayer(layerInGroup);  
        } }); }  
}
```

Time Interval The user has the option to retrieve hikes within a time interval selected in the dates-tab. If the hike interval button is clicked, the code in Listing 7.22 is executed. Firstly, the hikes list is emptied, and the hikes displayed on the map (if any) are removed. Secondly, the dates from the two date pickers are retrieved and put into a JSON representation. Thirdly, an Ajax POST request with the JSON data is executed. Lastly, if the retrieval is successful, the retrieved hikes within the interval is displayed in the list of hikes.

Listing 7.22: Click listener for hike interval button in `script.js`.

```

$('#intervalBtn').on('click', function() {
    $("#hikes-interval-list").html(''); //Reset the list
    removeHikesFromMap(); //Remove hike layers from the map
    var from = new Date($('#datetimepicker1').data('DateTimePicker').date()).
        getTime();
    var to = new Date($('#datetimepicker2').data('DateTimePicker').date()).
        getTime();
    var json = {'from': from, 'to': to}; //Create POST data
    $.ajax({
        url: 'includes/get-hike-interval.inc.php',
        type: 'POST',
        data: json,
        success: function (data) {
            var obj = JSON.parse(data);
            if (obj.error == 'error') {
                alert('Ingen turer aa vise i dette intervallet');
            } else {
                for (var i = 0; i < obj.length; i++) {
                    var id = obj[i].id;
                    var title = obj[i].title;
                    var dateStart = new Date(Number(obj[i].startdate));
                    var newHTML = '<li><input type="checkbox" id="'+id+'"/> '+
                        title+' '+dateStart.format("dd/mm/yyyy HH:MM")+</li>';
                    $("#hikes-interval-list").append(newHTML);
                }
            }
        },
        error: function(xhr, ajaxOptions, thrownError){
            alert('AJAX Error');
        },
        timeout: 15000
    });
});

```

Report Generation The jsPDF⁵⁸ JavaScript plugin was used to create PDF reports based on the hike data in the web application (see Figure 6.14). The code in

⁵⁸<https://github.com/MrRio/jsPDF>

Listing 7.23 is an extraction from the code that creates and saves a jsPDF with a specified file name. The wanted text is easily added to the document, but the placement of the text must be specified. Text style and font size can also be edited, along with some other features the jsPDF plugin supports. The full code for jsPDF creation when the report button is clicked can be found in Listing B.5 in the appendix.

Listing 7.23: Extraction from PDF creation code in `script.js`.

```
var doc = new jsPDF();
pageHeight = doc.internal.pageSize.height;
doc.setFontSize(12);
var dateNow = new Date();
var dateNowFormatted = dateNow.format("dd/mm/yyyy");
doc.text(195, 15, dateNowFormatted, null, null, 'right');
...
var dateNowFormattedDash = dateNow.format("dd-mm-yyyy");
var pdfName = 'report-'+dateNowFormattedDash+'.pdf';
doc.save(pdfName);
```

Sidebar Menu One of the functional requirements of the website was to have a menu that is expandable so it would not occupy the map at all times. Leaflet has many third-party plugins and a `sidebar-v2`⁵⁹ was used to implement the menu in Pecora web. The code in Listing 7.24 shows the one-line code for initializing the sidebar in JavaScript, and Listing 7.25 displays the HTML code for the sidebar. The sidebar has a parent `div`, which contains a sidebar tabs `div` and sidebar content `div`. The tabs `div` includes the links to the home, profile, dates, and settings tab along with their icons, and the content `div` includes the content of each of the tabs. The content of the tabs is not included in the listing below, but it can be viewed in the project repository on Github (see Appendix B.1 for URL).

Listing 7.24: Initialization of sidebar in `script.js`.

```
var sidebar = L.control.sidebar('sidebar').addTo(mymap);
```

Listing 7.25: Sidebar HTML in `index.php`.

```
<div id="sidebar" class="sidebar collapsed">
  <div class="sidebar-tabs">
    <ul role="tablist">
      <li><a href="#home" role="tab"><i class="fa fa-bars"></i></a></li>
      <li><a href="#profile" role="tab"><i class="fa fa-user"></i></a></li>
      <li><a id="dateLink" href="#dates" role="tab"><i class="fa fa-
        calendar"></i></a></li>
      <li><a href="#settings" role="tab"><i class="fa fa-gear"></i></a></
        li>
    </ul>
```

⁵⁹<https://github.com/turbo87/sidebar-v2/>

```

</div>
<div class="sidebar-content">
  <div class="sidebar-pane" id="home">
    <h1 class="sidebar-header">Pecora
      <span class="sidebar-close"><i class="fa fa-caret-left"></i></span>
    </h1> ...
  </div>
  <div class="sidebar-pane" id="profile">...</div>
  <div class="sidebar-pane" id="dates">...</div>
  <div class="sidebar-pane" id="settings">...</div>
</div>
</div>

```

Geolocation Functional requirement 9 of the web application said to have a button for showing the user's current location. The requirement has a low priority, but it was easily implemented as it existed a Leaflet plugin for this as well. The Leaflet Locate control⁶⁰ was used, and Listing 7.26 shows how the control was added to the Leaflet map. Figure 7.13 shows how the current location is shown on the map in the application.

Listing 7.26: Geo location code.

```
L.control.locate().addTo(mymap);
```



Figure 7.13: Web application screenshot of how Leaflet Locate control functions.

Further details can be found in the Github repositories (Appendix B.1).

⁶⁰<https://github.com/domoritz/leaflet-locatecontrol>

8 Results and Analysis

The analysis chapter will describe, explain and evaluate the findings from conducting interviews with relevant users of the Pecora system and observations from user testing. The first section covers the interviews with a sheep farmer and the County Governor of Trøndelag and their feedback to the Pecora prototype. The second section covers the user testing and results.

8.1 Interviews

The interview data generation method was mentioned in Chapter 3. This method gives room for detailed information and feedback from users of an information system. Two relevant interviews were conducted through the thesis project for information collection and for demonstrating the prototype applications. The next paragraphs elaborate the interviews separately in more detail.

8.1.1 Interview 1: Steingrim Horvli

The first interview was conducted with the sheep farmer Steingrim Horvli. It took place in Oppdal April 9, 2018. The interview was conducted together with another group working on a similar thesis project, and a joint questionnaire was produced in advance of the meeting. The questions were made to try to understand better how sheep herding is conducted today and to see if our system would be useful for the sheep farmers.

The meeting was split in two where the questions were asked first before a demonstration of the systems was presented to Horvli. The interview gave very relevant and useful background information about sheep herding, and it gave a better understanding of the problem at hand. Further, the demonstrations allowed Horvli to provide relevant feedback on what he liked about the prototype, and what should be improved or added before he would have liked to test it in real operations.

The key findings from the meeting included that the most critical and common to register during the supervision hikes are missing lamb. Ewe's have ties attached to their collar that indicate how many lamb they have. These facts were presented in Section 2.1, but were discovered through this interview with Horvli. The sheep farmer's profits are mainly based on the number of lamb he sells after the grazing season, as one gets 1000-1500 NOK per lamb, as opposed to only 10-20 NOK per ewe. Thus, it is understandable that observations of lamb are economically essential. It is also more common to observe missing lamb than missing grown sheep as lamb are smaller and therefore more vulnerable to predators than grown sheep.

Additionally, there were other aspects of sheep herding that were unknown. The aspects included:

- Organized supervision is conducted between farmers in the same area.

- The farmers use GPS tracking of the sheep, but they still need to conduct manual supervision.
- The farmers report collectively as a *beitelag* to the County Governor, and not individually.
- Mattilsynet plays a vital role in monitoring that supervision is carried out correctly.
- BeiteSnap (see Section 2.2.2) is already in use.

That Mattilsynet is highly involved is not unexpected as animal welfare is essential. GPS tracking was not a surprise either, but rather the fact that both GPS tracking and supervision is still needed together. Though, it can be comprehended considering the price of radio bells, and when the farmer has close to 600 outfield grazing sheep. Horvli mentioned that he had 40 radio bells, which covers approximately 7% of his herd (given 600 sheep). Further, Horvli mentioned that he had lost 90 out of 600 animals the last grazing season, which was 15% of the sheep herd he possessed. Thus, it is proved that proper supervision and tracking of the animals are vital jobs.

The full transcribed interview in Norwegian can be found in Appendix C. The prototype demonstration feedback is presented next and was not part of the transcribed interview. Instead, digital notes were taken of the feedback he provided.

Horvli's feedback on the mobile and web application was also highly significant for the project. To get criticism from a relevant user of an information technology system is the most optimal as they are the ones actually to use the system. If the relevant users approve of the user interface and the system flow, it is more likely to succeed.

The feedback on the prototype was generally positive, but expectantly, some changes would be needed, and some new features were mentioned to improve the system. The feedback is summarized in the following list:

- Horvli mentioned that the system seemed nice and that it would be interesting to try.
- **New feature:** He wants to share logged trips in the web application with other users/farmers that have animals in the same outfield pastures. This way, sharing information about their hike route would be simple, and they would avoid walking the same route in a row.
 - With the new feature above, it would be useful if the hike routes of each farmer had a unique color. The hikes of the various farmers would then be displayed in a separate and clear manner.
- **New feature:** Registration of missing, or too many lambs, within a group. Important for knowing if a predator has taken any lambs, or if the lamb has switched mother and is currently with another group of sheep.

- He wants to be able to register sheep and lamb separately, as opposed to only a total number of sheep.
- He wants to be able to register a scenario for a location like "X grown sheep, X lambs, X missing lambs, and X too many lambs".
- **Report generation:**
 - He wants a summary of all the hikes conducted throughout the season: The observations made on shepherd hikes, dates, how many animals collected, and how many let out on the outfield pasture.
 - The time is the essential knowledge, not exactly where the observations are made. With the notion of time, a pattern of when sheep/lamb started missing may appear.
 - What was missing is important in the report.
 - He wants the map reference of where disappearances were observed.
 - Could be useful with a summary of where animals were observed missing.
 - Producing the report yourself in the application is better than others producing it, like with BeiteSnap.
- The applications, and especially the mobile application, have to be simple to use. It must not be too much tampering with the mobile device during the hike, as the weather may vary. The most important is logging the supervision hike.
- The "description" field for the hike in the mobile application should be changed to "purpose".

At the time of the meeting with Horvli, the report generation feature was not implemented. The reason was due to little knowledge of what information was relevant in the reports. However, after the meeting, this became much clearer, and the feature was implemented roughly before the meeting with the County Governor.

8.1.2 Interview 2: The County Governor of Trøndelag

The second interview was with the County Governor of Trøndelag and was conducted in their office in Trondheim April 24, 2018. Eva Dybwad Alstad and Arnstein Lyngstad represented the County Governor in the meeting. This interview was also in cooperation with two other groups working on the same thesis problem, and a joint questionnaire was once again produced in advance of the meeting. Similar questions from the interview with the sheep farmer were included to get a new and another perspective on the same matter. Some new questions were also included and were created to be more relevant for the County Governor.

The meeting was split into three parts, where Alstad and Lyngstad controlled the first part of the meeting and gave some overall information about what the County Governor of Trøndelag does as a representative of the government. They had prepared well and had printed out some useful documents that were given at the end of the

meeting. In the second part, two of the groups demonstrated their prototype systems and were provided some relevant feedback. In the last part of the meeting, the groups took control of the meeting and asked the prepared questions. New and undiscovered information was provided throughout the meeting and led to a better understanding of sheep herding and why proper supervision is necessary. The information and findings from the meeting will be summarized in the following paragraphs.

The County Governor manages grant programs, such as regional environmental grants (Norwegian: *regionalt miljøtilskudd* (RMP)) and organized pasture use (Norwegian: *organisert beite bruk* (OBB)). They provide grants to the farmers per animal they have in the outfield pastures, and if they want to test new relevant equipment and technology, among other grants. They are also involved in the prevention and conflict-mitigating initiatives and compensation for loss of sheep due to predatory wildlife through *Miljøvernavdelingen*, which is the environmental protection department. Thus, *Miljøvernavdelingen* is the department that receives the applications regarding compensation for lost sheep. The reports received from the farmers after the ended grazing season are transferred to *Miljøvernavdelingen* so they can calculate the correct compensation. The reports are also used to calculate the correct grants.

The County Governor does also communicate with other authorities, such as *Mattilsynet*, the municipalities, and the agricultural organizations. *Mattilsynet* is interested in animal welfare and health. Thus, they receive the lists from OBB and controls supervision against each farmer themselves. *Mattilsynet* has laws that state what the farmers must do with their sheep to avoid loss, and if supervision is not conducted or documented correctly, this can result in a deduction of grants from the County Governor. The municipalities and organizations, on the other hand, are involved so they can affect how the County Governor prioritizes the grants and funds.

Further, it was learned that an electronic system called *eStil* [46] is used by the Agricultural Directorate, the County Governor, and the municipalities for processing reports and applications submitted by the farmers regarding grants, such as RMP and OBB. It was not discovered how the application forms look like in *eStil*, but a report schema for organized supervision was given in its paper form. The report schema is displayed in Figure 8.1. The schema has the purpose of ensuring documentation of conducted supervision throughout the grazing season, as required by the regulation called "*Forskrift om tilskott til organisert beitebruk*"⁶¹. It is envisaged that each trip will be recorded continuously on this schema. Relevant information to record may include dead and injured animals, causes, predators, stray dogs, general agitation, ewes with missing lamb, conditions on the grazing field, and other observations. In addition to the form, geographically mapping of observations may also be relevant. When the season is finished, the summary (see the bottom of the document of Figure 8.1b) must be completed and sent to the manager of the *beitelag*. The manager will submit the collected information to the County Governor.

Based on the submitted information and subsidies applications from the *beitelag* in the county, the County Governor can generate different statistics. They receive great amounts of data each year from the reports and applications and have generated

⁶¹<https://lovdata.no/dokument/LTI/forskrift/2003-04-01-427>

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 fortløpende 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 oppsummeringskjemaet 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 sleppt*100)					
Eigeninnsats, dagar:	Tilsyn	Sanking	Anna arbeid	Samla	Køyring, km

(a) Front page.

(b) Back page.

Figure 8.1: Report schema for organized supervision of outfield grazing sheep.

many statistics over the years. The statistics they presented in the meeting was that 77 516 lamb and 127 641 sheep were released in the outfield pastures of Trøndelag in 2017. It was stated that it was a high loss percentage that season, with 10.4% of the lamb and about 5% of the sheep lost. Moreover, *Kilden*⁶², which is the main map solution of NIBIO⁶³ (Norwegian: *Norsk institutt for bioøkonomi*), was mentioned for more relevant statistics. At Kilden one can get information about the different *beitelag* in the country, and how many sheep, lamb, cattle, and so forth, they have had in the outfield pastures. Additionally, one can see the number of lost animals and the loss percentage. A map layer with the loss percentage of lamb was picked at Kilden, and a Web screenshot is displayed in Figure 8.2. The color encoding is explained to the left-center of the screenshot, and the red, pink and dark purple colors show a loss percentage above 9%. An information box from *Sølendalen havnelag* in Rendalen is further displayed in the figure, and the lamb loss for this area was 38.6% (435 of 1127 lamb), which is extremely high compared to 10%. It is unknown how many of the lambs were taken by predatory wildlife.

⁶²<https://kilden.nibio.no/>⁶³<https://nibio.no/>

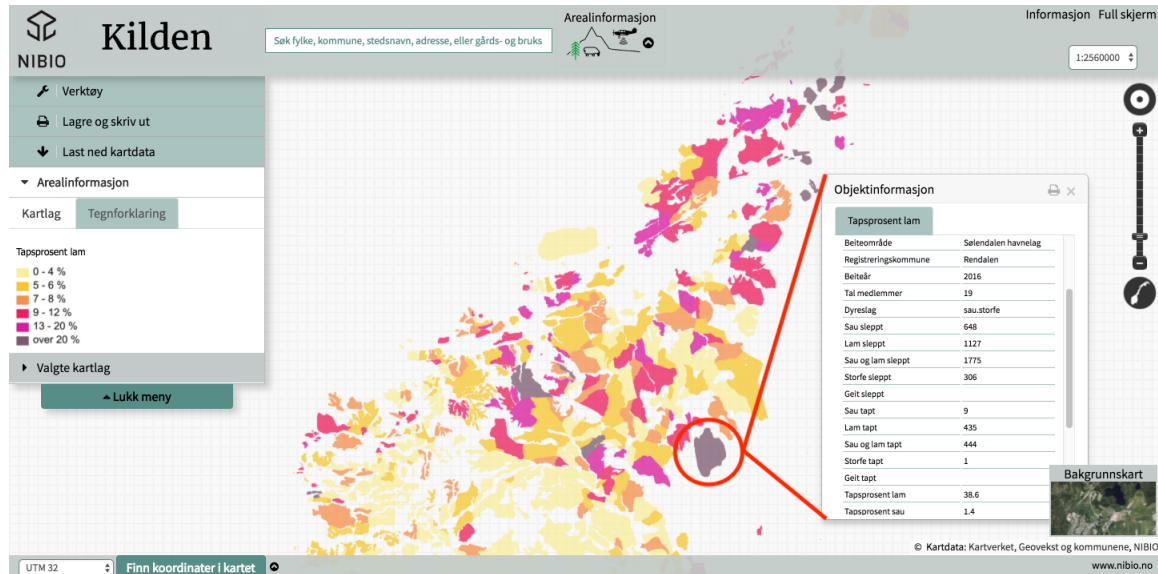


Figure 8.2: Screenshot of "lamb loss percentage" map layer at NIBIO Kilden [47].

As high percentages of sheep are lost each grazing season, it is the opinion of the County Governor that proper supervision is essential and can lead to a reduced sheep loss. Paragraph 19 of the regulation called "*Forskrift om velferd for småfe*"⁶⁴ states that animals in the outfield pastures are to be looked after at least once per week in areas without particular risks. In the event of suspected increased danger, the supervision must be intensified, so conditions that may cause poor welfare, sick, and injured animals are detected as early as possible. To further reduce loss, gathering the sheep earlier from the outfield pastures is also essential. If not, the sheep can be unnecessarily exposed to predatory wildlife at the beginning of autumn.

Despite high loss percentages, the County Governor has observed that the supervision has improved over the years with technology and that they assume it is interesting to see the progress. The radio bell has especially been vital for better supervision, but as it is only used on a percentage of the herd, supervision is still needed to cover large pasture areas with varying terrain. The idea of a mobile application for tracking supervision hikes sounded interesting to them, but as they get all the information they need as of today, they think the application will benefit the farmer more than the County Governor's interests.

Further details from the transcribed Norwegian interview can be viewed in Appendix D.

The feedback given on the prototypes and tips is presented in the following list:

- The County Governor gets detailed enough information from the farmers as of today, as it is enough to calculate the grants and subsidies, and for analyzing the evolution. However, reporting more detailed information could give more purposeful supervision, and it could be more beneficial for the farmers to get a good overview.

⁶⁴<https://lovdata.no/SF/forskrift/2005-02-18-160/>

- Sauekontrollen⁶⁵, which is a system for a comprehensive overview of the farmers' sheep from birth to death, was mentioned as they thought it could be interesting to connect the service with the Pecora system. The sheep farmers are reporting important data regarding their sheep throughout the season to Sauekontrollen, and they can use it to, e.g., compare herd populations.
- Lyngstad thought it would be interesting to test the prototypes on a *beitelag*. Before testing, the statistics for the chosen *beitelag* could be viewed on NIBIO so that the statistics after the prototype testing could be compared.
- **Prototype feedback:**
 - In the mobile application, the observation registration seems small and difficult to handle for larger hands. It should be easier to register observations with bigger buttons and text.
 - The County Governor is not interested in the hike route themselves, but it could be useful for the farmers to plan their trips. The County Governor is only interested in where sheep disappearances were discovered.
 - Producing the reports directly in the web application and being available for the farmer straight away seems better than others producing it, like with BeiteSnap (mentioned in Section 2.2.2).

8.2 Observations

The second data generation method picked was observations. Observations from user testing can provide crucial feedback on how intuitive and user-friendly applications are. The observations were conducted through simple user testing, which involves usability testing. Usability testing refers to evaluating a product by testing it with representative users [48]. Usually, a participant will try to complete typical tasks provided while being observed, and the goal is to identify usability problems and determine the user's satisfaction with the product that is tested [48].

A user test and description were developed after the prototypes had been edited a bit after the demonstrations for Horvli and the County Governor. The test description provided the test object with background information regarding the Pecora system so he could relate a bit more to the problem before executing the test tasks. The test object was further asked to pretend to be a shepherd and that he wanted to conduct a supervision hike. The test tasks were divided into two parts where the first part included tasks for the mobile application and the second part included tasks for the web application. As the user was only intended to conduct one hike in the mobile application, a test account was also used in the web application for visualizing several hikes. The test description and user tasks that were given to the test object can be viewed in Appendix E.

⁶⁵<https://www.animalia.no/no/Dyr/husdyrkontrollene/sauekontrollen/>

8.2.1 Plan

The goal of the user tests was to see how the users attacked the different tasks given and what their first instincts told them to do to complete them, compared to how they were expected to be completed. The observations from a test like this can say a lot about the user interface of an application, and help to see what is logical for the users and not. Results from user testing can help improve the user experience and be helpful input for changes in future versions of applications.

The test tasks are presented in the following list with a task ID (e.g., M1 or W1):

- **Mobile application:**

- M1. Register a user account and log in.
- M2. Download a map of the area you are currently in and where you wish to conduct a shepherd hike.
- M3. Begin a supervision hike with necessary details and the downloaded map.
- M4. Supervision hike: move around on the map and register the following observations with some distance in-between:
 - 20 sheep.
 - 10 lamb near the 20 sheep.
 - 1 golden eagle.
 - 1 dead sheep, which is not covered. Take a picture of it.
 - 10 white sheep.
 - When finished, move back to the starting point and end the hike.
- M5. Find the supervision hike you just saved.
 - Look at the hike on the map.
 - Find the information regarding the dead sheep you observed.
- M6. Test how you delete your hike, but ***do not*** delete it.
- M7. Synchronize your hike.
- M8. Sign out.

- **Web application:**

- W1. Sign into pecora.no with your user account.
- W2. Look at the general information regarding your last hike.
- W3. Display the information regarding the golden eagle.
- W4. Download a summary report regarding your hikes and open it.
- W5. Sign out.

Sign in with username "kari" and password "kari".

- W6. Display the last three supervision hikes on the map.
- W7. Remove the most recent hike from the map.
- W8. Find all supervision hikes between April 26 17:00 and May 3 and display them on the map.
- W9. Reset the hike list.

Sign out.

Four user tests were conducted, and an overview of the tests with their date can be found in Table 8.1 below. The first user test was conducted about three weeks before the following tests, and the reason behind it was the desire to change and improve the observation registration logic based on the results from the first test. Moreover, documentation and report writing were also ongoing and time-consuming tasks that needed to be done at the time.

Table 8.1: Overview of the conducted user tests.

ID	Test	Date
T1	User test 1	May 2, 2018
T2	User test 2	May 25, 2018
T3	User test 3	May 26, 2018
T4	User test 4	May 27, 2018

8.2.2 Results

To get the correct results, the tests were audiotaped to render the user comments and observations as correct as possible. The users needed to approve of an audio recording in advance of the tests, and each user was comfortable with it.

The test results are displayed in two different tables, Table 8.2 for the mobile application, and Table 8.3 for the web application. The tables include the task IDs and the observations made along the way through the user test. The observations made for each task is connected with the test IDs and are collectively summarized from the four tests. If similar observations from the tasks were made, the tests IDs are listed before the observation content (e.g., **T1, T2:** *Some observation*).

Table 8.2: Test results from the mobile application tasks.

Task	Observations
M1	T1, T2, T3, T4: Registration and signing in went fine.
M2	T1, T2: The users did not zoom or pan the map to find the wanted map area to download, they just downloaded it as it was displayed. T1, T3: The zoom-levels was not used. T4: The user zoomed and panned the map, and used the zoom-levels before download.
M3	T1: The user almost forgot to choose the downloaded map when she was beginning a new hike. T2, T3, T4: The users chose the correct map and started the hike.
M4	T1: Starting a new observation point and registering 20 sheep on the map went fine. The user remembered to save the observation point. However, the second time, the observation point was not saved, and the lamb, the golden eagle, and the dead sheep were all registered from the same observation point. T2, T3, T4: Registering most observations went okay. T1, T2, T4: Registering and taking a picture of the dead sheep went fine. T3: The user wanted to register the dead sheep with a picture through the picture-button. Writing details about a dead sheep should be made clearer. T1: The user wondered if it would be wrong to register 10 sheep and then picking 10 white sheep in the details. T2: When registering 10 white sheep, the user forgot to fill in 10 sheep in the first input field and chose only 10 white sheep in the dropdown. T3, T4: Registering 10 white sheep went fine. T1, T2, T3, T4: Ending the hike went fine.
M5	T1, T2, T3, T4: Finding the history of the recent saved hike and viewing the map went fine. Finding the information about the dead sheep went okay, but they needed to remember which observation point they had registered the observation.
M6	T1, T2: Trying to delete a hike went fine. T3, T4: The users wanted to delete the hike when they were viewing the details of it, but figured out how they could delete it from the list.
M7	T1, T2, T3, T4: Synchronizing the hike went fine.
M8	T2, T3, T4: The logout button was found easily.

Table 8.3: Test results from the web application tasks.

Task	Observations
W1	T1, T2, T3, T4: Signing in to the application went fine.
W2	T1, T2, T3, T4: The users struggled to find the general information of a hike on their own, and needed to be given hints of where they could find the information. T3, T4: The users understood that they could click on the route after a while.
W3	T1, T2, T3, T4: Finding the information regarding the golden eagle went fine.
W4	T1, T3, T4: The users struggled a bit to understand that the report generation button was in the <i>settings</i> tab of the menu. After finding it, it was not a problem to download and open the report. T2: The user went directly to the <i>settings</i> tab and downloaded the report, but he had looked through all the tabs to perform task W2.
W5	T1, T4: It took some time before the user found out that the sign out button was in the <i>profile</i> tab. T2, T3: The users went directly to the <i>profile</i> tab to sign out, but they had looked through all the tabs to perform task W2.
W6	T1, T2, T3, T4: The users went directly to the <i>dates</i> tab and clicked on the checkboxes of the last three hikes to show them on the map.
W7	T1, T2, T3, T4: Removing the most recent hike from the map went fine.
W8	T1, T2, T3, T4: Finding the hikes between a chosen time interval went fine and the datetime-pickers seemed user-friendly.
W9	T1, T2, T3, T4: The reset-button was found easily.

Before each user test started, the user was asked to think out loud while performing the tasks. The comments made along the way was helpful to understand *how* and *where* the user thought each task was meant to be completed. The comments from each task are summarized in the following list with the task and test IDs:

- **Mobile application:**

M1. Comments:

- **T4:** Annoying that the keyboard hid the input fields and that you could not scroll down to the remaining fields.

M2. *None.*

M3. Comments:

- **T1:** It was mentioned that "description" of the hike was something she would want to fill in after the hike.

- **T2:** Perhaps picking the desired map should be one of the first options so it is not forgotten and hidden behind the keyboard when filling out other details.
- **T4:** As with the registration, the keyboard was annoying.

M4. Comments:

- **T1, T2:** The users mentioned that they could scroll the application view out of the map area (the view shows grey areas where offline map tiles are not downloaded).
- **T1:** The user asked where the picture of the sheep would be stored, and suggested that the picture-option should be available when she was registering the dead sheep. The user also mentioned that it would be nice to register more observations at once, e.g., 10 sheep and 10 lamb, and their color.
- **T2:** The user asked where the picture of the dead sheep was displayed, and that it would be nice to show it in the pop-up window in the app.
- **T3:** The user thought she could register the dead sheep through the picture button, the same way as through the plus-button with the other observations.

M5. Comments:

- **T4:** The list with the observations was nice in the hike information view, liked that the time was labeled, and so forth. Would be nice if the GPS coordinates were listed too.

M6. Comments:

- **T1, T2:** Deleting a hike was very easy with the label telling you what to do.
- **T3, T4:** It would be nice to delete a hike when viewing its details too.

M7. *None.*

M8. *None.*

• **Web application:**

W1. *None.*

W2. Comments:

- **T1:** The user commented that she would probably find the hike information in the *dates* tab of the menu.
- **T2:** The user thought he would find the information in a table or similar, as one eventually will have many hikes to display.
- **T3, T4:** The users thought it was useful to show the information this way when having several hikes displayed on the map, but that it would be nice with a list similar to the one in the mobile application in the *home* menu.

W3. *None.*

W4. Comments:

- **T1:** The user commented that it was not intuitive to find the report downloading in the *settings* tab. She said she thought it would be in the *dates* tab. Further, she commented that it could be interesting to download reports based on the hikes that you have checked in the list.
- **T2:** The user knew that the report button was in the *settings* tab, as he had looked through the menu to find other information earlier.
- **T3:** The user commented that it was not intuitive to find the report downloading in the *settings* tab and that settings are more for changing a password, etcetera. Could be in the *profile* or *dates* tab instead.
- **T4:** The user meant it was not intuitive to find the report downloading in the *settings* tab, and that he expected it somewhere else. He also thought that a hike needed to be checked before downloading a report. So, it would be nice to choose which hikes to be part of the report, as maybe not all hikes are that interesting.

W5. Comments:

- **T2:** The user did not think it was intuitive to find the logout button in the *profile* tab. He would have liked it in the sidebar menu.
- **T4:** The user commented that it made sense to have the logout button in the *profile* tab.

W6. *None.*

W7. *None.*

W8. *None.*

W9. *None.*

Some general feedback on the applications was also provided and can be found in the following list:

- **T1:** Apart from the earlier comments, the user thought the system was okay and easy to use.
- **T2:** The user interface in the web application is delicate, but it would be nice with an overview of all features in the application when you first log in, and then have the option to view all the hikes on the map after that. Some of the UI logic could be better and more intuitive.
- **T3:** It was fun trying out the system and it was a good demonstration system. Some UI logic could be more intuitive.
- **T4:** The web application is nice with the map and how the hikes are displayed. The user also liked the list design of the hikes in the mobile application, and how the hike details were presented. He though some UI logic was not intuitive as well.

Based on the results of the first user test conducted May 2, 2018 (T1), the observation registration logic was edited and improved. The results motivated the change of the logic where one had to register an *observation point* first to be able to register an *observation*. In the first test, the user forgot to save the observation point after the second observation, which resulted in mapping the remaining observations to a wrong observation point. Regardless of that, the process *was* cumbersome, and it should have been improved earlier in the development. Hence, the process was a bit simplified, and the images of Figure 8.3 display the new registration logic where registration of an observation point has been removed. Now, a new observation is registered directly by pressing the "plus sign"-button, instead of having to push a second button to do the same action. The old logic is found in Section 6.1.

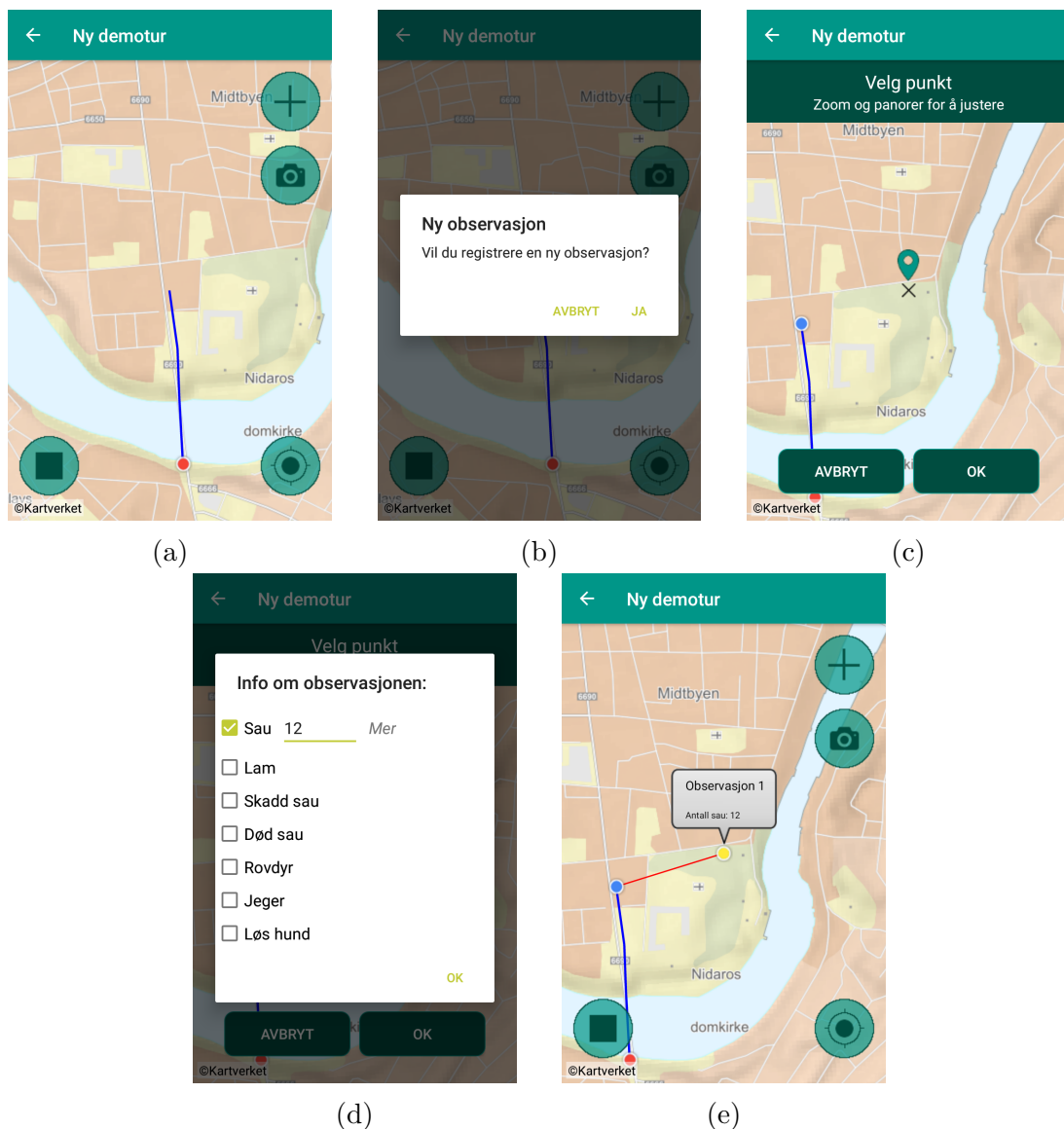


Figure 8.3: App screenshots of the new observation registration logic.

9 Discussion

This chapter discusses the results and findings throughout the master thesis with the research questions in mind. Moreover, the implementation of the Pecora prototype will be evaluated. Based on the evaluation and discussion, a future work section can also be found in this chapter. Lastly, the lessons learned from the thesis will be discussed.

9.1 Review of the Research Questions

The research questions are worth repeating here for a good guideline to evaluate the results and findings of this thesis. The four research questions presented in Section 3.2 were:

- **RQ1: "Will Pecora provide better and more detailed information to the government than how it is today?"**
- **RQ2: "Will Pecora provide better communication between farmers who cooperate with supervision than how it is today?"**
- **RQ3: "Will Pecora provide better and more detailed information for the farmer himself than how it is today?"**
- **RQ4: "Will Pecora increase the efficiency of supervision and reporting information to the government?"**

For an organized overview, the research questions are discussed separately.

Research Question 1 The first research question asks if Pecora will help to provide better and more detailed information to the government than how it is today, where today means writing details of the hike on paper or in a notebook.

The prototype mobile application of Pecora was made with GPS tracking on offline maps, option for taking pictures from the app, and overall digital tracking of a supervision hike. The application was based on the supervisor's experiences when he researched shepherding through participation and assisted a sheep farmer with shepherding. Through the research, Hvasshovd realized that rendering exact locations and observations were difficult with his written notes, and he also mentioned that one needed to be familiar with the area to be able to understand where the locations were by written notes. With GPS tracking in Pecora, the exact locations of a supervision hike and specific observations do not need to be rendered or described, as they are stored in a local database on the mobile phone, and eventually in a second global database if the hikes are synchronized.

Moreover, the web application of Pecora was created to, among other things, generate and download reports with a summary of all the hikes conducted by a sheep farmer. If Pecora summarizes the hike information from the database and calculates the numbers the County Governor requires after the ended grazing season, one can expect fewer faults. Furthermore, if the reports include all hike information with map coordinates, pictures, and so forth, the reports can be expected to be more trustworthy and provide better information.

However, as of today, the Pecora system does not include all the information and calculations that are necessary to report to the County Governor. The information they require can be seen in the table in Figure 9.1, which was presented in the document in Section 8.1.2. The necessary details and calculations include how many animals were released in the outfield pastures, how many were collected, the loss percentage, and their efforts in days of supervision, gathering, other work, and driving in kilometers.

	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

Figure 9.1: Summary table from the supervision document shown in Figure 8.1b.

In the interview with the County Governor (Section 8.1.2) they mentioned they were pleased with the information reported today, and that they do not need more information to perform their work with statistics and calculations of grants and subsidies. Despite this fact, they commented that a system like Pecora would be beneficial for more focused and trustworthy supervision as documentation would be adequately performed with GPS tracking and digitally saved information.

So, the question if Pecora will help to provide better and more detailed information to the government can be discussed. If the discussed calculations and information in the reports were included, Pecora would positively provide more detailed information to the government. However, more detailed information does not necessarily imply better information, and it is a tougher question to answer. As the County Governor is satisfied with the information they receive today, it is harder to argue for better information, but GPS locations and automatic calculations could perhaps increase the reported information quality.

Research Question 2 The second research question asked if Pecora would improve communication between farmers who cooperate with organized supervision. The sheep farmer Horvli mentioned that they had tried BeiteSnap (Section 2.2.2) and that it worked to a certain degree, but the question will be compared to sharing hike information written on paper or in a notebook.

As for the first research question discussed, GPS and digital tracking of hikes will result in more detailed tracking of shepherd hikes than written details on paper. With the hike route digitally stored, the possibility for sharing the supervision hike with other farmers seems simple. Furthermore, it will be no questions as to where the hike was conducted as the route will be tracked. However, as the Pecora prototype is implemented today, it is not easy to share one's hikes with other users. Thus, new features would have to be added to the system to support sharing among users.

Through the interview with Horvli, he mentioned that it was not a problem to share information in-between the *beitelag* today. He did not comment how they shared their information. Nonetheless, Horvli said he liked how the web application displayed the hike trails with different colors (see Figure 6.12) and that it could be useful to have a joint application where the farmers in the *beitelag* could view each others' hikes. This feature will be further discussed in the future work section (Section 9.2).

To conclude and answer the research question, a future version of Pecora may improve communication between the farmers within organized supervision if they can view other farmer's shepherd hikes in the application. Without this feature, the communication is apparently good enough as it is when it is told so by a sheep farmer who has had experience with supervision from many grazing seasons and shepherd hikes over the years.

Research Question 3 The third research question asked if Pecora would provide better and more detailed information for the farmer himself than how it is today, where today means writing details of the hike on paper or in a notebook.

This third question is closely entwined with RQ1, but it is also different as the sheep farmer and the County Governor have two different information needs. The County Governor is interested in detailed enough information to verify that the farmers are supervising their sheep, to calculate grants, and to create statistics. On the other hand, the sheep farmer is interested in their sheep and relevant observations, such as missing sheep or seen predators in the grazing areas. They must also document the shepherd hikes properly not to lose subsidies. In the meeting with the County Governor, they mentioned that as long as the Pecora applications satisfy the sheep farmer, it would satisfy them. Thus, it is the sheep farmer one should focus on when developing Pecora and its functionality.

In the meeting with Steingrim Horvli, he informed that supervision needed to be conducted once per week at minimum and that he spends about 10 to 12 hours on shepherd hikes per week through the grazing season. Keeping track of all this hike information by writing in a notebook seems like cumbersome work, where one must turn pages back and forth in the book to retrieve wanted information. As the government has no clear guidelines on what observations must be registered, every

detail is written down to document the hikes. He further commented that registering text and pictures are essential, but it is not quick to connect written details with a picture taken with one's camera or mobile device. With the Pecora application, text and pictures could easily be connected if it were implemented. At the moment, the Pecora app does not store the pictures taken with a connection to an observation, but this feature is discussed in the future work of Pecora, Section 9.2.

Moreover, Horvli mentioned that the farmers themselves mean that tracking the hike route is crucial on a shepherd hike although it is not needed to report to the County Governor. The hike route is beneficial to track as they need to cover different areas of where the sheep are wandering in the outfield pastures and sharing it with the farmers in the *beitelag*. Sharing the route is vital as walking the same trip twice in a row is an inefficient and unnecessary use of time. So, Horvli commented that the tracking in the Pecora application was useful and needed even though they have managed to share this information without mobile device tracking before.

Horvli and some other farmers in his *beitelag* had tried BeiteSnap the previous grazing season (Summer 2017). BeiteSnap was a new application last year, and Horvli and the others thought it was a useful application. This shows that farmers are interested in trying out new technology to make their work tasks as a farmer easier to conduct. However, the technology and applications must be available and reliable. Horvli commented that BeiteSnap had stopped working when he was tracking a shepherd hike and that the work he had put into the hike was a total waste of time. Thus, Pecora would need to be reliable if it were to be tested and released.

All in all, an application as Pecora could at least help to provide more organized information to the farmer, if not more detailed and better. If the application facilitates easy registration of detailed observations in a future version, the information could be more detailed as well. Moreover, the farmer could get better information and overview of missing sheep if the application would support better functions for registering too many or too few lamb within a group. To conclude, it is expected that digital information will provide more organized and well-structured hike information than written details on paper.

Research Question 4 The fourth, and last, research question asked if Pecora would increase the efficiency of supervision and the process of reporting information to the government than how it is today. Many aspects of how Pecora could help the shepherding has already been discussed, but some of them will be re-discussed here in the light of RQ4.

The supervision with the shepherd hikes in mind will probably not be more efficient with a system as Pecora, as the hours spent on supervision will still be the same. The supervision must be done by hiking the outfields pastures, and details must be recorded. The mobile application can perhaps be simpler to handle than a notebook if the observation registration is optimized, but the minutes spent on each observation will be quite similar to writing the details.

However, the post-hike processing can become more efficient with an information system, as calculations and summaries can be computed fast by computers. If the

information from the table in Figure 9.1 would be calculated and downloaded by a button in the web application of Pecora, this would save the farmer some time. As of today, the Pecora application only summarizes the hikes briefly in the generated reports, but this content could easily be changed with requested information.

The future work section below will discuss how it could be implemented.

9.2 Future Work

This future work section discusses improvements, changes, and features that should be implemented in the Pecora system before testing it on sheep farmers within a *beitelag* in the future. The changes are based on the feedback received from the sheep farmer Steingrim Horvli and the County Governor. Moreover, the user test results were good input on what should be changed about the user interface, in both the mobile and web application. The user test results and observations will be presented and discussed first.

The user test results were beneficial for the future work of the Pecora system, and the most crucial changes that should be done with the user interface based on the observations and comments from the tests (Section 8.2.2) are summarized in the following list:

- **User interface changes in the mobile application:**
 - Edit how the download of a map is represented in the mobile application. Could be more similar to the UI of the Norgeskart application, which is displayed in Figure 9.2a. The user test showed that it was not very intuitive to zoom and pan the map to download the wanted area. The zoom-levels can be removed as one person only used them out of four.
 - When starting a supervision hike, have the user pick an offline map first and have the most recent downloaded map as default, so the user does not forget it.
 - Make it possible to scroll in the views where there are input fields, so the keyboard is not in the way for the user.
 - Change "description" of a hike into "purpose" or something similar to not confuse the user.
 - In the supervision hike activity, remove the camera button and make it a part of the observation registration.
 - It would be nice to restrict the map view to the offline map boundaries so the user can not scroll out of the map to the grey areas where map tiles are not downloaded.
 - When registering a new observation, display the possible observations better, e.g., as in Figure 9.2b.
 - When registering a sheep observation, make it easier for the user to register color and quantity at the same time. Perhaps something similar to the sketched prototype in Figure 9.2c.

- In the history view of a hike, include the GPS coordinates of observations in the observations list. Further, show the observations details in this list, so the user does not need to tap an observation for more details.
- Add a delete hike button in the history details view.

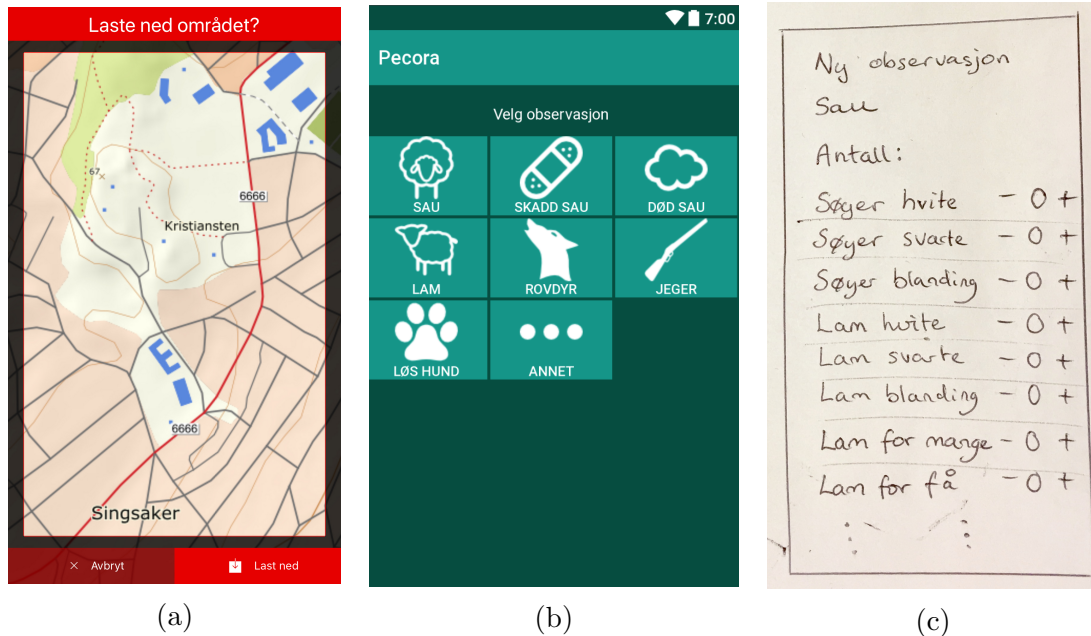


Figure 9.2: Possible UI improvements of the mobile application.

- **User interface changes in the web application:**

- Add the information of a hike in a list in the *home* tab of the menu, as the user test results showed that it was not very intuitive to click on the hike route on the map to get this information. Make the information list more similar to the hike information in the mobile application. See Figure 9.3 for sketched improvement suggestion. Despite this, keep the information in a popup of the hike route so the user can easily click on this when displaying several hikes on the map.
- Display pictures taken of observations in their popups when the markers are clicked. Further, the pictures must be included in the downloaded reports.
- Add another icon and tab in the menu, so the report generation button is more easily found or move the button to either the *home* tab or the *dates* tab. The user test showed that it was not intuitive to have it in the *settings* tab, and it is understandable.
- The test results showed that some users thought they needed to check hikes to download the report. Thus, it could be an idea to download reports with wanted hikes, where the user checks the hikes to be included in the report.

- Two out of four users thought the logout button was misplaced so that a logout button could be part of the sidebar menu, so one does not need to look for it in the *profile* tab.

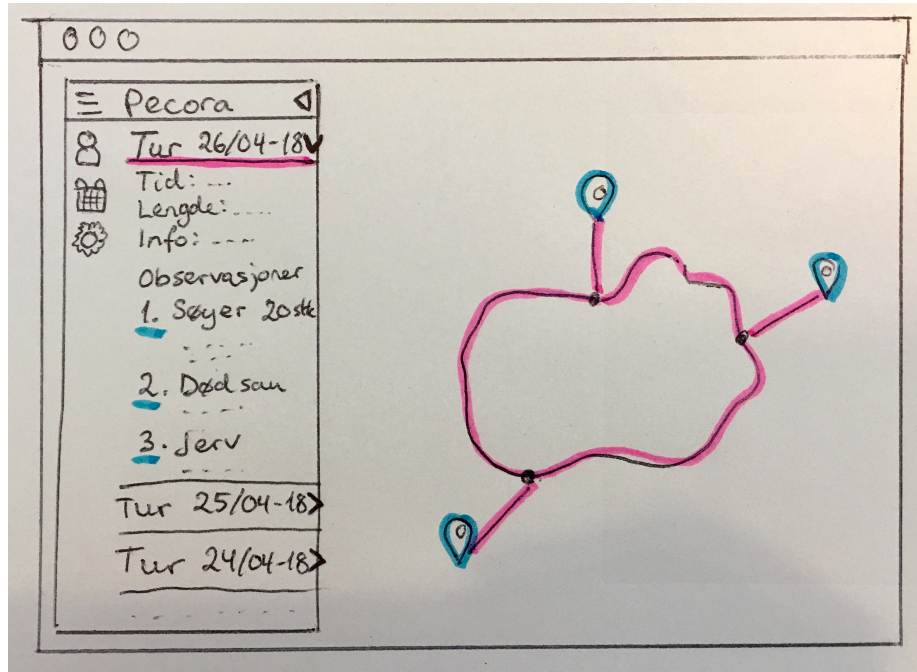


Figure 9.3: Paper prototype for improving the web application.

Based on the feedback from the interviews conducted in the thesis and the test results, some new functional requirements were derived as well. Table 9.1 includes the future functional requirements that should be implemented in the Pecora system to satisfy a sheep farmer's needs better, and to generate better and more detailed reports after the ended grazing season. The functional requirements have an ID, a priority ranging from low to high, and a description. At the beginning of the description, it is specified whether the requirement regards the mobile or web application.

Table 9.1: Future functional requirements for the Pecora system.

ID	Priority	Description
FR1	High	(Web) The user should be able to create a <i>beitelag</i>
FR1.1	High	(Web) Option for adding users to a <i>beitelag</i>
FR1.2	High	(Web) Option for deleting users from a <i>beitelag</i>
FR1.3	High	(Web) Option for viewing hikes from all users in a <i>beitelag</i>
FR1.4	High	(Web) Assign each user in a <i>beitelag</i> a unique color and display hike routes with the color

FR1.5	High	(App) Option for sending a notification to the users in a <i>beitelag</i> about a acute situation, e.g., predator attack or hurt sheep
FR2	High	(App) Change UI of observation registration (similar to Figure 9.2b and 9.2c)
FR3	High	(App) Option for registering observations with a picture taken or from gallery
FR3.1	High	(App & Web) Display the pictures taken/picked as part of the information in pop-ups
FR4	High	(App) Automatic synchronization of hikes to database server
FR5	High	(Web) The user should be able to view the hike information in a list in the <i>home</i> menu
FR6	High	(Web) The user should be able to download a detailed report including a summary of all hikes with pictures and GPS coordinates of important observations
		(Web) Calculate a shepherd hikes summary based on following registered data:
FR6.1	High	<ul style="list-style-type: none"> - Register how many sheep and lambs released in the outfield pastures - Register how many sheep and lambs gathered from the outfield pastures - Register how many hours spent on gathering the animals from the outfield pastures and other related work - Register how many kilometres driven in supervision context

The future functional requirement FR1 and its five sub-requirements revolve around the idea of registering different *beitelag* in Pecora so that the farmers within a *beitelag* can cooperate efficiently with the organized supervision. If this option was to be added, the system should support adding and deleting members from a *beitelag* (FR1.1 and 1.2), viewing of each others' hikes (FR1.3), and perhaps an emergency option to notify the farmers in the team if an acute situation occurs while on a shepherd hike (FR1.5). The requirements have gotten a high priority as the sheep farmer Horvli mentioned this feature, and therefore it is considered very important as the applications are meant to satisfy sheep farmers.

FR2, FR3, and FR4 revolve around the mobile application. FR2 involves changing the UI of observation registration, which was mentioned in the UI changes of the mobile application. FR3 is about the option to register observations with a picture, and its sub-requirement involves that the pictures should be included in the information popups in the mobile and web application. Thus, the pictures taken must be connected to an observation. FR4 involves that the app will synchronize the hikes automatically to the database server when it detects a network connection, so the user will not have to do it manually. FR2 and FR3 have been given a high priority as observation registration and the inclusion of a picture are considered highly significant

for the mobile application. The functional requirements must be implemented so the farmer can easily register different observations. FR4 was given a high priority too, as automatic synchronization between different components of a system is considered as a default setting today.

FR5 and FR6 regard the web application. FR5 involves that the system should display the hike information more clearly and more intuitive to the user, like in the sketch of Figure 9.3. FR6 revolves around a better report generation, and that it should be more similar to the supervision report that was received by the County Governor. The report generated by the system should include all the details the farmers have registered, and a summary table like the one of Figure 9.1. So, the system must support registration of the different data mentioned in FR6.1. FR5 and FR6 were given a high priority based on different reasons. FR5 got a high priority due to the user test results where the users expected to see the details of a hike as part of the menu, and more similar to the way it is visualized in the mobile application. It would give the user a better overview, and it would be more intuitive, which is crucial for user experience. FR6 is vital as Pecora has a purpose of increasing the efficiency of supervision hikes and the post-processing of the information recorded. Thus, FR6 must be implemented to fulfill this purpose.

With the user interface changes and implementation of the functional requirements mentioned in this section, the Pecora system will have a better chance of being successful through a test period with sheep farmers within a *beitelag*.

To process the hike data and generate reports more easily, the database design and implementation would also have to be reconsidered and improved. As mentioned in Section 7.1.5, the database design does not comply with the first normal form (1NF) as the values of *observationPoints* and *track* can be divided further. A suggestion to a new database design can be found in Figure 9.4.

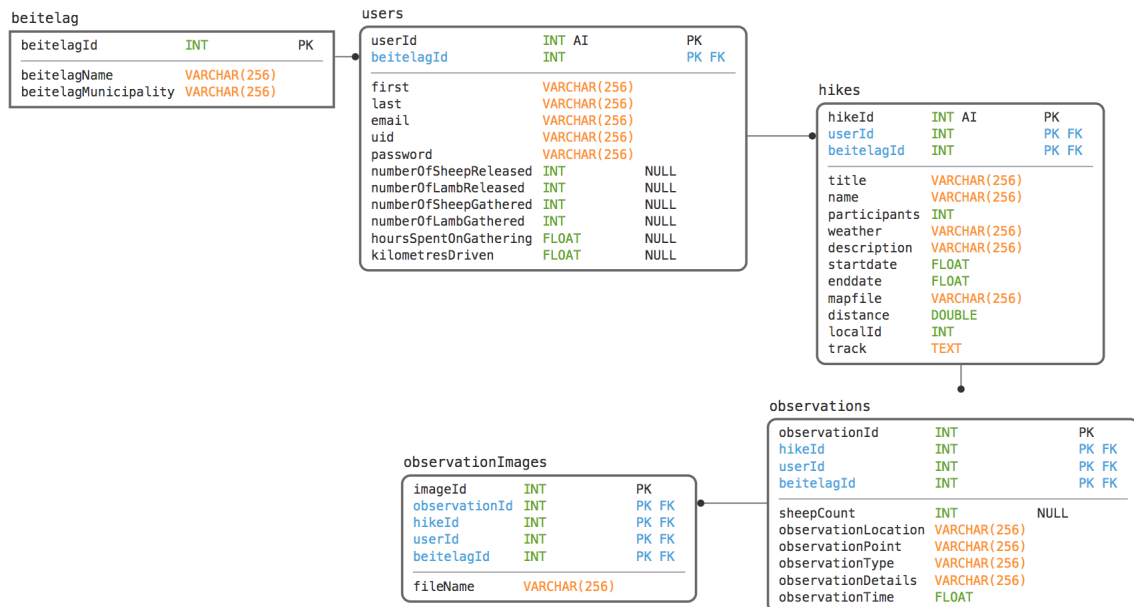


Figure 9.4: Suggestion for database design improvement.

Based on the new and future functional requirements, some changes to the database must be performed. The changes needed are summarized in the following list:

- **Database design changes:**

- A *beitelag* table can be added with an ID, name, and municipality. A *beitelag* can have a reference to 1 or more *users*.
- Add fields to the *users* table:
 - * Number of released sheep and lamb in the outfields pastures.
 - * Number of gathered sheep and lamb from the outfields pastures.
 - * Hours spent on gathering sheep and lamb from the outfields pastures.
 - * Kilometres driven in supervision context.
- Remove *observationPoints* from *hikes* table and create *observations* table. The table must have fields as ID, sheep count, location, observed from point, type of observation, details, and time of observation.
- A table for observation images can be added with fields as ID and file name, and a reference to *observations*.

Lastly, for this future work section, security measures are worth mentioning. As the applications designed and created throughout this thesis were only prototypes, proper security measures were not considered. This would be crucial to include in a release product. Some security aspects are summarized in the following list:

- **Security measures to consider:**

- Better password hashing. As of today, the password hashing used is the default one of PHP without a salt. It is recommended that passwords be hashed with SHA-256 or higher with a salt [49], and this should be included in a future version of Pecora.
- Encryption of location data. Location data is sensitive information, and as it will be stored in a database with reference to a user, this data should be encrypted.
- Proper validation of user input. Proper validation of user input should be implemented in a future version so no malicious code can be injected into the system and harm the database. String escaping, regular expressions, and validation of file extensions are examples.
- Web application framework. Using a web application framework for the website of Pecora would be expedient, and a proper architecture, e.g., Model-view-controller (MVC) [21].

9.3 Lessons Learned

Throughout the master thesis, some lessons were learned that is worth mentioning. This includes early involvement of end users, the importance of prototyping, and technology research. The three aspects are discussed separately.

Involving the End Users After a prototype of the Pecora system was finalized, it could finally be shown to a relevant end user for proper critique and feedback. The sheep farmer we met with provided useful information regarding shepherding and feedback on the prototype. This was used to implement some remaining planned features further and to plan the future work of Pecora. However, it would have been beneficial to meet with the sheep farmer earlier in the process. If the meeting was held earlier, the most critical features could have been focused on and been optimized, rather than spending unnecessary time on other features. Moreover, perhaps some of the future functional requirements could have been implemented so the Pecora prototypes could have been closer to a test product for a *beitelag*.

Prototyping The importance of prototyping has been discovered throughout this master thesis. Prototyping has been a significant factor as it was used to show something tangible to end users of Pecora. Prototyping is time-saving and straightforward as one can discover what users need in a system before implementing it. Hence, it can help to avoid spending too much time on something the users do not need. The paper prototypes made of the web application, for instance, turned out to be useful for planning the user interface. The paper prototype was shown to the supervisor, which resulted in relevant feedback on what should be added and removed.

However, more thorough prototyping could have been conducted for this supervision system by using online prototyping tools, such as Proto.io⁶⁶. With a tool like this, one can generate prototypes that feel and look like actual applications. Thus, if this kind of prototyping had been used early in the project followed by some quick user tests, perhaps some functionality of Pecora could have been better, more intuitive, and more user-friendly. The observation registration in the mobile application, for instance, could have been more user-friendly by prototyping and testing. It was spent much time implementing the feature, and in the end, it was proven by the test results that it was not an optimal logic for registering observations. So to conclude, prototyping can be crucial for software systems and development.

Technology Research Technology research and decisions proved to be a crucial and time demanding part of the Pecora software development. The cloud service decision was extraordinarily time-consuming as little experience in this area of development was present. It was experienced that searching for information, watching tutorials, and trying out the different technology was the most efficient way to decide which technology to use.

⁶⁶<https://proto.io/>

Earlier in Section 5.1 it was mentioned that web technology could have been used to develop the mobile application instead of developing specifically for Android by using Android Studio. This was discovered after the technology decisions were made, which shows how important it is to perform proper technology research. If web technology were used to develop the mobile application, much of the code would have been reusable for the web application as Leaflet would probably have been used to implement the map. Further, there would have been more time to implement other features, resulting in a better and more complete prototype. However, this is easy to see in retrospect, and one can only see this as an experience and lesson for future projects.

10 Summary and Conclusion

This finishing chapter includes a summary of the project and a final conclusion based on the previous discussion chapter.

10.1 Summary

As the mobile application of Pecora was prototyped in the specialization project, this thesis started with the planning of the web application that was part of the system from the beginning. The supervisor created a system specification for the mobile application, but the web application could be planned freely. During design planning, inspiration was found at Skisporet's web page, which shows ski trails with different colors on the Norwegian map. Paper prototypes were created of how shepherd hikes could be visualized on a map with this inspiration in mind. After paper prototyping, the system was further planned and divided into three different parts; a web application, cloud solution, and mobile application.

A methodology was chosen, and a design and creation strategy based on the supervisor's experiences from research by participation and the research questions was picked. Creating the prototype system was crucial to have something tangible to show a sheep farmer so he could understand his needs in a supervision system. Thus, the three parts of the system were designed and created, and a prototype was made and ready for a demonstration with relevant end users. The web application became quite similar to the paper prototypes shown in the design planning, which showed that the planning at the beginning of the project was essential and was a foundation for the implementation.

Two interviews and meetings were conducted throughout the master thesis. The first meeting was with the sheep farmer Steingrim Horvli, and this gave relevant information regarding shepherding and useful feedback on the prototypes made. The second meeting was with the County Governor of Trøndelag, and they did also provide relevant information as to how the sheep farmers must report supervision information after a grazing season and apply for different grants and subsidies. Based on the feedback, much work and features could have been implemented in the Pecora system, but as the project had time and system boundaries, they were included in the future work of the applications.

User testing of the applications was also conducted as user experience of an application is crucial for it to be successful. Four tests were conducted, and it was discovered that some user interface changes needed to be done to make the applications more intuitive. Unfortunately, the time did limit further development to improve the UI, and it was included in the future work as well.

A status table of the functional requirements is included here to show which requirements were implemented and not. The table only includes the requirements' IDs, but the full requirements tables can be viewed again in Chapter 4. In the table,

a check mark indicates that the requirement was implemented and a cross indicates the opposite. The list is found in Table 10.1.

Most of the functional requirements were implemented, apart from FR15 and FR16. FR15 involved registration of movement and area of an observation in the mobile application. FR16 involved that the user should be able to register an existing observation with a new location. Both of the requirements had a medium priority, which meant they were not highly prioritized in the project. Despite this, they are considered essential features of a supervision hike application. FR17 is also a functional requirement to mention, as it is implemented, but not thorough. FR17 revolves around taking pictures of observations, and one can take pictures in the mobile application. However, the pictures taken are not displayed in the mobile application or the web application, and they are not synchronized to the server. So this feature was also included in the future work of Pecora.

Table 10.1: Implementation status for all functional requirements from Chapter 4.

ID	Status	ID	Status
FR1	✓	FR9	✓
FR2	✓	FR10	✓
FR2.1	✓	FR10.1	✓
FR2.2	✓	FR10.2	✓
FR3	✓	FR10.3	✓
FR3.1	✓	FR10.4	✓
FR3.2	✓	FR10.5	✓
FR4	✓	FR10.6	✓
FR4.1	✓	FR11	✓
FR5	✓	FR11.1	✓
FR6	✓	FR12	✓
FR7	✓	FR13	✓
FR8	✓	FR14	✓
FR8.1	✓	FR15	✗
FR8.2	✓	FR16	✗
FR8.3	✓	FR17	✓
FR8.4	✓	FR18	✓
FR8.5	✓	FR18.1	✓
FR8.6	✓	FR18.2	✓
FR8.7	✓	FR18.3	✓
FR8.8	✓	FR19	✓

10.2 Conclusion

The research questions asked in Chapter 3 revolved around if a system like Pecora would increase the efficiency of free-range sheep supervision. Further, they asked if it would provide better and more detailed information to the government and the sheep farmer himself. With the interviews and feedback from Horvli and the County Governor as a foundation, Pecora could possibly fulfill this purpose. However, the features mentioned in the future work section would have to be implemented.

A future feature involved an option for creating a *beitelag* in the system and the possibility for viewing the hikes of members within the *beitelag*. This could improve the communication between the farmers that cooperate with supervision. Thus, it could perhaps improve the efficiency of organized supervision within the area. Another feature involved the report generation, and if the table in the document received from the County Governor were to be automatically calculated in the reports by the system. This could help save time for the shepherd, as he would not need to summarize all the hours he has spent on shepherd hikes, etcetera, if he fills in the needed information before and after the grazing season.

The features discussed were discovered through the interviews with sheep farmer and the County Governor, which shows that involving the end users early is essential in software development. If the sheep farmer had been involved earlier, perhaps the *beitelag* feature could have been implemented in this thesis project and the observation registration could have been better. Despite this, a prototype was created, and it was discovered useful information and features for further work of a supervision system for sheep farmers.

As of today, the current Pecora system could make supervision of free-range sheep in the outfield pastures more efficient, although some features are missing. The system could assist to avoid redundant work as the observations noted during the hikes would already be electronic. Further, rendering the different hikes and its' details would be more accessible for the sheep farmer himself as the hike data would be displayed on a map with all the details noted.

References

- [1] Frida Meland Schmidt-Hanssen. Mobile application for monitoring grazing sheep. https://www.dropbox.com/s/5psf8xyerko5ms5/TDT4501_Fordypningsoppgave_FridaMSH.pdf?dl=0, 2017. Retrieved 2018-05-29.
- [2] Jon Inge Bragstad. Merking med bjelleslips. <http://www.nsg.no/rovdyr-og-utmarksbeite/merking-med-bjelleslips-article2433-1328.html>, 2011. Retrieved 2018-04-12.
- [3] Anita Borkamo and Steinar Rostad Breivik. Norske sauer føder for mange lam. https://www.nrk.no/nordland/_-store-lammekull-er-dyreplageri-1.7693253, 2011. Retrieved 2018-04-25.
- [4] Kjell Smestad. Slipstvang. <http://www.kjellsmestad.no/files/slips.html>, 2008. Retrieved 2018-04-12.
- [5] Anne Skjølvold. Til fjells med radiobjeller. <https://www.trollheimsporten.no/til-fjells-med-radiobjeller.4795792-137312.html>, 2010. Retrieved 2018-04-12.
- [6] Martin NH. Sauer i den norske fjellheimen. https://upload.wikimedia.org/wikipedia/commons/7/7a/Sauer_i_fjellet_cropped.jpg, 2012. Retrieved 2018-04-12.
- [7] William F. Wood. The wolverine, gulo gulo is the largest land-dwelling species of the family mustelidae (weasels). https://en.wikipedia.org/wiki/Wolverine#/media/File:Wolverine_01.jpg, 2004. Retrieved 2018-04-12.
- [8] Martin Mecnarowski. Lynx lynx, nationalpark bayerischer wald, deutschland. [https://no.wikipedia.org/wiki/Gaupe#/media/File:Lynx_lynx_1_\(Martin.Mecnarowski\).jpg](https://no.wikipedia.org/wiki/Gaupe#/media/File:Lynx_lynx_1_(Martin.Mecnarowski).jpg), 2008. Retrieved 2018-04-12.
- [9] Juan Lacruz. Golden eagle aquila chrysaetos, la cañada, Ávila, spain. https://upload.wikimedia.org/wikipedia/commons/c/c6/Chrysaetos_La_Ca~nada_20120114_1.jpg, 2012. Retrieved 2018-04-12.
- [10] Telespor. Telespor. <http://telespor.no/>, 2018. Retrieved 2018-04-06.
- [11] BeiteSnap. Beitesnap. <http://www.beitesnap.no/>, 2018. Retrieved 2018-04-06.
- [12] Fant AS. Beitesnap. <https://play.google.com/store/apps/details?id=no.beitesnap.beitesnap&hl=no/>, 2018. Retrieved 2018-04-15.
- [13] Findmy. Findmy. <http://findmy.no/>, 2018. Retrieved 2018-04-06.

-
- [14] Briony J Oates. *Researching information systems and computing*. Sage, 2005.
- [15] Jarg Bergold and Stefan Thomas. Participatory research methods: A methodological approach in motion. *Historical Social Research/Historische Sozialforschung*, pages 191–222, 2012.
- [16] Ken Schwaber. Scrum development process. In *Business object design and implementation*, pages 117–134. Springer, 1997.
- [17] Skisporet. Skisporet. <https://skisporet.no/>, 2018. Retrieved 2018-04-12.
- [18] Libby Bawcombe. The hamburger menu-icon debate. <https://www.theatlantic.com/product/archive/2014/08/the-hamburger-menu-debate/379145/>, 2014. Retrieved 2018-04-04.
- [19] Cappelen Damm. Samfunnsfaglig metode. <http://delta.cappelendamm.no/vgsamf/binfil/download2.php?tid=1714577&h=1c3f3d2c9dd4879f06b772756231d32c&kap=1685917>, 2018. Retrieved 2018-05-07.
- [20] Vincent Rainardi. Functional and nonfunctional requirements. *Building a Data Warehouse: With Examples in SQL Server*, pages 61–70, 2008.
- [21] Len Bass. *Software Architecture in Practice*. Pearson Education India, 2007.
- [22] Google. What is cloud computing? <https://cloud.google.com/what-is-cloud-computing/>, 2018. Retrieved 2018-01-17.
- [23] James Vincent. 99.6 percent of new smartphones run android or ios. <https://www.theverge.com/2017/2/16/14634656/android-ios-market-share-blackberry-2016>, 2017. Retrieved 2018-04-15.
- [24] Apple. Start developing ios apps (swift). <https://developer.apple.com/library/content/referencelibrary/GettingStarted/DevelopiOSAppsSwift/>, 2018. Retrieved 2018-04-15.
- [25] Google. Android studio. <https://developer.android.com/studio/index.html>, 2018. Retrieved 2018-04-15.
- [26] Dmytro. ios vs android development. <https://theappsolutions.com/blog/development/ios-vs-android/>, 2018. Retrieved 2018-04-15.
- [27] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.
- [28] Clutch. Clutch. <https://clutch.co/cloud>, 2018. Retrieved 2018-04-15.
- [29] James Gosling, Bill Joy, Guy L Steele, Gilad Bracha, and Alex Buckley. *The Java Language Specification*. Pearson Education, 2014.

-
- [30] W3C. Extensible markup language (xml) 1.0 (fifth edition). <https://www.w3.org/TR/REC-xml/>, 2008. Retrieved 2018-04-15.
- [31] SQLite. About sqlite. <https://www.sqlite.org/about.html>, 2018. Retrieved 2018-04-17.
- [32] Google. Save data in a local database using room. <https://developer.android.com/training/data-storage/room/>, 2018. Retrieved 2018-04-26.
- [33] Google. Transmitting network data using volley. <https://developer.android.com/training/volley/index.html>, 2018. Retrieved 2018-04-17.
- [34] Mozilla. Javascript. <https://developer.mozilla.org/bm/docs/Web/JavaScript>, 2018. Retrieved 2018-04-17.
- [35] phpMyAdmin. Bringing mysql to the web. <https://www.phpmyadmin.net/>, 2018. Retrieved 2018-04-17.
- [36] The PHP Group. What is php? <https://secure.php.net/manual/en/intro-what-is.php>, 2018. Retrieved 2018-04-17.
- [37] JSON. Introducing json. <https://www.json.org/>. Retrieved 2018-05-21.
- [38] W3C. Extensible markup language (xml). <https://www.w3.org/XML/>. Retrieved 2018-05-21.
- [39] W3C Working Group Note. Web services architecture. <https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest>, 2004. Retrieved 2018-04-18.
- [40] Kartverket. Kartverkets wms- og cache-tjenester. <https://www.kartverket.no/data/api-og-wms/>, 2018. Retrieved 2018-04-15.
- [41] Osmroid. Osmroid. <https://github.com/osmroid/osmroid>, 2018. Retrieved 2018-04-15.
- [42] Ben Popper. Google announces over 2 billion monthly active devices on android. <https://www.theverge.com/2017/5/17/15654454/android-reaches-2-billion-monthly-active-users>, 2017. Retrieved 2018-04-18.
- [43] Charles Suscheck. Defining requirement types: Traditional vs. use cases vs. user stories. <https://www.cmcrossroads.com/article/defining-requirement-types-traditional-vs-use-cases-vs-user-stories>, 2012. Retrieved 2018-05-09.
- [44] Ramez Elmasri and Shamkant Navathe. *Fundamentals of database systems*. Addison-Wesley Publishing Company, 2010.

- [45] Andrew and AWS Tutorial Series. Connecting to rds mysql using phpmyadmin (ec2, rds). https://www.youtube.com/watch?v=Bz-4wTGD2_Q, 2015. Retrieved 2018-02-08.
- [46] Statens landbruksforvaltning. estil - fagsystemet, saksgang og årshjul. <https://www.fylkesmannen.no/Documents/Dokument%20FMVE/Landbruk%20og%20mat/Kompetansesamlinger/130524%20sStil/Hva%20er%20eStil.pdf>, 2013. Retrieved 2018-05-23.
- [47] NIBIO. Kilden - arealressurs. https://kilden.nibio.no/?X=7040032.08&Y=326553.45&zoom=2&lang=nb&topic=arealinformasjon&bgLayer=graatone_cache&layers_opacity=0.75&catalogNodes=87&layers=beite_sau_lam, 2018. Retrieved 2018-05-22.
- [48] Usability.Gov. Usability testing. <https://www.usability.gov/how-to-and-tools/methods/usability-testing.html>, 2013. Retrieved 2018-04-15.
- [49] The PHP Group. Safe password hashing. <https://secure.php.net/manual/en/faq.passwords.php>, 2018. Retrieved 2018-06-04.

Appendices

A Specialization Project

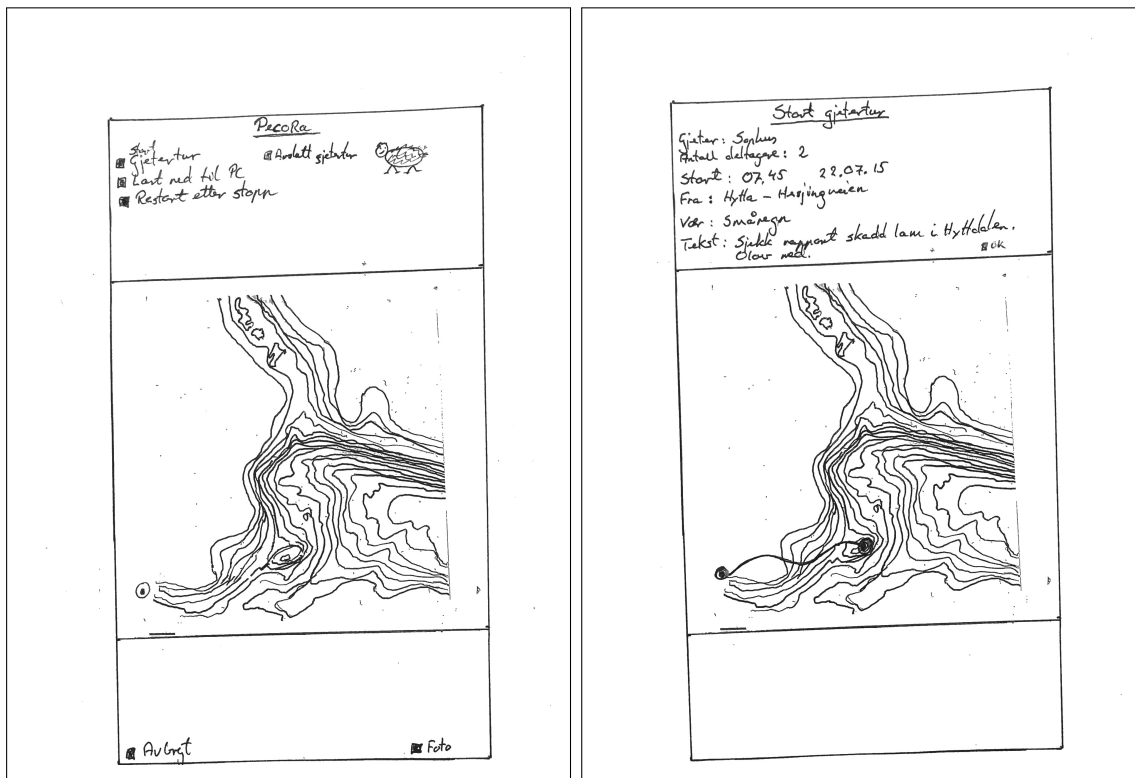
This appendix contains details from the specialization project conducted in the Autumn 2017. The first section includes the link to the specialization project report, and the second section includes some of the pages of the mobile application specification created by the supervisor, Svein-Olaf Hvasshovd.

A.1 Report and Specification Links

- Specialization project report: <https://goo.gl/dLVBUR>.
- Mobile application specification: <https://goo.gl/XRj5eL>.

A.2 Mobile Application Specification

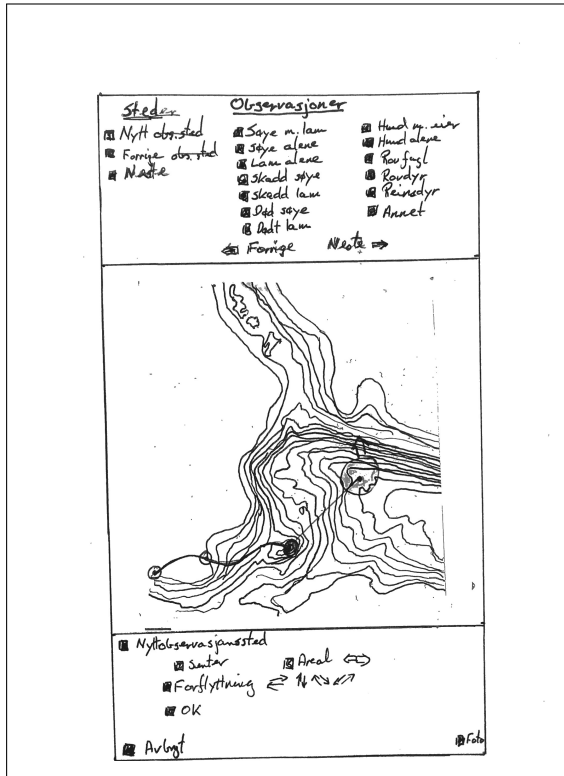
This appendix section includes some selected pages from the mobile application specification.



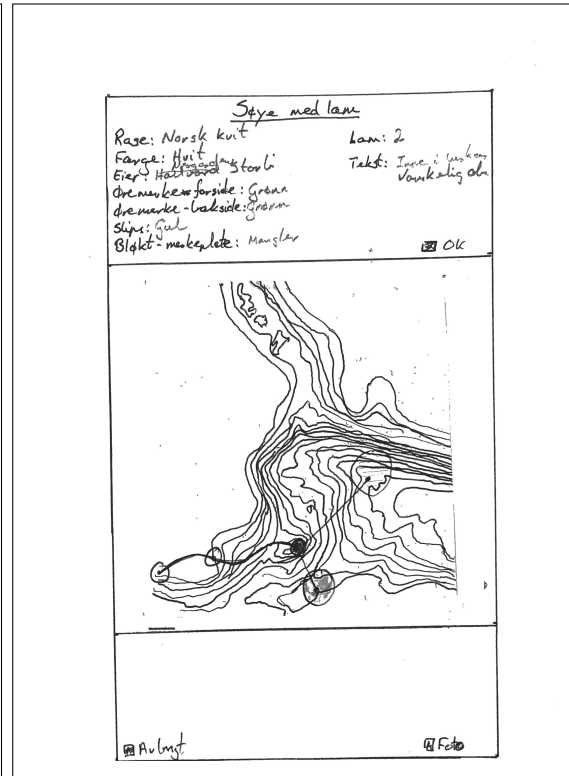
(a) Main menu.

(b) Begin shepherd hike.

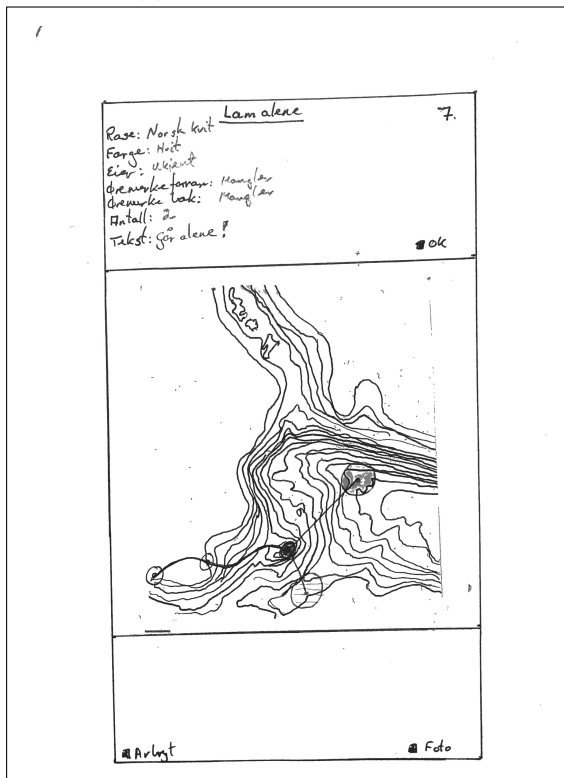
Figure A.1: Pecora mobile application guideline created by Svein-Olaf Hvasshovd.



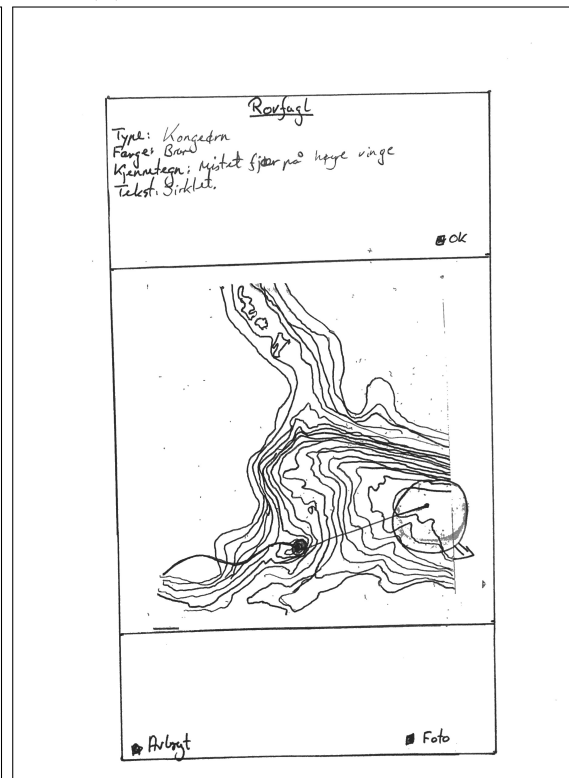
(c) Register observation.



(d) Observation: ewe with lamb.



(e) Observation: lamb alone.



(f) Observation: golden eagle.

Figure A.1: Pecora mobile application guideline created by Svein-Olaf Hvasshovd.

B Code

This appendix contains relevant code snippets from the projects that were mentioned in the implementation section (Section 7.2).

B.1 Github Repository Links

- Mobile application repository: <https://github.com/fridamsh/pecora>.
- Web application repository: <https://github.com/fridamsh/pecora-web>.

B.2 Server APIs

Listing B.1: The server API `loginUser.php` for login in the app.

```
<?php
include_once '../includes/dbh.inc.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $uid = mysqli_real_escape_string($conn, $_POST['username']);
    $pwd = mysqli_real_escape_string($conn, $_POST['password']);

    if (empty($uid) || empty($pwd)) {
        $json['error'] = 'E1'; //E1 - empty username or password
        echo json_encode($json);
        mysqli_close($conn);
    } else {
        $sql = "SELECT * FROM users WHERE user_uid='$uid' OR user_email='$uid';"
            ;
        $result = mysqli_query($conn, $sql);
        $resultCheck = mysqli_num_rows($result);
        if ($resultCheck < 1) {
            $json['error'] = 'E2'; //E2 - user does not exist
            echo json_encode($json);
            mysqli_close($conn);
        } else {
            if ($row = mysqli_fetch_assoc($result)) {
                $hashedPwdChecked = password_verify($pwd, $row['user_pwd']);
                if ($hashedPwdChecked == false) {
                    $json['error'] = 'E3'; //E3 - wrong password
                    echo json_encode($json);
                    mysqli_close($conn);
                } elseif ($hashedPwdChecked == true) { //Login user and echo
                    user data
                }
            }
        }
    }
}
```

```

        $json['success'] = 'success';
        $json['id'] = $row['user_id'];
        $json['first'] = $row['user_first'];
        $json['last'] = $row['user_last'];
        $json['email'] = $row['user_email'];
        $json['username'] = $row['user_uid'];
        echo json_encode($json);
        mysqli_close($conn);
    }
}
}
}
} else {
    $json['error'] = 'E4'; //Error
    echo json_encode($json);
    mysqli_close($conn);
}
}

```

Listing B.2: The server API registerUser.php for registration in the app.

```

<?php
include_once '../includes/dbh.inc.php';

if ($_SERVER["REQUEST_METHOD"] == "POST") {
    $first = mysqli_real_escape_string($conn, $_POST['first']);
    $last = mysqli_real_escape_string($conn, $_POST['last']);
    $email = mysqli_real_escape_string($conn, $_POST['email']);
    $uid = mysqli_real_escape_string($conn, $_POST['uid']);
    $pwd = mysqli_real_escape_string($conn, $_POST['pwd']);

    if (empty($first) || empty($last) || empty($email) || empty($uid) || empty(
        $pwd)) {
        $json['error'] = 'E1'; //E1 - empty fields
        echo json_encode($json);
        mysqli_close($conn);
    } else {
        if (!preg_match("/^[a-zA-Z]*$/", $first) || !preg_match("/^[a-zA-Z]*$/",
            $last)) {
            $json['error'] = 'E2'; //E2 - not valid first and lastname
            echo json_encode($json);
            mysqli_close($conn);
        } else {
            if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
                $json['error'] = 'E3'; //E3 - email not valid
                echo json_encode($json);
                mysqli_close($conn);
            } else {
                $sql = "SELECT * FROM users WHERE user_uid='$uid'";
            }
        }
    }
}

```



```

        $result = mysqli_query($conn, $sql);
        $resultCheck = mysqli_num_rows($result);
        if ($resultCheck > 0) {
            $json['error'] = 'E4'; //E4 - user taken
            echo json_encode($json);
            mysqli_close($conn);
        } else {
            $hashedPwd = password_hash($pwd, PASSWORD_DEFAULT);
            $sql = "INSERT INTO users (user_first, user_last, user_email,
                user_uid, user_pwd) VALUES ('$first', '$last', '$email',
                '$uid', '$hashedPwd')";
            $result = mysqli_query($conn, $sql);
            if ($result == 1) {
                $json['success'] = 'Inserted user';
            } else {
                $json['error'] = 'E5'; //E5 - insert error
            }
            echo json_encode($json);
            mysqli_close($conn);
        }
    }
}
} else {
    $json['error'] = 'E5'; //Error
    echo json_encode($json);
    mysqli_close($conn);
}
}

```

Listing B.3: The PHP file login.inc.php.

```

<?php
session_start();

if (isset($_POST['submit'])) {
    include 'dbh.inc.php';
    $uid = mysqli_real_escape_string($conn, $_POST['uid']);
    $pwd = mysqli_real_escape_string($conn, $_POST['pwd']);
    //Error handlers
    //Check if inputs are empty
    if (empty($uid) || empty($pwd)) {
        header("Location: ../login.php?login=empty");
        exit();
    } else {
        $sql = "SELECT * FROM users WHERE user_uid='$uid'";
        $result = mysqli_query($conn, $sql);
        $resultCheck = mysqli_num_rows($result);
        if ($resultCheck < 1) {

```

```

        header("Location: ../login.php?login=error");
        exit();
    } else {
        if ($row = mysqli_fetch_assoc($result)) {
            //De-hashing the password
            $hashedPwdChecked = password_verify($pwd, $row['user_pwd']);
            if ($hashedPwdChecked == false) {
                header("Location: ../login.php?login=error");
                exit();
            } elseif ($hashedPwdChecked == true) {
                //Log in the user here
                $_SESSION['u_id'] = $row['user_id'];
                $_SESSION['u_first'] = $row['user_first'];
                $_SESSION['u_last'] = $row['user_last'];
                $_SESSION['u_email'] = $row['user_email'];
                $_SESSION['u_uid'] = $row['user_uid'];
                header("Location: ../index.php?login=success");
                exit();
            }
        }
    }
} else {
    header("Location: ../login.php?login=error");
    exit();
}

```

Listing B.4: The showHikeOnMap() method in script.js.

```

function showHikeOnMap(hike) {
    var id = hike.id;
    var title = hike.title;
    var name = hike.name;
    var participants = hike.participants;
    var weather = hike.weather;
    var description = hike.description;
    var startdate = hike.startdate;
    var dateStart = new Date(Number(startdate));
    var enddate = hike.enddate;
    var dateEnd = new Date(Number(enddate));
    var mapfile = hike.mapfile;
    var distance = hike.distance;
    var userId = hike.userId;
    var localId = hike.localId;
    var observationPoints = hike.observationPoints;
    var track = hike.track;

    mapItems = [];
}

```

```

var polylineColor = createRandomColor();

//Decode observation points
var jsonObservationPoints = JSON.parse(observationPoints);
var totalSheepCount=0;
for (var i = 0; i < jsonObservationPoints.length; i++) {
    var latitude = Number(jsonObservationPoints[i].locationPoint.mLatitude);
    var longitude = Number(jsonObservationPoints[i].locationPoint.mLongitude);
    var pointParent = new L.LatLng(latitude, longitude);
    var date = new Date(Number(jsonObservationPoints[i].timeOfObservationPoint));
    var observationList = jsonObservationPoints[i].observationList;
    var marker = L.marker(pointParent, {icon: observationPointIcon});
    marker.setOpacity(0.5);
    if (observationList.length == 1) {
        marker.bindPopup("<b>Punkt "+jsonObservationPoints[i].pointId+" kl. "+date.format("HH:MM")+ "</b><br><i>"+observationList.length+" observasjon</i>"+ "<br><b>Sau sett:</b> "+jsonObservationPoints[i].sheepCount);
    } else {
        marker.bindPopup("<b>Punkt "+jsonObservationPoints[i].pointId+" kl. "+date.format("HH:MM")+ "</b><br><i>"+observationList.length+" observasjoner</i>"+ "<br><b>Sau sett:</b> "+jsonObservationPoints[i].sheepCount);
    }
    mapItems.push(marker);
    totalSheepCount+=Number(jsonObservationPoints[i].sheepCount);
    //Decode observations
    for (var j = 0; j < observationList.length; j++) {
        var latitude = Number(observationList[j].locationObservation.mLatitude);
        var longitude = Number(observationList[j].locationObservation.mLongitude);
        var pointChild = new L.LatLng(latitude, longitude);
        // Make marker for observation
        var marker = L.marker(pointChild, {icon: blueIcon});
        if (observationList[j].typeOfObservation == 'Sau') {
            marker.bindPopup("<b>Observasjon "+observationList[j].observationId+ "</b><br><b>Type:</b> "+observationList[j].typeOfObservation+ "<br><b>Antall:</b> "+observationList[j].sheepCount+ "<br><b>Detaljer:</b> "+observationList[j].details);
        } else {
            marker.bindPopup("<b>Observasjon "+observationList[j].observationId+ "</b><br><b>Type:</b> "+observationList[j].typeOfObservation+

```

```

        "<br><b>Detaljer:</b> "+observationList[j].details);
    }
    mapItems.push(marker);
    // Add polyline between observation point and observation
    var line = new L.Polyline([pointParent,pointChild], {
        color: polylineColor,
        weight: 3,
        opacity: 0.4,
        smoothFactor: 1
    }).bindPopup('<b>Punkt '+jsonObservationPoints[i].pointId+' kl. '+
        date.format("HH:MM")+
        '</b><br><i>Observasjon '+observationList[j].observationId+
        '</i><br><b>Type:</b> '+observationList[j].typeOfObservation+
        '<br><b>Sau sett:</b> '+observationList[j].sheepCount);
    mapItems.push(line);
}
}

//Decode track
trackPointList = [];
var jsonTrack = JSON.parse(track);
for (var i = 0; i < jsonTrack.length; i++) {
    var latitude = Number(jsonTrack[i].mLatitude);
    var longitude = Number(jsonTrack[i].mLongitude);
    var point = new L.LatLng(latitude, longitude);
    trackPointList.push(point);
}

var markerStart = L.marker(trackPointList[0], {icon: startIcon});
markerStart.bindPopup("<b>Start</b> "+dateStart.format("HH:MM"));
mapItems.push(markerStart);
var markerEnd = L.marker(trackPointList[trackPointList.length-1], {icon:
    stopIcon});
markerEnd.bindPopup("<b>Slutt</b> "+dateEnd.format("HH:MM"));
mapItems.push(markerEnd);

var trackPolyline = new L.Polyline(trackPointList, {
    color: polylineColor,
    weight: 4,
    opacity: 0.9,
    smoothFactor: 1
}).bindPopup('<b>'+title+'</b><br>'+dateStart.format("dd/mm/yyyy HH:MM")+
    '+dateEnd.format('HH:MM')+
    '<br><b>Gjeter:</b> '+name+
    '<br><b>Deltakere:</b> '+participants+
    '<br><b>Antall sau sett:</b> '+totalSheepCount+
    '<br><b>Vaer:</b> '+weather+

```

```

    '<br><b>Distanse:</b> '+distance+' km'+
    '<br><b>Detaljer:</b> '+description);
mapItems.push(trackPolyline);
mymap.fitBounds(trackPolyline.getBounds());

//Add all map items to the map
var layer2 = L.layerGroup(mapItems);
layer2.id = id;
layer.addLayer(layer2);
}

```

Listing B.5: Click listener for report button in script.js.

```

$('#reportBtn').on('click', function() {
  // Default export is a4 paper, portrait, using milimeters for units
  $.ajax({
    url: "includes/get-report-information.inc.php",
    type: "GET",
    success: function (data) {
      //Parse data
      var obj = JSON.parse(data);
      if (obj.error == 'error') {
        alert("Ingen turer aa vise i rapport");
      } else {
        //Create PDF
        var doc = new jsPDF();
        pageHeight= doc.internal.pageSize.height;
        //Set date, title and icon
        doc.setFontSize(12);
        var dateNow = new Date();
        var dateNowFormatted = dateNow.format("dd/mm/yyyy");
        doc.text(195, 15, dateNowFormatted, null, null, 'right');
        doc.setFontSize(20);
        doc.text(105, 25, 'Pecora', null, null, 'center');
        doc.setFontSize(22);
        doc.setFontType('bold');
        doc.text(105, 37, 'Generert Rapport', null, null, 'center');
        doc.addImage(imgData, 'PNG', 95, 44, 20, 20);
        //Page content
        doc.setFontType('normal');
        doc.setFontSize(14);
        doc.text('Beitelag: Ukjent', 15, 75);
        doc.text('Beiteaar: '+dateNow.format('yyyy'), 15, 85);
        doc.text('Tilsynsperson: '+name+' '+lastname, 15, 95);
        var lineUnit = 110;
        //Loop through hikes list
        for (var i = 0; i < obj.length; i++) {
          var startdate = obj[i].startdate;

```

```
        var enddate = obj[i].enddate;
        var description = obj[i].description;
        var observationPoints = obj[i].observationPoints;
        //Decode observation points
        var jsonObservationPoints = JSON.parse(observationPoints);
        var totalSheepCount = 0;
        for (var j = 0; j < jsonObservationPoints.length; j++) {
            totalSheepCount += Number(jsonObservationPoints[j].
                sheepCount);
        }
        var dateStart = new Date(Number(startdate));
        var dateEnd = new Date(Number(enddate));
        //Check if a new page must be added
        if (lineUnit+30 >= pageHeight) {
            doc.addPage();
            lineUnit = 20 // Restart height position
        }
        //Add hike data to PDF
        doc.setFontType('bold');
        doc.text('Dato: '+dateStart.format("dd/mm/yyyy")+
            'Beskrivelse: '+description, 15, lineUnit);
        lineUnit+=10;
        doc.setFontType('normal');
        doc.text('Antall sau sett: '+totalSheepCount, 15, lineUnit);
        lineUnit+=20;
    }
    //Download PDF
    var dateNowFormattedDash = dateNow.format("dd-mm-yyyy");
    var pdfName = 'report-'+dateNowFormattedDash+'.pdf';
    doc.save(pdfName);
}
},
error: function(xhr, ajaxOptions, thrownError){
    alert("AJAX Error");
},
timeout: 15000
});
});
```

C Interview 1: Steingrim Horvli

This appendix contains the interview with the sheep farmer Steingrim Horvli who we visited in Oppdal on April 9, 2018. The meeting was audiotaped and later transcribed. Since the interview was conducted in Norwegian, it will be presented in its original form and will not be translated. The summary of the interview is found in Section 8.1.1.

C.1 Roles

The roles in this interview are presented in the following list:

- A = Andreas Kjerstad,
- S = Steingrim Horvli,
- SO = Svein-Olaf Hvasshovd, and
- F = Frida Schmidt-Hanssen.

C.2 Interview

A: Hvilke krav er det til oppfølging av sau, og hvordan foregår dette?

S: For utmarksbeite, er kravet tilsyn minimum én gang pr uke. Når det skjer et eller annet, så skal det økes.

A: Hva kan eksempelvis skje?

S: Et eller annet angrep, eller at du ser det er noe som ikke stemmer i beiteområdet.

SO: Så Steingrim, du sier en gang i uka?

S: Det er minimum én gang pr uke.

SO: Men er det hver eneste sau du skal følge opp én gang i uka?

S: Det har du ikke sjans til. Du skal prøve å få sjekka mesteparten.

SO: Mesteparten, akkurat. Så hvis du har sau som er spredt over fire områder, så er det klart at du har ikke sjans til å nå over dem én gang i uka.

S: Nei, men så fungerer det slik at vi har organisert tilsyn. Det er utplukket en del personer, mer som en vaktuke. De skal ha minimum fire dager pr uke. De utfyller de områdene vi ikke klarer å sjekke ut. Det kommer i tillegg til den ene dagen vi er pålagt å være ute for å sjekke dyra.

SO: Så det er for å dekke opp et stort område?

S: Det er for å dekke opp et stort område.

SO: Den siste biten her var jeg ikke klar over. Det er vel ikke alle som har det, er det?

S: Stort sett alle områder som er utsatt for rovdyr har det.

SO: Okay. Nei, jeg kan ikke huske at Hallvard hadde det.

S: Jo, dette har vi hatt i alle fall i 15 år.

A: Hvilke utfordringer eksisterer i dag i forbindelse med tilsyn av sau?

S: Utfordringa er å finne dem.

SO: Har du noen triks for å finne dem utover å gå, og prøve på de stedene der du normalt forventer å finne dem?

S: Vi har den radiobjellen vi bruker, for å finne ut hvor de er i terrenget. Så bruker vi dette som utgangspunkt.

A: Hva er hovedårsaken til tap av sau på beite?

S: Det er rovdyr her i området. For noen år siden, da det ikke var rovdyr... mange år fikk vi inn igjen alle dyra.

A: Hvor mange blir typisk tatt i løp av en sesong?

S: Forrige sesong så mistet jeg 90 dyr av 600.

F: Hva er de typiske rovdyrene?

S: I Oppdal nå, så er det jerv, gaupe og ørn.

SO: Har du noen formening om fordeling mellom dem?

S: Nei, det er vanskelig, de har så forskjellige måter å gjøre det på. Kongeørna er typisk tidlig på våren. Da kommer det bort. Gaupen gjemmer det godt, så da er det vanskelig å finne igjen. Jerven er kanskje det enkleste å finne igjen. Han er stort sett i høyfjellet og du klarer å se det på avstand.

SO: Av de 90 som du tapte, er det mest jerv?

S: Problemet er at vi ikke fant igjen så mange, men vi mistenker gaupa i fjor. Vi fikk et ganske stort inntrykk av gaupe.

A: Er det noe samarbeid mellom bønder når det gjelder registrering av sau?

S: Jada, det er organisert alt det er.

A: Hvordan foregår det?

S: Det er, som jeg nevnte, den vaktplanen og dyrene holder til i samme område, så når du går en tilsynstur så sjekker du ikke bar dine, du sjekker alle. De er kodemerket, slik at vi vet hvor mange lam hvert dyr skal ha. Hvis jeg treffer nabeons sauer så ser jeg den kodemerkinga, og vet at det dyret skal ha så og så mange lam.

A: Hva slags kodemerking er det?

S: Vi har fargekoding på bjelle.

SO: Du har det på bjella, ikke på slipset?

S: Begge deler, men det henger i bjella alt det her. Da har vi en farge for ett lam, en farge for to lam, og en farge for tre lam. Det er felles for hele Oppdal. De gjør dette andre steder i Norge også, men de bruker ikke samkjørte farger på kodene. Så andre plasser kan de ha andre farger for ett, to og tre lam da.

SO: men innenfor Oppdal så er det én (farge)?

S: Ja, innenfor Oppdal er det én. Men hvis du for eksempel kommer til Rennebu, så kan det være de har noe annet.

A: Finnes det noen utfordringer med deling av informasjon mellom bønder i dag, når det gjelder samarbeid?

S: Nei, egentlig ikke. Det fungerer.

A: Hvordan foregår informasjonsinnsamlingen når du gjennomfører en slik tur? Hva slags teknologi som blir brukt, osv.

S: Tidligere har det bare blitt notert i ei bok, hva vi har gjort om dagene, penn og papir. Men her i fjor tok vi i bruk BeiteSnap, som det kalles. Den fungerte til en viss grad.

SO: Kan du si litt om hvordan den fungerer? Er det en tekstlig beskrivelse, eller noe du fyller ut?

S: Den fungerer slik at dersom jeg går ut på en tilsynstur så startet jeg logging på den, den logger hele turen. Hvis man ser et eller annet, så tar man et bilde. Dersom det er min, så legger jeg den bare inn i logging. Dersom det er andre sine dyr så legger jeg den ut, og alle som bruker BeiteSnap får den opp på telefonen sin. Det finnes to utgaver av den. En for oss brukere, og en for den vanlige farende fant, og det er en gratisversjon som man kan laste ned. Det er sånn av da jeg la inn den så markerte jeg området dyra mine går på, så alt som skjer utenfor det området det får ikke jeg opp på telefonen min. Men alt som skjer innenfor det kommer inn på telefonen min.

A: Så dersom sauene dine går utenfor dette området så får du ikke informasjon om dem?

S: Nei, men derfor har jeg tegna et ganske stort område. Den skal egentlig ikke gå utenfor.

A: Blir det her brukt av mange turgåere?

S: Det var første året i fjor, det var i bruk. Det var noen, men ikke så mange. Det har kanskje litt med informasjonen ut til publikum her da.

SO: Jeg var ikke klar over den.

S: Nei, den kom ut i fjor.

A: Hva slags informasjon er det som blir registrert i dag?

S: Nei, det er egentlig alt som vi ser. Dersom alt er normalt, så blir det også registrert.

A: Er det noe informasjon som er påbudt av myndighetene å registrere?

S: De har påbudt alt, tror jeg. De skal ha mest mulig informasjon. De skal vite når alt skjer, lokasjon osv. De skal vite hva vi har på utmarksbeite.

A: Er det noen formelle retningslinjer for hva som skal registreres?

S: Nei.

SO: Hvem er det som vil ha denne informasjonen?

S: Mattilsynet og fylkesmannen.

SO: Hvem er det dere sender inn til?

S: Vi sender inn til fylkesmannen.

A: På hvilket format sendes informasjonen inn?

S: Det er ferdige utfylte skjema. Elektroniske skjema som fylkesmannen har. Det er der all informasjonen går ut. All informasjonen som går til fylkesmannen er samla. Vi er et lag som operer i et område, så blir all informasjonen samla inn i det laget og sendt samla. Vi har en samlingsrunde før den datoen, der alle kommer med op-

plysningene sine, og så blir det send videre.

SO: Hvis vi skal prate med fylkesmannen om dette her, har du noe navn?

S: Nei, akkurat no har jeg ikke det, for etter sammenslåinga av fylkene så er det nye folk, så jeg aner ikke.

SO: Har du noe navn fra tidligere?

S: Vi har en... hva er det han heter da? Jeg kommer ikke på navnet hans.

SO: Da er det bare for oss å kontakte fylkesmannen og spørre om dette her.

S: Jeg aner ikke hvem som styrer dette her nå.

A: Det skjemaet, er det noe du laster ned og fyller ut? Eller fylles det ut på nettet?

S: Det er et standard for de lagene.

A: Er dette noe vi kan få tilgang til?

S: Kanskje.

SO: Da er det naturlig at vi kontakter fylkesmannen og spør pent om å få tilgang til det.

S: Det er jo mer som en årsrapport for området dette. Problemet er at det ligger en del personlige opplysninger, siden det er ferdigutfyllt av oss, med personopplysninger og alt.

SO: Hvor mange gårder er det som er sammen om dette her?

S: Det er både sau og storfe.

SO: Er det hele søndre Trondheimen?

S: Ja.

A: Så alle dere sender én felles rapport?

S: Ja. Når vi kommer til årets slutt, og skal søke om erstatning, og den biten der, så må du inn med mer utfyllende opplysninger. Da er det hver enkelt.

A: Når du er ute og registrer, hvordan er vanligvis mobildekningen i området?

S: Stort sett er det mobildekning

A: Er det viktig å registrere hvor du går, når du gjennomfører en slik runde?

S: Det er viktig for oss selv hvor vi har gått. Det er vel ikke så interessant for andre foreløpig. Jeg ser ikke bort fra at det kravet kommer.

A: Hvor mange timer i uka vil du anslå at du bruker på å gå tilsynsturer?

S: Det er varierende. Det avhenger av vært, i tillegg har du oppgaver hjemme du skal utføre også. I snitt tror jeg at jeg bruker én og en halv dag. En dag skal vi regne 7,5 timer på, så 10-12 timer i uka bruker jeg nok på tilsyn.

SO: Bruker du mer tid enn det andre gjør, eller ligger de fleste på omtrent det samme?

S: Jeg tror det er jevnt over ganske bra med tilsyn. Men det varierer litt, noen leier jo hjelp for å gjøre tilsyn for seg og noen gjør det selv.

SO: Jeg spør fordi jeg har forstått at Hallvard brukte mye mindre tid på det. Men du må gå?

S: Jeg har ikke noe vei i områdene, så jeg må gå.

A: Vet du hvordan informasjonen som sendes inn til myndighetene blir brukt?

S: Nei. Det har jeg ikke noen anelse om.

A: Blir det noen reaksjoner dersom sauene ikke blir fulgt opp tilstrekkelig?

S: Da blir det reaksjoner, ja.

A: Vet du hvilke reaksjoner?

S: Nei, det har jeg ikke prøvd ut enda, men Mattilsynet er fort inne i bildet.

SO: Har du hørt om noen som har fått slike reaksjoner?

S: Nei, jeg har ikke det.

A: Du nevnte at du har tatt i bruk BeiteSnap, har du testet noen andre løsninger?

S: Jeg vet ikke om det er noe annet på markedet.

A: Det finnes løsninger med GPS-tracking, er dette noe som blir brukt?

S: Ja, jeg bruker det mye. Det er den radiobjella som er. Da får vi sporet hvor de er hele tiden, hvor de er i terrenget. Jeg lurte på om jeg har en 40 slike. Har det på 40 dyr. Problemet er prisen på dem. Det er en forholdsvis høy innkjøpspris, og så er det en forholdsvis høy årspris.

A: Vet du hva den prisen ligger på?

S: Bjella ligger på 1700-1800, og årsabonnementet pr bjelle ligger på rundt 200 kroner.

A: Hvilke fordeler ser du med å bruke slike hjelpemidler som radiobjelle og BeiteSnap?

S: Radiobjella er en nødvendighet, etter at man har begynt å bruke det. Man har en helt annen kontroll. Den gir beskjed så fort det skjer noe med dyrene. Dersom det er sykdom så får man beskjed.

A: Hvordan detekteres dette?

S: Det er bevegelse. Når dyrene blir dårlige så beveger de seg lite, og det kommer en beskjed om at de bruker så og så lite område at da må vi være oppmerksomme på at det kan være noe galt.

A: Når det gjelder BeiteSnap da?

S: Den må jeg si at jeg ikke har så mye erfaring med enda, men den fungerer, så jeg tror det kan bli et godt hjelpemiddel.

A: Er det noen mangler du ser med BeiteSnap?

S: Problemet i fjor var at det datt ut for oss, så vi fikk ikke fullført loggen. Kanskje den stoppa etter ei periode og da var den dagen egentlig vekkasta. Når det fungerer er det kjempegreier.

SO: Har du noen formening om årsaken til at det datt ut?

S: Nei, det var et problem de hadde, hva som forårsaket det vet jeg ikke.

SO: Er det et privat firma som driver dette?

S: Ja, jeg tror det er et privat firma.

F: Krevde den appen at man hadde mobilnett?

S: Nei. Du kunne gjøre alt med den når du var utenfor dekning, og så fort du hadde dekning, så gikk det ut. Kartet var nedlastet.

SO: Var det mange av dere som brukte dette?

S: I laget vårt så var vi vel en fire-fem, tror jeg.

A: Da tror jeg vi har kommet gjennom spørsmålene jeg har her.

SO: Er det noe du vil fortelle oss, Steingrim, som vi burde ha spurt om?

S: Nei, jeg tror egentlig vi har fått det med oss. Det er jo klart det er store utfordringer med dette her da.

F: Var det slik at BeiteSnap hadde en kombinasjon av at man kan registrere ting man så og den GPS-sporingen med radiobjella?

S: Ikke radiobjella, de er to forskjellige ting, så det går ikke på radiobjella, nei.

SO: Det du kunne registrere var bilder?

S: Ja, og tekst.

F: Er det ofte slik at du observerer sauene fra avstand?

S: Ja det er mye av det vi gjør. Kikkert er noe av det viktigste vi har med oss. Det er et av det viktigste redskapene man har.

F: For eksempel når du tar bilder, er det når du finner noe som er nærmere?

S: Det er stort sett hvis det er et eller annet som har skjedd. Døde dyr, syke dyr, eller dyr som mangler lam. Da prøver jeg å få tatt bildet av nummeret på det dyret.

SO: Du ser på det på fargen på slipset? Det skulle vært tre lam, men det er bare to, så er det åpenbart at her mangler det noe.

S: Ja, så da må vi prøve å få notert det, og finne ut hvorfor det ikke er der.

SO: Er manglende lam noe av det mest vanlige du ser?

S: Ja. Det er da vi begynner å få mistanke om at noe har skjedd og begynner å sette inn ekstra ressurser i det området.

SO: Det er jo klart at lammet kan ha byttet mamma.

S: Det kan det godt. Det kan være sammen med andre dyr. Det må registreres dyr som har med seg for mange lam også.

SO: Jeg har vært en del ute og fulgt opp sau, og da må jeg si at erfaringsmessig er det vanskelig å skille lam og voksne dyr. I alle fall utpå året.

A: Er det noe merking av lam, som gjør at de skiller seg fra voksne?

S: Ikke noe annet enn det første nummeret i øreklypa, den indikerer fødselsår.

SO: Men å få sett det tallet er ikke noe du får gjort.

S: Det er ikke noe lett å se det, nei.

SO: Ville det i ...

D Interview 2: The County Governor

This appendix contains the interview with the County Governor of Trøndelag who we met in Trondheim April 24, 2018. Eva Dybwad Alstad and Arnstein Lyngstad represented the County Governor. The meeting was audiotaped and later transcribed. Since the interview was in Norwegian, it will be presented in its original form and will not be translated. The summary of the interview is found in Section 8.1.2.

D.1 Roles

The roles in this interview are presented in the following list:

- SD = Stian Dysthe,
- E = Eva Dybwag Alstad,
- AL = Arnstein Lyngstad,
- SO = Svein-Olaf Hvasshovd,
- A = Andreas Kjerstad, and
- F = Frida Schmidt-Hanssen.

D.2 Interview

SD: Hva er det som er deres rolle i forbindelse med oppfølging av sau?

E: Vi forvalter tilskuddsordninger, som er aktuelt for sauebrukere å søke på. Den har vi, den ordningen for tiltak i beiteområder, som jeg snakket om, det er jo investeringsmidler eller planlegging og utprøvningsmidler, for eksempel å prøve ut nytt utstyr. Der er det vi som får til den ordningen. Så har vi og en ordning som heter organisert beitebruk eller drift av beitelag. Vi har en tilskuddordning gjennom..

AL: *[Unclear]*

E: Der er det midler som går til beitelagene per dyr dem har, som de sanker på beitet.

AL: Er det dyr som dem har i organisert beitebruk?

E: Ja, dyr på utmarksbeite. Det er i forhold til tilskudd.

AL: Også er det produksjonstilskudd som vi også forvalter, og det er kommunene da. Og ved klageinstans er det og tilskudd per dyr. Også har vi noe som heter SMIL-tilskudd, som er en rydding av beite så det blir innmarksbeite. Det går til sauebrukerne da, kan du si. Da tror jeg ikke det er flere tilskuddsordninger. Også er vi involvert i Miljøvern avdelingen sitt forebyggende, konflikt-dempende tiltak og erstatning for tap av sau og dyr med tanke på rovvilt.

SD: Blir det da sendt inn rapporter til dere? For å få erstatning? Altså, søknad

om erstatning.

AL: Ja, Fylkesmannen, ja. Som Miljøvernavdelingen, de får sånne rapporter.

E: Søknaden kommer dit. Så, det vi og gjør, rapportene som kommer gjennom organisert beitebruk er når de søker på det tilskuddet så må de rapportere "Hvor mange dyr har vi hatt på beite? Hvor mye tid har vi brukt på tilsyn?" og sånne ting. Da kommer det ut rapporter, og da snakker vi med Miljøvernavdelingen, slik at de kan bruke det som dokumentasjon for å beregne tilskuddet eller for å beregne erstatningen.

AL: Det tilskuddet de får, organisert beitebruk, gjennom det tilskuddet der er per sanket dyr.

E: Det er jo det vi gjør i forhold til penger, men vi har jo kontakt gjennom brukerne våres, kontakt med Mattilsynet, som har ansvaret for det med dyrevelferd og sånt. Og ...

AL: Kommunene.

E: Kommunene. Også organisasjonene har vi kontakt med. For eksempel når de skal begynne å prioritere det her med tilskudd når de har veldig mange søkere så har vi en dialog med næringsorganisasjonene, da. Geit og bondelag..

AL: Sånn at næringen får være med oss å påvirke hvordan vi prioriterer en del midler.

E: Og å komme med innspill på hva som er riktig.

SD: Spørsmål to svarte dere litt på, men hvilke andre myndigheter er altså involvert i oppfølging av sau og hva er deres roller? Som Mattilsynet, hva er det deres ansvar er?

AL: Dyrevelferd.

E: Og dyresykdommer og dyrehelse.

AL: Unngå at det blir spredning av sykdom.

SD: Men er de interessert i innrapportering av oppfølging av sauene? For eksempel, er de interessert i hvor sauene går om sommeren osv.?

E: Ja, de er interessert. Listene vi får på organisert beitebruk, det bruker vi å sende til Mattilsynet og vi forteller dem om det. Så de får rapportene. Og Mattilsynet jobber mer opp mot hver enkelt forbruker og går på tilsyn der, og da kan de jo se på sånne ting. Mattilsynet er og med og sitter på møtene til Rovviltnemnda og er interessert i hvordan man forvalter rovdirene. Og hvis det er slik at det er en veldig akutt situasjon i et område, veldig mye bjørn på beite for eksempel, så har Mattilsynet myndighet til å pålegge bonden å sanke sauene hjem. Tidlig nedsanking. De har noen regelverk som sier hva sauebrukeren skal gjøre med sauene sine for å unngå tap.

AL: Det kan både skje gjennom pålegg fra Mattilsynet eller det kan være frivillig, og at man får tilskudd til det for å forebygge. Ellers så kan det nevnes en annen plass de rapporterer til, til sauekontrollen, deres eget system, der man rapporterer inn viktig data om sauene iløpet av sesongen. Det er en database de bygger opp, der de kan kjøre sammenligning mellom besetningen osv. for eksempel.

E: Det kunne vært interessant i forhold til appen deres, om den kunne kommunisere med det systemet der.

SD: Hvilke krav er det til oppfølging av sau i dag?

AL: Vi stiller jo krav gjennom organisert beitebruk.

E: Der er det krav om at de skal dokumentere antall dagsverk tilsyn per uke. Det er det kravet i forhold til organisert beitebruk.

AL: Også har jo Mattilsynet, eller matloven som er regelverket Mattilsynet forvalter, noen minimumskrav for tilsyn oppført i den. Også har vi da krav for å få tilskudd.

SD: Men det er en gang i uka, er det det som er minimumskravet?

E: Ja, det står det i forskriften om velferd for småfe; ”Dyr som holdes på utmark skal sees etter minst en gang per uke i områder uten særskilt risiko”. Også er det og et krav om at hvis det kan være risiko så må dem ha mer tilsyn.

SD: Blir sauene fulgt godt nok opp i dag?

E: Da må du spørre Mattilsynet.

AL: Jeg tror nok mange gjør en stor innsats, men så kan det være vanskelig i store uoversiktlige beiteområder, spesielt hvis det er kronglete terreng, mye vegetasjon. Så selv om man gjør den tilsynsjobben så kan det være krevende å få sett nok dyr og få god nok oversikt. Og med dagens utvikling så ser man at besetningsstørrelsen øker, antall sauebønder blir færre, og det betyr og at når man skal organisere tilsynet så har du færre personer å spille på. Så tilsynsoppgaven blir enda mer krevende, flere dyr og når du da i tillegg har en del terreng som er uoversiktlig så ser vi at det er en utfordring å få tilsynet effektivt nok.

SO: Du sier at dere forlanger tilsyn en gang i uka, men skal man da se hver enkelt sau? Eller hvor nøye skal man være?

E: Det kravet står ikke, at du skal se hver enkelt. Det står det ikke, og det er ikke særlig realistisk hvis du har de på utmarksbeite. Men det her ville jeg egentlig snakket med Mattilsynet om, hvordan de håndterer det.

AL: Det står ikke spesifisert hvor mange dyr du må se i regelverket. Det kan være stor forskjell, noen kan jo se mange av dyrene og få god oversikt, men så kan det og være noen som ikke får sett så mye dyr og ikke har like godt oversikt.

E: Stor forskjell på skogsbeite og det å klare å ha oversikten der med mye skog, og terrenget går opp og ned.

SO: Jeg ser jo bare i Oppdalsområdet.. Du rekker ikke å gå over området sauene er på en dag, ikke en sjans.

AL: Det ser vi jo at radiobjellene har vært med på å forbedre tilsynet en del, med at dem da henger på sendere på en del av sauene også får du et kart opp også kan du måle tilsynet. Det ser vi har gitt forbedring, så teknologi det er interessant.

SD: Er det interessant å se hvor bøndene har gått når de har gjort tilsynsturen?

E: Ja, det tror jeg kan være lurt for å planlegge neste tilsynstur.

SD: Okei, men det er ikke slik at dere er interessert i å se det?

E: Nei, jeg tror man heller må tenke først og fremst hva beitebrukerne har behov for.

AL: Vi trenger ikke å gå inn og se på det. Men det er jo en veldig bra måte for næringen for å få mer målretta tilsyn, når man har gode systemer for å loggføre hvor du har gått. Og hvis man i tillegg har sendere på en del av dyra, så kan du se hvor dem har gått, da begynner vi å nærme oss en stor forbedring.

SO: Så du ser faktisk den serteringen her som supplement til den andre, du?

AL: Jeg tenker det kan utfylle, og det kunne vært interessant med noen som faktisk kunne prøvd.

SD: De rapportene som sendes inn i dag, hvilken form blir de sendt på? Er det skjema på internett?

E: Mhm, for organisert beitebruk er det et eget system eStil, heter det, som man legger inn og rapporterer inn på. Og så et på produksjonstilskudd, der skal du også sette inn hvor mange dyr du har på utmarksbeite. Der kommer det et nytt system nå.

AL: Ja, men det skal i hvert fall settes inn.

SD: Men er det standardisert på noen måte, hvordan det rapporteres?

E: Mhm. Det er det.

AL: De skriver jo inn når de søker tilskudd, så skriver de hvor mange dyr de har, hvor mye dyr de har sluppet og sankt. Så da får jo vi laget en statistikk, og vi har jo ganske mye statistikk på dataen våres. Det er utifra søkingen på tilskudd.

SD: Hvor ofte får dere inn rapporter?

AL: Det er jo per gang de søker tilskudd, også lages det rapporter. Når er det vi får dem?

E: For organisert beitebruk så er det en gang i året, der de melder inn til 1. november og da bruker rapportene å være klar.. Og der skal kommunene gå inn og se at det er rett, det de har skrevet. Den endelige rapporten pleier å være klar til jul. Jeg har hatt litt datatekniske problemer på de rapportene nå på grunn av sammenslåingen, så jeg får faktisk ikke ut en sånn rapport jeg har brukt å få ut før i Trøndelag. Men jeg har fått ut en rådatafil, så må jeg sitte å bearbeide den selv. Håper at det blir bedre til neste år.

AL: Men det er jo faktisk for både selve tallet på antall dyr, men så er det jo en kartfesting for hvor beitelagene er også. Og der kan man gå inn på MIS-verktøyet og velge kartlag med beitebruk og få opp hvor du har de forskjellige beitelagene og hvor mye dyr de har osv. Det er et brukbart system altså. Det er en video som har det, du kan gå inn på der.

E: Kilden. Der kan du krysse av på beitelag og da får du frem beitelagsgrensene, også får du frem statistikk for de siste årene.

AL: Så, det beitelaget dere eventuelt skal samarbeide med, da kan man gå inn der og se på dataene på det.

SO: Jeg har faktisk sett en del av de dataene, men jeg tror det er papirform jeg har sett de på.

AL: De er tilgjengelig på NIBIO.

E: Det er det. Der holder de akkurat på å oppdatere grensene mellom beitelagene på det kartet..

SD: I dagens rapporter, er det detaljert nok informasjon som sendes inn? Eller er det behov for mer fra bøndene?

E: Det er godt nok i forhold til å beregne tilskuddet, og det er jo det vår oppgave

er. Men det kan hende at beitebrukerne har et mer behov for å holde oversikten og trenger å vite hvor har vi gått. Det er to forskjellige behov her da.

AL: Vi har et enklere behov enn beitebrukeren selv.

SO: Så tilfredsstiller vi beitebrukeren, så tilfredsstiller vi dere og?

AL: Ja, absolutt.

A: Hadde det vært mulig for oss å se det skjemaet de må fylle inn?

E: Ja, det går an.

SO: Jeg tror vi prøvde å gå inn, men vi hadde problemer.

E: Ja, akkurat det skjemaet der, eStil-skjemaet, der er ikke jeg heller noen ekspert. Men jeg kan vise det til dere etterpå, hvis dere vil.

SD: Hvordan validerer dere at sauene blir fulgt godt nok opp? Eller er det deres område?

AL: Det er vel kanskje mer Mattilsynet. Så hvis Mattilsynet kommer med rapporter om at "Her er det for dårlig oppfølging", at de har brutt med dyrevelferden eller liknende, da kan vi trekke tilskudd.

E: Ja, så det er i forhold til tilskudd.

AL: Vi avvelter da Mattilsynet, det er de som skal ut å .. ?

E: Akkurat nå så holder vi på å fastsette et nytt regionalt miljøprogram og en ny forskrift for beregning av miljøtilskudd.

AL: Herunder beitebruk.

E: Herunder organisert beitebruk. Og det som er satt i forslaget nå er at det er krav til beitelaget om at de må dokumentere tilsynet, men det står ikke noe mer enn det. Så det betyr at vi har jo en kontrollfunksjon her, så vi kan reise ut til et beitelag og kontrollere om det er slik de har rapportert inn.

AL: Da kan vi gå inn å se.

E: Da kan vi spørre "Vi vil se listene dere har på tilsyn", og hvis de da ikke dokumenterer det og ikke har hatt noe tilsyn så kan det være grunnlag for å trekke tilskudd. Det er et avvik.

SD: Men er det noen andre måter å straffe bøndene på, enn å trekke tilskudd?

AL: Vi forvalter jo tilskudd, så hvis det er noe feil oppimot tilskudd, så er det jo trekk i tilskudd det går på.

SO: Jeg tror det er hardt nok.

AL: Mattilsynet har jo egne virkemidler, så de kan jo si at du ikke får lov til å drive med sau lenger. I verste fall.

SD: Vet dere hvor mange sau som forsvinner ca. i året eller hver sesong ute på beitet?

AL: Ja, det vet vi jo egentlig.

E: Det får vi inn i rapportene gjennom organisert beitebruk.

AL: Totaltapet var bortimot 10%.

E: Jeg har satt opp en beregning for Trøndelag nå utifra rådataen og det var 10%. For lam.

AL: Og det synes vi er altfor mye. 10% av lammene forsvinner i løpet av.. Nå er ikke alt på grunn av rovvilt, det er totale tapet er 10%, så det kan være andre dødsårsaker

også.

A: Vet dere hvor stor andel som blir tatt av rovvilt?

E: Det er vanskelig. Vi vet hvor stor andel som blir dokumentert tatt av rovvilt.

AL: Også vet vi hvor mange som blir erstattet. At det sannsynliggjort er rovvilt, men hvor mange av de 10 prosentene som er det?

E: Jeg husker ikke tallet på det, men hvis vi går inn på Rovvilt-portalen så finner vi de tallene.

AL: Men det er en ganske stor andel som er sannsynliggjort og som har fått erstatning, men det er igjen en bit som enda er ukjente årsaker.

SO: Du sier 10%, er det av alle sau eller av lam?

AL: Det er av lam, men av voksendyr er det 5% eller?

E: Ja, eller litt over i år nå. Tallene står ikke her, men det er jo sånn vi finner ut av i statistikken.

AL: Det er lammene som er mest utsatt, men når det gjelder bjørn, så går den mye på voksne dyr. Men andre rovvilt går mer på lam.

SD: Ser dere noen sammenheng mellom oppfølging og hvor mye som forsvinner? Altså, tror dere ved mer oppfølging så vil færre sau forsvinne i løpet av sommeren? Og omvendt?

E: Mhm.

AL: God oppfølging vil kunne, ja.. Klart at noen områder er veldig utsatt uansett da, selv om det er god oppfølging. Men at god oppfølging tror vi noen ganger kan være bra, og det kan være beskåret besetninger som ligger nærme hverandre, uten at man finner .. drifta som kan være med å bidra.

E: Det å kanskje følge opp etter at man har vært på tilsyn og sjekke hva det er som foregår, og gjøre noe, som for eksempel med det å sanke inn dyrene litt tidligere og ikke la de går for lenge ute om høsten.

AL: Det er nok noe av det viktigste. Venter man for lenge med å sanke inn, fordi at det er noe rovvilt som er veldig aktivt utover høsten, og hvis man får sanket ned dyra før det blir for sent, så kan man redde mange da.

SD: Dere nevnte jo den BeiteSnap appen tidligere, vet dere om noen som bruker den?

E: Jeg vet ikke om noen som har brukt den i Trøndelag, men jeg vet om noen som bruker den i Oppland som har prøvd ut den der. Jeg har kontakt med Sidsel Dønne som er fylkesmann der. Dem har hatt noen møter med noen beitelag og har prøvd ut. Men det er sikkert noen som har begynt å prøve det litt i Trøndelag.

SO: Steingrim nevnte at han hadde testa.

SD: Men i den appen så kan den generere en rapport.. som kan sendes inn?

F: Det hørtes ut som han Steingrim, som vi snakket med, at de hadde prøvd det ut, men at han kunne ikke selv laste ned noen rapport fra den dataen han hadde, men at BeiteSnap tok seg av det. Og sendte kanskje noe for beitelaget.

E: Da hørtes det mye bedre ut det dere hadde tenkt, da.

SD: Vi kan ta ett til, hvis dere har tid. Dere nevnte at dere har mye

statistikk og sånn, hvordan bruker dere den informasjonen som samles inn fra de rapportene fra bøndene?

E: Først og fremst bruker vi det til beregning av tilskudd. Vi trenger den dokumentasjonen for å finne ut om hvor mye tilskudd som skal betales ut. Også bruker vi det og for å lage statistikk.

AL: Analyserer og ser på utvikling og sånn.

E: Og sender tallene over til Mattilsynet, som sagt. Og vi sender det og til Miljøvernnavdelingen som sitter å beregner denne erstatningen, og at de kan ha det som bakgrunnsdokumentasjon og sammenligne med de tallene de får inn på søknadene. Så det er veldig bra med den statistikken vi får. Vi får en god oversikt.

SD: Det var vel det vi hadde.

E User Test

This appendix includes the user test description the test person was given (see the document in Figure E.1), and the different test tasks for the mobile (document (a) of Figure E.2) and web application (document (b) of Figure E.2).

Brukertest av Pecora

Takk for at du vil delta i en brukertest av Pecora!

Bakgrunnsinformasjon om Pecora

Pecora er et system som har et mål om å effektivisere tilsyn av sau på utmarksbeite. Dokumentasjon av utført tilsyn på beitet skal leveres til Fylkesmannen etter endt beitesesong, og det kreves at man utfører tilsyn minst én gang i uka i løpet av beitesesongen. I dag brukes penn og papir for å skrive ned observasjoner når gjeteren er ute på oppsynsturer. Det er et ønske om å digitalisere dette og å enkelt generere en oppsummeringsrapport av turene på slutten av sesongen.

Pecora appen har blitt utviklet for å erstatte penn og papir på oppsynsturer. Appen loggfører turen gjeteren går i terrenget ved hjelp av GPS og har støtte for registrering av de vanligste observasjoner gjort på utmarksbeitet. De vanligste observasjonene er døde og skadde sau med årsaker, rovdyr, løshunder, generell uro, søyer som mangler lam, beiteforhold, men også andre ting. Kartfesting av noen observasjoner kan være aktuelt og ved hjelp av mobilappen løses dette enkelt. Det er også vanlig å ta bilder av visse observasjoner, som for eksempel tegn på en død sau (ullrester, o.l.).

Turene fra mobilappen kan enkelt synkroniseres mot Pecora's webapplikasjon. Webapplikasjon viser turene gjeteren har gått, og man har muligheten til å se flere turer samtidig på kartet og turer fra en ønsket tidsperiode. Det er også mulig å laste ned en PDF-rapport med en enkel oppsummering av turene man har gått.

Annen informasjon

I brukertesten skal du utgi deg for å være en gjeter som ønsker å gjennomføre en oppsynstur. Du skal bruke demo-versjonen av appen slik at du ikke behøver å gå en faktisk tur ute.

Figure E.1: The test background and description.

<p>Brukertest</p> <p>Del 1 - App</p> <p>Steg 1 Registrer en bruker og logg inn.</p> <p>Steg 2 Last ned et kart over terrenget du befinner deg i hvor du ønsker å utføre en oppsynstur.</p> <p>Steg 3 Start en oppsynstur med ønsker informasjon og kartet du lastet ned.</p> <p>Steg 4 Oppsynstur: <i>(Du beveger deg rundt på kartet ved å holde inne fingeren på stedet du vil gå til.)</i></p> <ul style="list-style-type: none">- Beveg deg rundt på kartet og registrer følgende observasjoner med litt avstand fra hverandre (om annet ikke står):<ul style="list-style-type: none">- 20 sau- 10 lam i nærheten av de 20 sauene- 1 kongeørn- 1 død sau som ikke er dekket til<ul style="list-style-type: none">- Ta et bilde som bevis- 10 hvit sau- Når du er ferdig beveger du deg tilbake til startpunktet og avslutter turen. <p>Steg 5 Finn frem til oppsynsturen du nettopp lagret.</p> <ul style="list-style-type: none">- Se turen på kartet.- Gå tilbake og finn informasjonen om den døde sauene du observerte. <p>Steg 6 Test hvordan du sletter turen din, men <u>ikke</u> slett den.</p> <p>Steg 7 Synkroniser turen din.</p>	<p>Del 2 - Web</p> <p>Steg 7 Logg inn på pecora.no med brukeren din.</p> <p>Steg 8 Se på den generelle informasjonen om turen din.</p> <p>Steg 9 Se på informasjonen om kongeørnen.</p> <p>Steg 10 Last ned en oppsummeringsrapport og åpne den.</p> <p>Steg 11 Logg ut.</p> <p><i>For å bruke mindre tid skal du nå logge på en annen bruker for å ha tilgang på flere turer.</i></p> <p>Steg 12 Logg inn med brukernavn "kari" og passord "kari".</p> <p>Steg 13 Vis de 3 siste turene på kartet.</p> <p>Steg 14 Fjern den siste turen fra kartet.</p> <p>Steg 15 Finn alle oppsynsturer mellom 26. april kl. 17:00 og 3. mai og vis alle på kartet.</p> <p>Steg 16 Tilbakestill lista.</p> <p>Steg 17 Last ned en oppsummeringsrapport.</p> <p>Steg 18 Logg ut.</p>
--	---

(a)

(b)

Figure E.2: The test steps for the mobile and web application.