Bachelor's project

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Johannes Möckel
Ole Martin Ødevald Ruud
Roald Strangstadstuen

Distributed Equipment Tracking in Unreliable Network Environments

Exploring Affordable Digitization of Equipment Tracking for the Norwegian Home Guard

Bachelor's project in Engineering - Computer Science, IT-Operations and Information Security

Supervisor: Sony George

May 2020

NTNU
Norwegian University of Science and Technology

Johannes Möckel
Ole Martin Ødevald Ruud
Roald Strangstadstuen

Distributed Equipment Tracking in Unreliable Network Environments

Exploring Affordable Digitization of Equipment Tracking for the Norwegian Home Guard

Bachelor's project in Engineering - Computer Science, IT-Operations and Information Security
Supervisor: Sony George
May 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of Science and Technology

Abstract

The communications section of the Norwegian Home Guard must always have an overview of the locations of their equipment. They want to acquire a digital system to track their equipment. The system will be distributed across multiple storage locations with heterogeneous infrastructure, while staying within the budget of the Norwegian Home Guard. Our exploratory work consists of a requirement analysis, researching existing solutions and developing a proof of concept. The core technologies we researched are distributed systems, databases and computer-readable tagging solutions. We bring forth some key recommendations to fulfill both technical and economical requirements. There have been no revelations which would prevent the Norwegian Home Guard from continuing the development of the system.

Sammendrag

Sambandsseksjonen, G6, i Opplanske Heimevernsdistrikt skal til en hver tid ha kontroll over materiell og utstyr. De ønsker å anskaffe et digitalt system for å spore utstyret deres. Hovedutfordring er at systemet må settes opp ved mange lokasjoner med ulik infrastruktur, samtidig som det overholder Heimevernets økonomiske rammer. For å utforske mulighetsrommet har vi gjennomført behovsanalyse, undersøkt eksisterende løsninger og utviklet et konseptbevis. De viktigste teknologiene vi har undersøkt er distribuerte systemer, databaser og maskinlesbar fysisk merking. I oppgaven legges det frem flere forslag for å imøtekomme de tekniske og økonomiske kravene. Det er ikke gjort noen funn som forhindrer Heimevernet i å fortsette utviklingen av systemet.

Acknowledgements

First of all we would like to thank Major Leif Tanum from the communications section of the Norwegian Home Guard as well as the Norwegian Home Guard as a whole. We appreciate you taking time and allowing us to visit Terningmoen, as well as submitting the task in the first place.

We would also like to thank senior adviser Espen Thorseth of the department of information security and communication technology at NTNU for being available when Leif Tanum was busy. He helped us with the initial steps, defining the limitations and the end goal of the project, and supported us along the way. He has also provided us with most of the practical information needed for this thesis.

We thank our thesis adviser, associate professor Sony George of the department of computer science at NTNU for guiding us through the process of writing a bachelor thesis. He continuously helped us improve our thesis by providing recommendations and pointing out flaws. His input has been invaluable.

We would also like to thank SAAB at Rena represented by support manager Østen Almqvist for allowing us to visit. He and his coworkers helped tremendously with their practical experiences. This visit provided many key arguments for the thesis and further helped us visualize the end goal.

Lastly, we would also like to thank the several individuals that helped us proofreading the thesis.

Contents

Figures

Glossary

Bulk-scanning The action of scanning multiple items at the same time. 9, 18, 22, 34

Individually-tracked equipment Equipment that has a unique identifier and is tracked based on this identifier. 2, 18, 19

Information Technology Infrastructure Library A set of detailed practices for IT service management that focuses on aligning IT services with the needs of business. 6

Inventory Management Software A software system for tracking inventory levels, orders, sales and deliveries. 6, 16

NATO Stock Number A 13-digit numeric code used for identifying "standardized material items of supply" recognized by all NATO countries. 8

Proof of concept A proof of concept is a demonstration of an idea or method to prove it's feasibility or demonstrating it's practical purpose. 1

Quantity-tracked equipment Equipment without a unique identifier, that is tracked by quantity. 2, 17–19

Radio-frequency identification A technology that uses electromagnetic fields to automatically identify and track tags attached to objects. 7

SAAB SAAB AB is a Swedish aerospace and defense company that supplies military training equipment to the Norwegian Armed Forces. vii, ix, 17–19, 22, 23, 33

The Norwegian Home Guard The Norwegian Home Guard has 40,000 soldiers all over the country and they serve as a quick mobilisation force for the Norwegian Armed Forces. vii, ix, 1, 3, 11, 12, 15–17, 19, 20, 22, 23, 25, 28, 33

Warehouse Management System A software application designed to support and optimize warehouse functionality. 15

Acronyms

ACID Atomicity, Consistency, Isolation and Durability. 21

API Application Programming Interface. 21

IMS Inventory Management Software. 6, 7, 15, 16, *Glossary:* Inventory Management Software

IS delivery form (meaning *innleveringsseddel*) (example found in Appendix A). 1, 16, 17, 25

ITIL Information Technology Infrastructure Library. 6, 7, *Glossary:* Information Technology Infrastructure Library

MVCC Multi-Version Concurrency Control. 21

NSN NATO Stock Number. 7, 8, 23, *Glossary:* NATO Stock Number

PoC Proof of concept. xi, 1, 11–13, 29, 30, 32, 33, 45, *Glossary:* Proof of concept

QR code Quick Response code. 7, 8, 18, 22, 23, 34, *Glossary:* Quick Response code

RFID Radio-frequency identification. 7–9, 17, 18, 22, 23, 34, *Glossary:* Radio-frequency identification

UHF Ultra High Frequency. 9

US submission form (meaning *utleveringsseddel*) (example found in Appendix A). 1, 16, 17, 25

WMS Warehouse Management System. 6, 15, 16, *Glossary:* Warehouse Management System

Chapter 1

Introduction

In this chapter we will describe the motivation and background behind the thesis. Based on the task description we will define some requirements for the resulting software system. Further, we will make a hypothesis based on the requirements and define some research questions to help us explore the relevant domains of the system.

1.1 Background and Motivation

The communications section of the Norwegian Home Guard must always have an overview of the locations of their equipment, such as radios, antennas, cables, etc. Such equipment can be expensive, take a long time to replace if damaged or lost, or contain cryptographic keys. Currently this control is maintained via the use of paper receipts called submission form (US) (meaning *utleveringsseddel*) and delivery form (IS) (meaning *innleveringsseddel*). This process is labor intensive and can potentially be prone to cause errors, some of which can be hard to discover. Additionally, when errors are discovered, they can take significant time and effort to correct.

The main goal of the project is to give the Norwegian Home Guard a solid foundation for further developing an equipment tracking system. This includes proving the feasibility of such a system — by developing a Proof of concept (PoC) — and providing several educated recommendations and considerations. The project will conclude with a general recommendation to the Norwegian Home Guard on whether we believe further efforts to digitize the process are advisable.

1.2 Preliminary Requirements

The following list is our initial outline of requirements based on the task description. Any further requirements that come to light will be added to a revised list of requirements. We will also add clarifications for existing requirements. The revised list of requirements can be found in Section 4.3.

1. The primary use case of the system is to keep track of the location, state of and the responsible party of equipment.
2. The system shall use some kind of computer readable tagging solution to ensure ease-of-use when handling equipment.
3. The system needs to enable the user to retroactively register or correct equipment movements.
4. The system shall remain locally available and functional despite network partitioning from other parts of the system, as it will be deployed across multiple geographical locations.
5. The system must be able to re-synchronize information across different locations after a network partition has been resolved.
6. The system needs to be able to handle both individually-tracked equipment and quantity-tracked equipment.
7. The system needs to maintain a limited history of movements for individually-tracked equipment. The history shall be searchable.
8. The system shall require minimal training for storage facility operators. The user interface needs to be easy to understand in potentially stressful situations.
9. Creating, deploying, using, maintaining and decommissioning the system needs to be economically viable.

1.3 Hypothesis

We claim that it's feasible to create a distributed equipment tracking system which satisfies the requirements outlined in Section 1.2.

1.4 Research Questions

To aid exploration and research into the topic of distributed inventory management systems, we chose the following research questions:

1. How does Brewer's Conjecture [1] — also know as CAP theorem — affect the proposed distributed equipment tracking system?
2. What are the key advantages and disadvantages of different equipment tagging solutions?
3. How can the user interface best ensure ease of use and help to prevent erroneous usage?
4. What are the main economic factors to consider when creating, deploying, using, maintaining and decommissioning the equipment tracking system?

1.5 Limitations and Assumptions

Limitations are topics we choose not to investigate, even if some parts might be highly relevant to the system. We will also make assumptions regarding aspects of the system environment.

Most of the limitations and assumptions stem from the fact that information required to research a given topic is not disclosed to us. This is due to the regulations of the Norwegian Armed Forces, preventing them from sharing certain information with us. Some topics might also hinge on the actual implementation details of a finalized system. As we are not delivering a final product, we will be unable to draw conclusions on these topics. In these cases, we will highlight some of the challenges that would need to be handled during the development of the actual implementation.

Assumptions

- We assume that the Norwegian Home Guard will already provide a network infrastructure which is both isolated and encrypted.
- We assume that the physical access to the system will be controlled by external access control mechanisms.

Limitations

- We will not consider physical constraints of equipment tracking tags, such as what glue should be used or if the tags need to be shielded from sunlight.
- We will not deliver a fully functioning system, rather a recommendation on the viability of such a system. If a digitalized system shall be implemented based on our findings, then such a system would be created from the ground up.

1.6 Dissemination

This thesis is going to be given to Espen Torseth — our contact person with the Norwegian Home Guard — who will then disseminate the thesis to the relevant parties that can continue the process.

Chapter 2

Background

In this chapter we will highlight some of the technical background information we will use in this thesis. This will primarily be information about distributed systems, existing asset and equipment tracking solutions and tagging technologies.

2.1 Distributed Systems

There are multiple ways to define distributed systems. Tanenbaum defines a distributed system as "a collection of independent computers that appears to its users as a single coherent system" [2, p. 2]. He further emphasises that this highlights two important aspects of a distributed system; that the different components of the system are autonomous – hence still functional when separated from the system – and that the users of the system experience it as a single entity.

Since the components of the system are autonomous, they need some way to collaborate to ensure that the system can appear as a single entity to its users. This collaboration is a core aspect of any distributed system. For the system to be functional in dynamic environments the communication between different nodes has to be resilient to network partitioning. The system must also be able to reach a consensus on the state of the system, and its underlying data.

The main goal of a distributed system is to ensure that a service is available to the users of the system [2, p. 3]. Networks are inherently unreliable and limited in capacity, which means the access to the system is fundamentally limited by the connection between the users and the system. As the distance between the users and the system increases, the access to the system is dependent on an increasing amount of network nodes. This causes an increased delay and adds more fallible parts to the system. A distributed system can limit the dependence on network infrastructure by moving the service closer to the users. This means that if users loose access to one entry point of the system, they can quickly start utilizing another entry point. This somewhat accounts for failing network nodes. Finally, a distributed system could also give users access to a service despite being isolated from other nodes. The isolated node could then opportunistically synchronize the changes with the rest of the system.

A distributed system enables the service administrators to provide high availability to their users. However, it also poses many challenges. Peter Deutsch, who worked at Sun Microsystems, put forth eight fallacies of distributed systems [2, p. 16];

1. The network is reliable.
2. The network is secure.
3. The network is homogeneous.
4. The topology does not change.
5. Latency is zero.
6. Bandwidth is infinite.
7. Transport cost is zero.
8. There is one administrator.

Developing a system in an environment as highlighted in the above is essentially a game of trade-offs. As we dive deeper into Brewers Conjecture [1] and distributed databases [3] these trade-offs become more apparent.

Brewers Conjecture Brewer's Conjecture[1] — also called CAP theorem — states that a distributed system can only ever attain two of the following three properties; consistency, availability, and partition tolerance. Consistency means that the system should appear like an atomic data object, in the sense that each operation should appear as occurring in a single instant [1, ch. 2.1]. Availability means that every non-failing node in the system should respond to a request within reasonable time [1, ch. 2.2]. Partition tolerance means that a node in the system should continue to respond to requests despite being isolated from the rest of the system [1, ch. 2.3].

2.2 Existing Solutions

There are a variety of different existing solutions for tracking assets and equipment already on the market. These can mostly be divided into three different categories meant for different purposes; Warehouse Management Systems (WMSs), Inventory Management Software (IMS) and Information Technology Infrastructure Library (ITIL) systems. All these different categories are intended for different use cases, and typically offer different features. All these solutions track inventory in some way and will as such be considered.

Warehouse Management System The first category of asset and equipment tracking solutions are WMSs. Some examples of WMSs we looked at were WISE WMS [4] and Infoplus Warehouse Manager [5]. As the name suggests, these systems are meant to manage warehouses. Of the three overarching categories this one initially seems the most promising, as WMSs are used to manage wares within a warehouse and try to improve the flow of wares inside a warehouse. They also try to optimize incoming and outgoing orders of wares.

Inventory Management Software Next, we will look at IMS. Some examples of IMS we looked at were Brightpearl [6] and Katana [7]. IMS is meant to manage inventory levels, orders, deliveries and sales. While these systems often can track inventory as it moves between locations, they are usually more focused on tracking the remaining amount of wares at a location, and coordinating the ordering of new wares as the stock of them runs out. This is well suited for stores, where goods come in, are kept in a localized storage before getting moved, after which new wares need to be bought to restock the local storage.

Information Technology Infrastructure Library Lastly, we also took a look at ITIL systems. One example of such a system we looked into is Cherwell IT Service Management [8]. These systems tend to be very extensive, including features such as license management, and many more features that go way beyond the scope of what is required for the purposes of tracking equipment between different locations. The extensiveness of these systems means they are more difficult to use, and require substantial training. These systems are not intended to be used on-site in warehouses, and rather meant for configuration management and managing IT assets, such as licenses. They are not intended to be used to track wares in a warehouse.

Their immense complexity and vastly different use cases makes them unsuited for the task at hand. As such, while we had initially intended to further research ITIL systems as some of them contain asset tracking capabilities, we have decided to revise this and will not pursue ITIL systems any further as they are patently unsuited for our task without the need of further discussion.

2.3 Equipment Tagging Technologies

The goal of equipment tagging technology is to enable a computer to identify a physical object. This allows the operator to quickly tell the computer about individual items that are being processed. The main benefits of tagging instead of manually entering an identification code is efficiency, ease-of-use and error prevention.

There are many different technologies for marking equipment with computer readable tags. Some common tagging solutions are barcodes, Quick Response codes (QR codes) and Radio-frequency identification (RFID) tags.

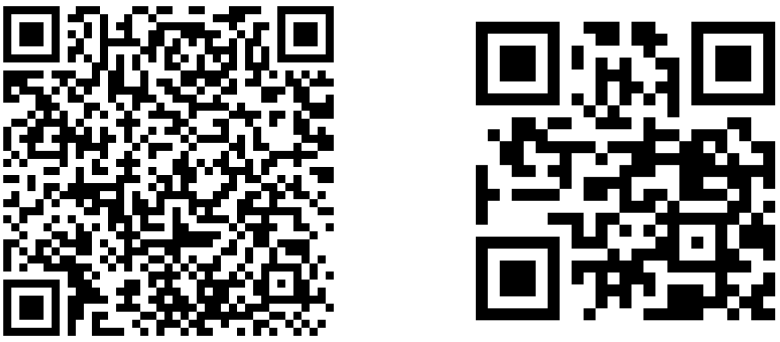
(a) EAN-13

(b) Code 128 containing NATO Stock Number

Figure 2.1: Some example barcodes

Barcodes Barcodes are commonly used visual codes. Due their usage in global consumer product identification they can be found on virtually any consumer product in the world[1]. There exist many barcode specifications and versions [9], which tend to vary in what types of data they can contain. Figure 2.1a shows a barcode specification which is used for global consumer products, while Figure 2.1b shows an encoded NATO Stock Number (NSN) in Code 128. Both barcodes in Figure 2.1 were generated using revilo's `barcode_gen` application[2].

The main benefits of barcodes is their simplicity compared to QR code and RFID, and their small form factor. A barcode can be downscaled vertically without loosing information, however this may make it more vulnerable to wear-and-tear. This is because less of the code has to disappear, or become otherwise obfuscated, before the code becomes unreadable. Another benefit is that since barcodes are widespread, the equipment to create and read them is easily available at a reasonable price.



(a) QR code containing 117 bytes **(b)** QR code containing NSN

Figure 2.2: Some example QR codes

QR codes QR codes are visual codes that can contain arbitrary data. They can be thought of as two dimensional barcodes. Both example QR codes in Figure 2.2 were generated using Fukuchi's `qrencode`[3].

The main benefit of QR codes is that they contain some error correction [10]. At the highest error correction level, up to 30% of the code can be missing or damaged without affecting the readability of the QR code. Additionally QR codes can also contain large amounts of data — up to 23,648 bits [10] for the highest capacity barcodes with the least amount of error correction — compared to barcodes.

[1] GS1 — a non-profit global standards organization — developed a global business standard for barcodes which is used across the globe for product tracking and identification. See `https://www.gs1.org/standards/barcodes/ean-upc`

[2] `https://github.com/revilo/barcode_gen/`

[3] `https://fukuchi.org/works/qrencode/`

Since the code is two dimensional, and relies on image analysis, it requires a more sophisticated reader than a barcode. Further, since they can contain more arbitrary data than a regular barcode, one should also consider the possibility that an attacker could craft malicious tags that can potentially exploit system vulnerabilities when read [11].

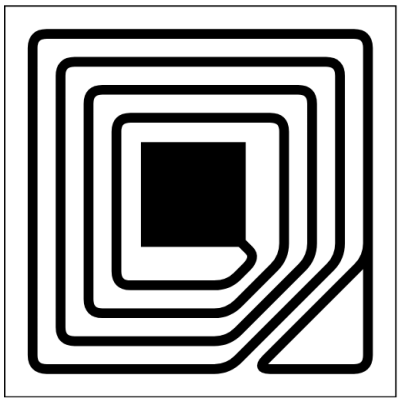


Figure 2.3: An example RFID tag

Radio-frequency identification tags RFID tags use radio waves in order to send both an identifier for the tag itself, as well as optionally some other arbitrary data. This information is sent over radio waves by the tag as soon as a compatible reader is brought within range. This range can vary greatly based on what frequency, tags and reading equipment is used. The technical specification of RFID can be found in ISO 18000-series. We have primarily looked at part 6 [12] which concerns Ultra High Frequency (UHF) RFID.

These tags exist in many different shapes and sizes. Common to them all is that they have a tiny radio transponder. An example of the visual appearance of a tag can be found in Figure 2.3[4]. In the example the surronding lines are the antenna and the central black box is the processing chip.

This technology has 2 main advantages over image based scanning technologies; with UHF RFID you can scan multiple tags at the same time, which we will henceforth refer to as bulk-scanning, and additionally penetrate certain materials when scanning. However the ability to scan multiple items can also be a challenge in a storage facility as one can accidentally scan items that are next to the item that one wishes to scan. Further, equipment used to create and read RFID tags is more expensive and less widespread than visual code based alternatives.

[4]The image was created using draw.io, a online drawing program. `https://draw.io`

Chapter 3

Method

To explore and answer our research questions we have chosen four different methods; contineous requirement analysis, analyzing existing solutions, on-site visits, and creating a simple PoC. We believe these methods will give us the best basis for answering our research questions. As we are exploring a complex task, we chose methods that enable us to dig deeper into the underlying problems. We hope this insight will provide a thorough basis for potential future development of an equipment tracking system.

3.1 Continuous Requirement Analysis

The goal of performing a requirement analysis is to ensure that the provided requirements are accurate to the task description. Further, analyzing the requirements will give us a solid foundation for providing accurate recommendations to the Norwegian Home Guard.

The requirement analysis will be a continuous process throughout the project which will be interleaved with the other methods. The on-site visits will be one of the more important parts of this analysis because it will enable us to both question experts and examine the current solution the Norwegian Home Guard is using. Further, knowledge gathered from other research and personal experience will be part of this analysis.

The results of applying this method should be a revised list of requirements and several discussions surrounding specific technologies which aim to fulfill those requirements.

3.2 Analysis of Existing Solutions

The goal of analyzing existing solutions for equipment tracking and storage management is threefold; gaining a better understanding of the problem domain, discovering key advantages and disadvantages of various existing systems and evaluating if any existing solutions satisfy our requirements.

To analyse existing solutions, we will start by taking a broad look at what kind of solutions exist. We will use the official information by the service provider, crowd sourced information (such as Wikipedia[1]), and openly accessible reviews. For each solution we examined what the intended use case was, what the feature set included, and if there were any highlighted issues with it. Based on the previous examination we will group the solutions based on use case. Lastly, we will evaluate if the different groups satisfy our requirements.

The results of applying this method should be background information about different existing solutions, added to Chapter 2, and a discussion of our findings, added to Chapter 4.

3.3 On-site Visits

The goal of performing on-site visits is gathering valuable information and revising preliminary requirements, as found in Section 1.2. This will include a visit at the Norwegian Home Guard and other potential locations. The on-site visits will provide much needed insight and will most certainly expose otherwise hidden considerations.

The main reason we chose to conduct on-site visits is that the process surrounding equipment tracking is complex and there are a lot of caveats. Informed and valuable responses require the respondent to have extensive knowledge of the infrastructure, organization, and processes related to equipment tracking. Thusly, we believe on-site visits will be a valuable source of information.

On a contrary note, on-site visits with experts may cause the results to be affected by what Zajonc calls the "mere-repeated-exposure paradigm" [13]. This is a cognitive bias which highlights that individuals usually develop a preference for things simply because they are familiar with it. One aspect that might be particularly affected by this is opinions regarding the user interface. This is likely due to the fact that experts of a system will be proficient with the user interface and hence not have the same issues or concerns as an outsider. We should be conscious about this when assessing the results of the visits.

To perform the on-site visits, we will first gather as a group and prepare some explicit goals for the visit. This should include producing a document with relevant questions and keywords. During the visits, we should ensure that the prepared topics are both discussed and further clarified if necessary. Lastly, we should encourage stakeholders to clarify potentially lacking parts of the task description.

The results of applying this method should be a summary, added to Chapter 4.

3.4 Proof of Concept

The goal of creating a PoC is to test our hypothesis, defined in Section 1.3, and demonstrate the feasibility of our recommendations. Further, it provides us with

[1] `https://wikipedia.org`

a good indication on what aspects would need to be further explored.

The creation of the PoC will be depended on the results of the previously mentioned methods. These results will play an important part in choosing what aspects of the system should be part of the PoC. We intend to create a simple application which will demonstrate the feasibility of several recommendations.

The results of applying this method should be a simple demonstration of some of the recommendations found through the other methods.

Chapter 4

Results and Discussion

In this chapter we will present our results, discuss our findings and make some recommendations based on them. We will also highlight some points that need to be taken into further consideration by the Norwegian Home Guard which we will not draw further conclusion on as per the limitations outlined in Section 1.5.

4.1 Analysis of Existing Solutions

In this section we will highlight some overarching issues that all pre-existing solution we examined suffered from. We will also more specifically evaluate and discuss individual issues with WMSs and IMS mentioned in Section 2.2 in order to determine whether or not they are suitable for the purpose of equipment tracking at the Norwegian Home Guard.

Overarching issues There are som other bigger issues shared by all solutions we examined. These issues stem from the fact that the Norwegian Home Guard is not a regular company, but rather part of the Norwegian Armed Forces, and as such must be prepared to function in combat situations. A system for the Norwegian Home Guard would have some stricter than usual requirements for confidentiality, as well as other unusual requirements. Examples of such requirement are that the system needs to be able to function isolated from other storage locations and without access to the internet. Additionally, we need to be able to operate under conditions with low bandwidth and high latency. Thus, we deem it to be hard to find an existing solution that fulfills these requirements. Due to this a new system will likely must be developed from the ground up.

Warehouse Management System (WMS) Initially WMSs seemed like a promising solution for our problem. Going by name alone, something that manages warehouses sounded like a good candidate given that the Norwegian Home Guard has many locations, each with its own stored equipment and wares. However,

WMSs focus more on optimizing locations of wares inside the warehouses. Meanwhile our system should focus on tracking equipment as it moves between locations, as well as tracking who is response for it at any given time. Tracking wares as they move outside of the warehouse is not the intended use case for WMSs. As such, we realized that WMSs does not satisfy our requirements. Considering that the main use case of a WMS thusly does not match ours, coupled with some shared overarching issues that all pre-existing solutions bring (see below), we determined that a WMS would not be a suitable solution in this case.

Inventory Management Software (IMS) As described in Section 2.2, IMS tracks inventory levels, incoming and outgoing wares and tries to optimize for inventory levels. The main issue with using a IMS for our purposes is that is assumes that wares that leave the warehouse are not returned and need to be replaced. In a regular store setting where items are sold and not returned this makes sense. This would however not work here, as the Norwegian Home Guard largely reuses most of its equipment. Wares do not need to be restocked once they are checked out on training, as most of the equipment will be handed back into the warehouse once the training is over. Another important feature required by the solution is the ability to track the history of individual items. This is not something most IMS is set up to do. An additional issue with using a IMS solution for this purpose is that most such solutions would require more training than we would ultimately like to require, and the need to be able to retrospectively register events would not necessarily be supported.

4.2 Summaries of On-site Visits

4.2.1 The Norwegian Home Guard

Early on in our research we had an on-site visit the Norwegian Home Guard at Terningmoen. We had three main reasons why we wanted to visit them.

1. Get more information about the Norwegian Home Guard as a whole, better understand how they are structured and potentially learn about other requirements that we hadn't considered yet.
2. Get a tour of their current storage rooms to get an idea for what environment our solution must operate within.
3. Get some more insight into how the current IS and US based solution works, and if possible, ask what wishes the employees at the Norwegian Home Guard had for our system.

At Terningmoen we met up with Leif Tanum — the project owner — who gave us an introduction to how the Norwegian Home Guard is structured and what their purpose is. We then received a tour of various areas of the base including all their storage facilities. We also got to talk with some of the people that would be

using our system and got a demonstration of how the current IS and US works in practice.

The visit gave us a lot of insights that impacted what our requirements for the system were. We learned from Leif that our system would be deployed to around 750 different locations, and that some of these locations do not have reliable internet access, and some might even just be able to communicate via low-bandwidth radio connections that need to be reserved for more critical communications.

While taking a tour of their storage rooms we quickly realized that the Norwegian Home Guard didn't use specialized storage rooms for their equipment. Storage rooms could be anything from empty halls with shelves to defunct office rooms. What we learned from looking at the storage rooms at Terningmoen is that the storage rooms our system is to exists within are not optimized for storage of equipment, might have little to no walking space, and are often spread great distances apart on different parts of the different bases.

We also got a lot more insight into their current process using IS and US by having one of the storage facility workers walk us through how such a form would be filled out in practice. The main takeaway from this was that exactly how to fill them out was unclear. There were multiple fields that were either no longer used, would be filled differently every time or just completely repurposed over time.

Lastly while talking to the storage facility worker, he also mentioned to us that what we were trying to accomplish here reminded him a lot of "SAAB's system". Upon hearing this we inquired further we were informed that SAAB at Rena — who supplies the Norwegian Home Guard with training equipment — is using a very similar system to the one we were working on. When bringing this up to Leif, we received contact information for SAAB and were encouraged to take further contact with them.

4.2.2 SAAB

After our visit at the Norwegian Home Guard — in which we were informed of a similar system to ours at SAAB — we immediately sought contact with SAAB and arranged a on-site visit to SAAB in Rena where they are managing the combat simulation equipment used by the Norwegian Army, including the Norwegian Home Guard. We had hoped to gain insight into some of the technical decisions they made when developing their system that we could take into account for ours. Their experiences would be especially relevant for us since SAAB uses their equipment in similar conditions as the Norwegian Home Guard. Some of the questions we had were:

1. How do they handle their data storage, especially in respect to tracking the history of items?
2. How do they handle quantity-tracked equipment?
3. Have they attempted to use RFID or other tagging solutions?

From our visit we learned that SAAB uses Microsoft SQL Database[1] — a centralized relational database management system — to store all their data, including historical data. Their system has tracked around 350.000 pieces of equipment over a period of over 18 years, and the total size of the database is around 1GB. This means that we likely will not face capacity issues either.

Further we also learned how they handle the distinction between individually-tracked equipment and quantity-tracked equipment. The identifier of all their equipment consist of a type identifier (meaning *artikkel nummer*) and a serial identifier (meaning *serie nummer*). For quantity-tracked equipment the serial identifier is always 0.

SAAB's Tagging Experience

The most important insight we received from the visit was their experience regarding tagging. At SAAB they had been experimenting with RFID, QR codes and barcodes, and they told us about their experiences with each of the three tagging alternatives.

Radio-frequency identification Their experience with RFID was that it's hard to get the balance of reading just the right tags. Some of the issues they faced where the following:

- They would frequently read extraneous tags, in addition to the tags they were originally intending to scan.
- They found the need to use shielded enclosures[2] inconvenient when bulk-scanning items.
- They found that a majority of items need to be individually inspected for damage regardless, voiding the advantage of bulk-scanning items that RFID can provide.

QR codes With QR codes they experienced one predominant problem; the physical size. One can downscale the size of the codes, however this makes them harder to read and easier to damage. Further — due to the two-dimensional form factor — the tags were hard to apply on smaller items such as cables.

Barcodes In their experience, they did not run into many problems with barcodes. Occasionally the tags get worn out, however barcodes usually have their corresponding numbers printed under the code, which can be used as a secondary way of identification.

[1] Microsoft SQL Database is a relational database management system from Microsoft. `https://www.microsoft.com/en-us/sql-server/`

[2] A box large enough to pass the items through. This would be similar to a luggage scanner at an airport, but for scanning for tags instead.

4.3 Revised Requirements

The following is a updated list of requirements we have for the system. This list is based on the list we presented in Section 1.2. We have added new requirements and some requirements have been made more specific after our visits at the Norwegian Home Guard and SAAB. All modifications are marked using bold text.

1. The primary use case of the system is to keep track of the location, state of and the responsible party of equipment.
2. The system shall use computer readable tagging solution to ensure ease-of-use when handling equipment.
3. The system needs to enable the user to retroactively register or correct equipment movements.
4. The system shall remain locally available and functional despite network partitioning from other parts of the system, as it will be deployed across multiple geographical locations **(around 750)**.
5. The system must be able to re-synchronize information across different locations after a network partition has been resolved.
6. The system needs to be able to handle both individually-tracked equipment and quantity-tracked equipment.
7. The system needs to maintain a limited history of movements for individually-tracked equipment. The history shall be searchable.
8. The system shall require minimal training for storage facility operators. The user interface needs to be easy to understand in potentially stressful situations. **Users shall be able to effectively learn how to use the system within 30 minutes of training.**
9. Creating, deploying, using, maintaining and decommissioning the system needs to be economically viable[3].
10. **The system needs to be usable in suboptimal storage environments, such as multiple spread storage rooms, cramped storage rooms with little walking room or space, and other repurposed rooms used for storage.**
11. **The system needs to be able to function and be maintained in locations with lacking or nonexistent internet connections, as well as potentially low-bandwidth radio connections.**

4.4 The Implications of a Distributed System

In this section we will discuss the implications of developing a distributed system. Further, we will aim to provide the basis for answering research question 1.

As the system will be distributed, as required by requirement 4 in Section 4.3, hence we have to consider Brewer's Conjecture as explained in Section 2.1.

[3]We did not find a concrete number for what would be considered economically viable for the system, so we will instead just consider differences in economic cost.

The most important requirement is that the service must operate distributed across multiple locations without consistent inter-connectivity, also called being **partition tolerant**. This stems from fact that the system will be deployed across unreliable network connections and sometimes in entirely isolated networks.

The second requirement is that the service needs to be **available** when deployed. If the service is not responding to requests from users or is unreasonably slow to respond, then the service would not be a efficient solution for tracking equipment.

As the service already needs both availability and partition tolerance, it cannot provide **consistency** as determined by Brewer's Conjecture. However, as our system will be partially synchronous [1, s. 4], it can provide a weaker form of consistency. Essentially the service can provide a practical compromise where we arrive at some eventual consistency.

As some nodes of the system will be in remote connections, a synchronization might not be available via a network connection. We are suggesting to have optional synchronization via the use of radios, telecommunications or even via portable storage media. We leave it up to the Norwegian Home Guard to figure out what aspects are practical for this solution. This factor might increase the development cost of the system, however would be essential for especially remote locations or deployed military storage facilities.

4.5 Choosing a Database Back-end

In this section we will discuss advantages and disadvantages of multiple database solutions. Finally, we will provide a recommendation for a suitable database back-end.

A way of satisfying requirement 4 — that the system has to be distributed across multiple geographical locations while remaining available despite network partitions — is by choosing a database management system that already provides distributed storage and consensus. This is enough to satisfy the requirement in our case, since the only part of our system that needs to distribute across multiple locations is the data. The only communication between different locations should ideally be data synchronization.

Choosing a distributed database management system would enable the user-facing application to be mostly unaware[4] of the distributed nature of the system, and hence greatly simplify it's architecture and implementation. Hence, we will only consider distributed database management systems, which means that popular database management systems such as PostgreSQL[5] and MySQL[6] will not be considered.

[4]The application still has to use a data model which is designed for a distributed storage system.

[5]PostgreSQL is a open source object-relational database system with over 30 years of active development. `https://www.postgresql.org/`

[6]MySQL is a open source relational database management system owned by Oracle Corporation. `https://www.mysql.com/`

We will assess two different distributed database management systems — CouchDB[14] and Couchbase[15] — to see if they meet the requirements highlighted in Section 4.4.

CouchDB CouchDB is a distributed database management system which uses a document-oriented storage model. It enables distributed servers to reach eventual consistency, even after long periods of non-synchrony. Further it provides fault tolerance due to it's Atomicity, Consistency, Isolation and Durability (ACID) properties. Lastly, it's use of Multi-Version Concurrency Control (MVCC), as further explained by Bernstein, Philip A. and Goodman, Nathan[3], enables it to serve any number of clients to concurrently, and thus, provide the clients with a consistent snapshot of the database in every transaction.

The main feature that separates CouchDB from other database management systems is the ability to provide distributed updates and replication, even when servers are isolated from the network for an extended period of time. This means that the system can be set up in different geographical locations or across unreliable network connections, and still be functional and available to the local users. When a network between the servers is available, the databases will synchronize and merge previous changes. Further this synchronization and replication can be precisely controlled using the replication API[7], which would enable the system to control both when and what is synchronized. This enables the administrators of the system to prioritize what applications consume the potentially limited network bandwidth.

Couchbase Couchbase, formerly known as Membase, is a distributed database management solution which also uses a document-oriented storage model. This database is optimized for interactive enterprise applications using replication and horizontal scaling to enable a high throughput of transactions. The system is native to the cloud, enabling itself to be highly available due to massive scaling possibilities.

Despite seeming quite promising for our use case, this database system does not provide a crucial component to our system; tolerance to longer network partitions with later synchronization. Couchbase does provide some partition tolerance by being able to handle transactions despite being disconnected to parts of the system, however being cloud-native means that the system does not expect longer periods of isolation. This means that controlling the aspects of re-synchronization after network partitions is highly limited. Further the software expects to be run in a networked environment where the different part of the service can freely use bandwidth for synchronization purposes, which means that the system could potentially occupy much needed bandwidth in low-bandwidth radio-based networks with unessential, non-time-critical synchronization and replication traffic.

[7] CouchDB API enables an authorized administrator to send replication requests which specify what data to replicate and to which server. `https://docs.couchdb.org/en/stable/api/`

Conclusion Based on the comparison of CouchDB and Couchbase, we would recommend using CouchDB as the database back-end. Based on its ability to provide detailed control of database synchronization and replication, and further its merge conflict resolution enables it to properly fulfill the requirements highlighted in Section 4.4. Finally, it is in part what enables the entire system to satisfy the requirements regarding distributed operations.

4.6 Choosing a Tagging Solution

In this section we will discuss what tagging solution we ultimately recommend, and provide the basis for answering research question 2. The various tagging solutions we considered over the course of the project were RFID tags, QR code and barcode. Each of these will be discussed in their own paragraph below, followed by a conclusion.

Radio-frequency identification Initially the system was supposed to utilize RFID as a tagging solution. This is due to the benefit of bulk-scanning it brings. We also determined that RFID would not pose a security risk on field missions. This is due to RFID having a so limited range, that there would be other more effective solutions of tracking people. Additionally, due to the fact that scanners are emitting signals in order to detect tags, it would also be very easy to detect a rogue scanner.

However, we worried that the bulk-scanning capabilities of RFID might cause problems by accidentally scanning unrelated items. During our visit at SAAB we asked them if they had any experience with RFID tags for the purpose of equipment tracking. SAAB confirmed our suspicions. They reported issues with unintentional scanning of items, and issues with properly utilizing the bulk-scanning capabilities effectively (as summarized in Section 4.2.2).

Since we observed that many of the Norwegian Home Guard's locations have crammed storage facilities. Hence, we believe that unintentional bulk-scanning will be an even bigger issue for them. As such, we do not recommend RFID as a suitable solution for equipment tracking at the Norwegian Home Guard.

Quick Response code Another potential tagging solution is QR codes. We considered QR code codes as a good solution due to their error correction capabilities we mentioned earlier in Section 2.3.

However, during our visit at SAAB they informed us that the square form-factor of QR codes makes them hard to attach to smaller objects. They had particular issues with adding them to cables as these are both small and cylindrical. Dues to this we do not recommend using QR codes for the equipment tracking system at the Norwegian Home Guard.

Barcode The third potential tagging solution for our purposes are barcodes. Barcodes are, as we have previously discussed, widely used. This means that equip-

ment to both read and create barcodes should be easy to procure. This also means that most people should be familiar with the use of them.

Further, some military equipment comes with pre-existing barcodes already. Some radios we saw at our on-site visits had 2 barcodes attached already. A NSN identifying the type of equipment, and another with the radio's serial number.

We have not found any glaring issues with the use of barcodes for the purpose of equipment tracking. The only downsides we found were that different barcode specifications can only hold limited types of content. Additionally barcodes only have a limited practical capacity linked to their size. Neither of these restrictions are particularly relevant for the purposes of this proposed system. SAAB also did not highlight any issues they had experienced with barcodes and is currently using barcodes.

Conclusion In conclusion, we believe that barcodes are the best fit for our purposes. Barcodes are easy to read, widely known, and equipment for reading and creating barcodes is inexpensive and readily available. Both RFID and QR codes have substantial issues associated with their use. This is something barcodes do not share with them, as barcodes seem rather problem-free for the proposes of equipment tracking.

4.7 Software and Hardware Considerations

4.7.1 Software Architecture

In this section we will discuss some of the possible software implementations the system could use. The actual implementation is left for the Norwegian Home Guard to decide on and implement from the ground up, but we will highlight some different architectures that could be used.

It is important to note that regardless of architecture, the system will need to adhere to requirement 4 from Section 4.3, regarding it being a distributed system. This means that individual locations need to be able to function without connection to the rest of the system. In this case, this means that every location needs to have a local database keeping the storage data. During regular operations this database would communicate and synchronize with all other databases on the network.

Peer-to-Peer Peer-to-peer refers to an architecture where every node in a network directly communicates to every other node on the network, as seen in Figure 4.1[8]. For bigger networks with many nodes subgroups can be created so that any node only communicates with other nodes part of the same subgroup. This is to avoid having nodes communicate with hundreds of other nodes. When using CouchDB as a distributed database, as discussed in Section 4.5, then these would use a peer-to-peer architecture between themselves to keep the system synchronized.

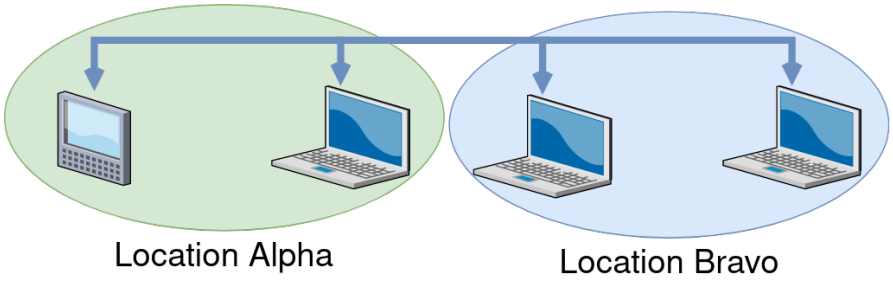


Figure 4.1: Peer-to-peer architecture for clients

A big advantage of the peer-to-peer architecture is that there is no single point of failure, as every node communicates with multiple other nodes directly.

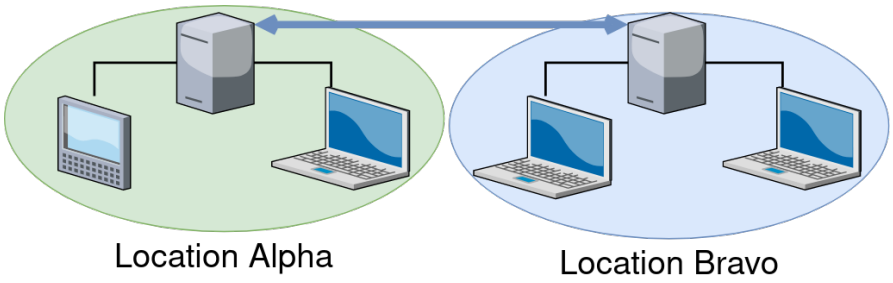


Figure 4.2: Local client-server architecture with server peer-to-peer connections

Client-Server The client-server architecture refers to a setup where there is one or more servers and a range of clients, as seen in Figure 4.2[8]. In these systems the clients are usually only interfaces for users to interact with. The clients then relay information from the user to the server, which then processes the request and sends a response back to the client. This leads to a scenario in which clients usually require less intensive hardware, since it is the server which processes the request.

In this specific case a server-client setup could be used locally at the various locations, where the server holds the CouchDB database and communicates to multiple local clients. In this case CouchDB would still synchronize itself with other servers via peer-to-peer connections. The advantage of using a local client-server setup would be that multiple clients can be used without each needing to synchronize its own version of the database. This would also allow for clients to require lower specifications then they otherwise would and potentially enables for handheld clients.

[8]The image was created using draw.io, a online drawing program. `https://draw.io`

4.7.2 Hardware

In this section we will discuss what kind of hardware requirements a system such as this might require. What hardware needs to be acquired is going to play a huge role in how economically viable the system will be. The economic impact of this will be discussed in Section 4.9.

Observations Regarding Existing Hardware During our visit at the Norwegian Home Guard we observed that there were entire shelves dedicated to retired computers. These computers have outlived their usefulness due to outdated specifications. Depending on just how outdated these computers are these could potentially be repurposed for our proposed system. This is particularly relevant if a local client-server architecture is used where the requirements of the clients would be comparatively lower than those of the server.

Hardware Requirements We will attempt to outline some potential hardware requirements for this proposed system. These will in reality be highly dependent on the finalized implementation. Every location will require either; one or more computers running the system in a purely peer-to-peer setup or one computer acting as a server and multiple clients.

The actual required specifications of the server and client computers will depend on the actual implementation of the finalized system. However, in a server-client setup the clients could potentially have low enough requirements for old existing hardware or slim clients such as a Raspberry Pi [16], a low cost, credit-card sized computer that plugs into a computer monitor, keyboard or similar.

Other hardware that might be required is as follows:

- Some for of network equipment to communicate with other locations.
- Computer monitors, unless the computers used have inbuilt displays.
- Scanners compatible with the chosen tagging solution.
- A printer, if legacy IS and US printouts are to be supported.

4.7.3 Updating and Patching the System

Like any system, flaws and other circumstances will requires the system to be updated from time to time. Since there are estimated to be around 750 locations that this system is going to be deployed to, there will be a significant amount of work involved in updating every one of the systems among the different locations. Additionally, there are huge differences in how and how often the systems will be used at different locations. Because of this the updating procedure needs to be feasible for both busy and remote locations. There are also locations with limited or no internet access, and many other edge cases that need to be considered.

After exploring different approaches to this problem, we found that the most sensible solution would be to have the systems default to updating over the internet, and then also have an alternate method of updating so that systems can

be updated even if they are totally isolated. An example would be to have the option to update via a USB flash drive and a password, or to manually download via radio. What we found to be important is that the system can be updated in multiple ways, depending on what works best for the individual locations.

It is also important that older versions of the system are compatible with other nodes running a newer version of it. This will enable separate locations to update their systems at separate times, without breaking connection or communication.

4.8 User Interface

In order to answer research question 3, we looked at how we could design some parts of the main user interface as a point of reference for future design decisions. Here we are focusing more on the user workflow, than the graphical design. The goal is to provide some suggestions and potential guidelines to help develop an effective and easy to use user interface.

For the system to be used efficiently, the user interface must be designed with properties that further ease of use in mind. Most importantly, it needs to be intuitive, flexible and streamlined. With an intuitive user interface, the path from one part of the procedure to the next should be clear. In a flexible user interface, it should be easy to change parameters and options related to the current step in the procedure. One should not need to search in unrelated other menus in order to change settings relating to the current step in the procedure. With streamlined user interface similar actions should be easily taken by similar means across the system. In a best case scenario a user who can use one part of the system should be able to use all parts of the system without much training.

We took a deep dive into the procedure of scanning items in and out of the system and made a process diagram in Figure 4.3. The diagram was created using `plantuml`[9]. Below are some words used in the diagram that needs some further explanation.

- **Scan** in this context means to input the information of one or more items into the system. This could either be done by scanning the barcode, or just writing the barcode number into a field manually.
- **Parking** is a term of a non-committed delivery plan. It can be used to create plans for what each unit that is getting equipment is getting, without thinking about specific serial numbers etc.
- **Metadata** is various data that identifies who is getting equipment, what the equipment is to be used for (usually what exercise) as well as potentially some time frame of when you are expecting this equipment to be returned.

It is important to note the black bars in Figure 4.3, as it shows the streamline capabilities of the system. This help the system to be easy to use as you are familiar with the steps from the other interactions you have had with other parts of the

[9]PlantUML is a diagram generation tool using text files as input. `https://plantuml.com/`

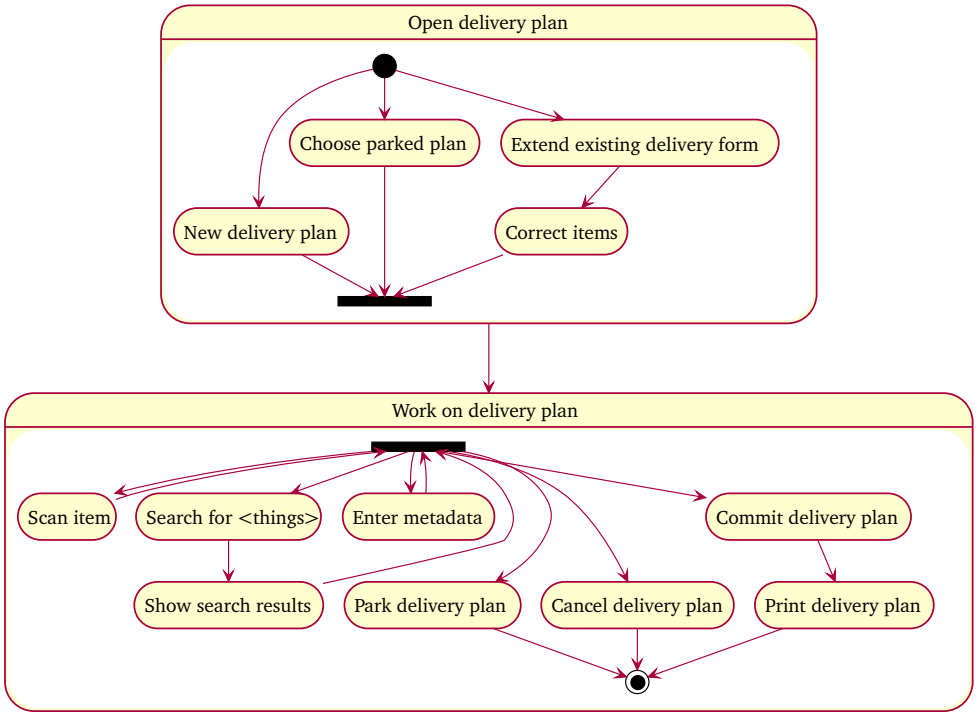


Figure 4.3: A process diagram for storage movements

system. This enables the user to build up an intuition for how to perform tasks within the systems. For example, there are multiple ways to open a plan, but once you have opened a plan there is no way to open another plan before somehow closing the current one.

In this model you will use “extend existing delivery plan” to hand in equipment as well as replacing broken equipment. In the process of handing in items they should be flagged as ’in need of repair’ if they need any maintenance before being handed out again. If you just hand in you commit that delivery plan, while if you need to hand out equipment to replace broken ones you continue scanning like if you were handing out equipment in the normal way. This process is used the same if wrong items were handed out. This could also be used when moving equipment between locations. You can hand out the equipment from that location, and scan it in to that location.

Some other thoughts we have arrived at regarding history inspection. There should be a way to create customized queries for what you are searching for example. Like find all transactions that happened in regard with this exercise, or find all equipment that has was handed out at location X, to the unit(s) that is older than two weeks old. This should probably be done interactively so each step that shows the results that you need. There might also be a need for saving some of these as they might want to be used in the future.

There should also be a setting to choose between a light and a dark theme.

We expect there to be quite different lighting conditions for the locations and this will make the system more adaptable.

4.9 Economic Factors

In this section we will highlight some economic considerations which should be addressed when implementing the system. The goal is not to provide an extensive list of all economic factors, but rather highlight a few key considerations in accordance with research question 4.

The Norwegian Home Guard is looking for a system which is economically viable, as outlined in requirement 9 in Section 4.3. This constraint stems from the fact that they have quite a limited budget already, and would rather allocate funds to military exercises. This means that one of our main foci will be to research what factors will contribute in getting a good cost to value ratio of the system.

In the following list we will highlight some key considerations and how they will affect the economic cost of the system.

Using existing hardware or acquiring new hardware

Utilizing existing hardware

- will require more development time due to the necessity of portability and platform independence.
- will require more effort during deployment due to heterogeneous environments.
- will require more effort during maintenance due to the varying maintenance needs. There is an increased risk of possible performance and security issues due to outdated software on the different host systems. Extensive maintenance can lead to downtime and reduce the efficiency of the system.
- will reduce hardware costs when deploying the system.
- will possibly lower the initial costs of setting up the system, and hence make it a smaller upfront investment.

Acquiring new homogeneous hardware

- will lower development costs due to the lacking requirement of portability.
- will simplify deployment and maintenance of the system due to homogeneous environments.
- will require a bigger upfront investment, which will increase the consequences of a failed project.
- might disregard differing needs among storage facilities, where some might require more infrastructure or equipment to operate effectively.

Platform dependence of the software

Developing a platform independent application will require more development time due to the increased complexity. However, platform independence is important if existing hardware is used, as different locations might have different hardware and platforms. Platform independence will also be more demanding to maintain and hence, this will be more expensive.

Investing in project and software design

We would also like to highlight the economic impact of project and software design. In his recent work, Löwy [17] has shown how to design both projects and systems which are resilient to change. By following Löwy's advice on designing a system which encapsulates volatility, the inevitable future changes to the requirements will be easier and cheaper to handle. This will also enable the system to remain relevant further into the future.

By investing early on in design and permitting a longer development period, the developers can focus on creating a extensible core for the system. Despite an increased initial development cost, this will normally substantially reduce the cost of maintenance and further development of the system.

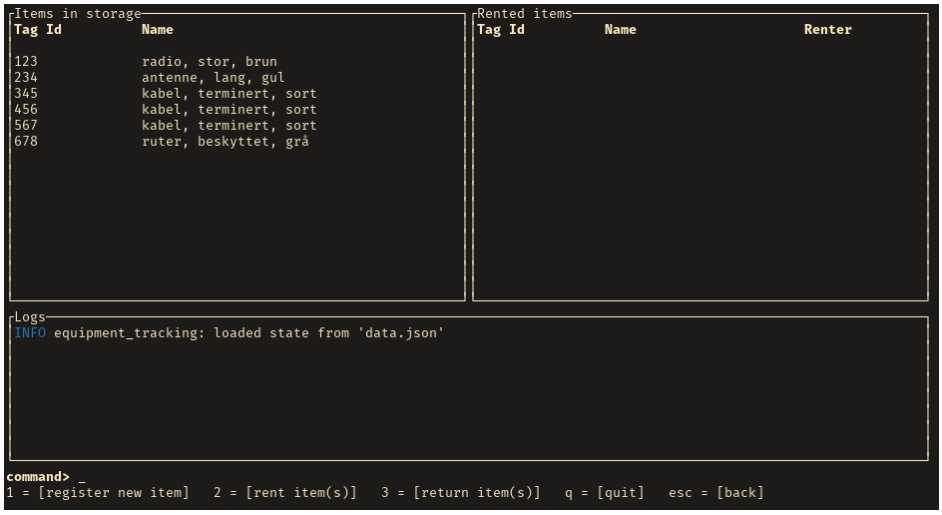
4.10 Proof of Concept

As a part of the research, we have developed a proof of concept of the client application. The application demonstrates the feasibility of a few of the recommendations we made in earlier chapters, specifically targeting user interface, hardware and platform considerations. The code of the PoC can be found in Appendix B.

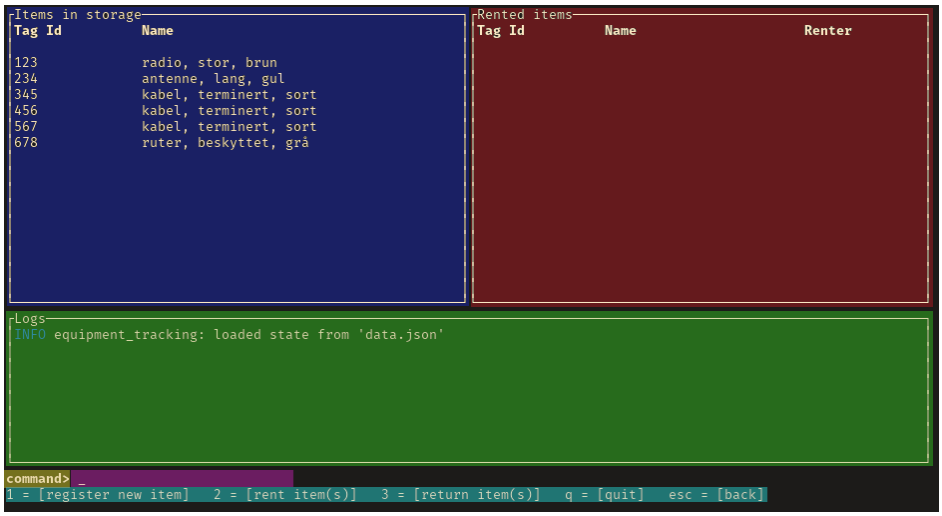
This application is developed on the premise that it should be runnable across multiple platforms without requiring a graphical user environment. The main reason for building a text-based user interface is that it is less demanding on the underlaying hardware. Further, text-based user interfaces are often more portable than graphical based user environments. Another consideration is that the application is controllable using only a barcode scanner or a keyboard.

Overview of the user interface In Figure 4.4 we can see the user interface of the application. For further explanation of the various sections of the interface, see Figure 4.4b. As mentioned in the previous paragraph, the interface is text-based and very simplistic. This interface might be novel for users which are used to graphical user interfaces from modern operating systems. However, after overcoming the initial adoption period, we believe the interface will be both quicker and easier to use than traditional graphical interfaces.

Walk-through of renting out items In Figure 4.5 we can see a walk-through of renting out items using the application. One key observation was that users often



(a) The plain appearance of the interface

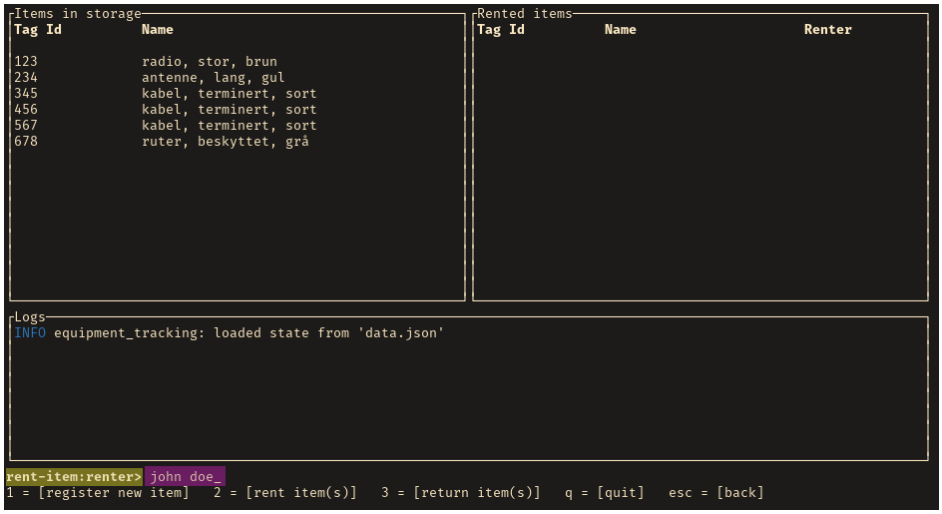


(b) The interface including colored sections for visibility. The blue section shows the items that are currently in storage, the red section shows items that are rented to users, the green section shows application logs including errors and informational messages, the yellow section is the command prompt which shows what is currently requested, the purple section is the input field and the cyan section is the command help text.

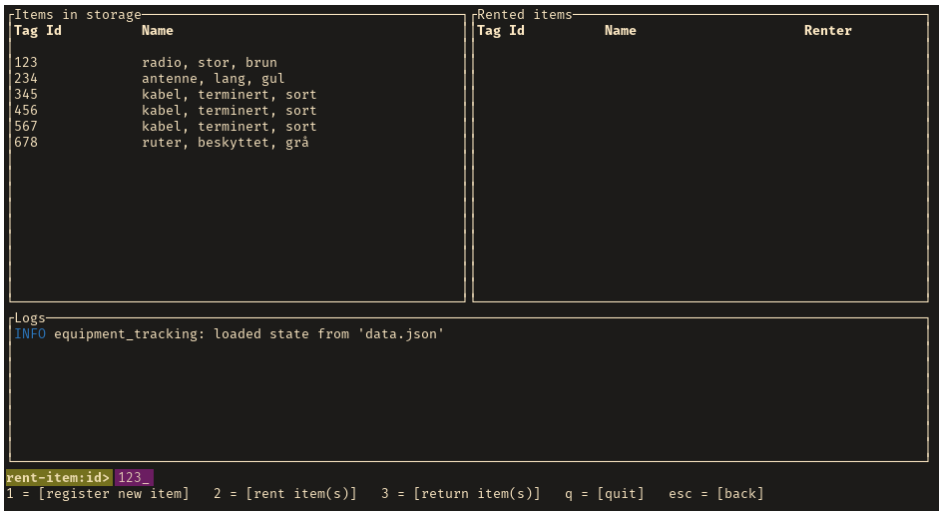
Figure 4.4: Overview of the text-based user interface for the PoC

rent multiple items. Hence by supplying the renter's identifier first, we can keep on prompting the operator for more item identifiers. This means that the process is streamlined for handing out multiple items to one renter at a time.

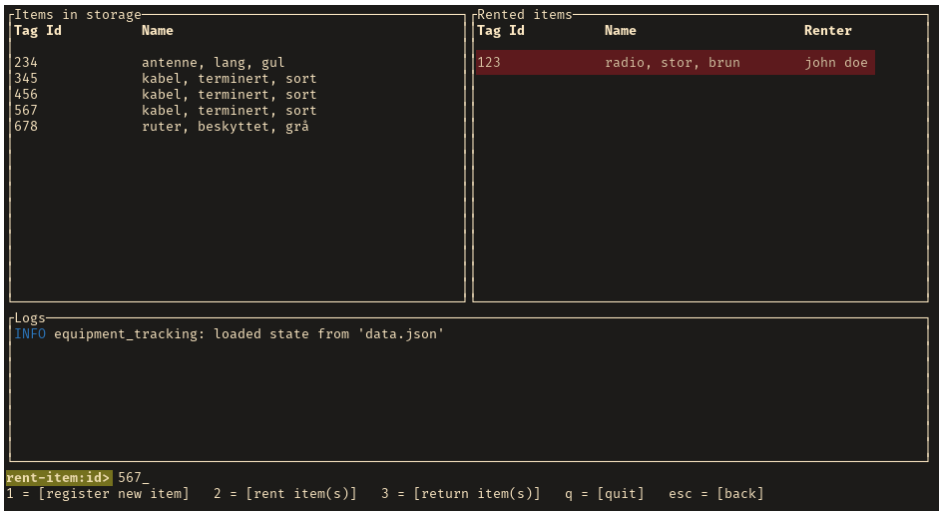
Another key observation is that all parts of the process are completed without the use of a mouse. Instead this is accomplished by using the barcode scanner as a keyboard device. This means that the operator can control the entire application by scanning specially crafted barcodes. In a military context, this means that the operator does not have to remove their gloves, which would be especially beneficial in cold environments.



(a) Prompt for renter's name after selecting "rent item(s)". The command prompt (yellow section) shows a terse description of what is requested from the operator. In this case the operator has already entered the name of the renter in the input field (purple section).



(b) Prompt for item identifier after selecting "rent item(s)". The command prompt (yellow section) now shows that it's requesting a item identifier, and the operator has already entered "123".



(c) Prompt for another item identifier after confirming previous item. The confirmed item has now moved to "Rented items" (red section). To enable the operator to rent out multiple items for a single renter quickly, we prompt again for an item identifier.

Figure 4.5: A walk-through of the process of renting items in this PoC

Chapter 5

Conclusion

5.1 General Conclusion

In this project we claimed that it is feasible to create a distributed equipment tracking system for the Norwegian Home Guard which satisfies the requirements outlined in Section 1.2 and then further refined in Section 4.3.

In order to do prove our claim, we first researched existing solutions and some of the underlying technologies we intended to use. By doing this we showed that no existing solution is suited to act as the Norwegian Home Guard's equipment tracking system. We also found CouchDB to be a valid database back-end for this system. With the help of SAAB's experience, we also concluded that the most suitable equipment tagging solution for this project would be barcode. We also explored the economical implications of such a system and it's requirements. Lastly, we created a Proof of concept to demonstrate the feasibility of a suggested user interface and it's platform independence.

We have not found any conflicting requirements or otherwise contradictory evidence to these claims. As such, we will recommend the Norwegian Home Guard to continue the development process.

5.2 Answering the Research Questions

5.2.1 Question 1

How does Brewer's Conjecture [1] — also know as CAP theorem — affect the proposed distributed equipment tracking system?

In general we found that Brewer's Conjecture forces the system to chose between consistency, availability and partition tolerance. Based on requirement 4 we have to choose availability and partition tolerance. However, because our system will be partially synchronous we chose a weaker form for consistency, called eventual consistency. The recommended database CouchDB supports this form of distributiveness.

This question was discussed in Section 4.4.

5.2.2 Question 2

What are the key advantages and disadvantages of different equipment tagging solutions?

We have researched barcode, QR code and RFID technologies.

Barcodes only unique downsides is that there are restrictions to what type of data certain barcode specifications can store and its limited practical capacity. None of these downsides are relevant for the proposed system.

QR code has the advantage of being fault tolerant and can store relatively large amounts of data in a small surface area. Nonetheless, QR code readers are usually more expensive than barcode scanners while at the same time being unreliable when scanning smaller QR codes. Because of this it is hard to attach QR codes to smaller items such as cables while still remaining readable.

RFID was promising due to its bulk-scanning capabilities. However, it was hard to correctly scan only the intended tags.

This question was discussed in Section 4.6.

5.2.3 Question 3

How can the user interface best ensure ease of use and help to prevent erroneous usage?

Despite exploring this question we have not deem our answer complete, yet we have found some key considerations. The user interface shall be able be operable without the need of a pointing device. Further, the user interface shall be as simplistic and streamlined as possible.

This question was discussed in Section 4.8.

5.2.4 Question 4

What are the main economic factors to consider when creating, deploying, using, maintaining and decommissioning the equipment tracking system?

The main economic factors to consider are;

- Using existing hardware or acquiring new hardware
- Platform dependence of the software
- Investing in project and software design

This question was discussed in Section 4.9.

5.3 Future Work

The following is a list of topics that require some additional research outside of the normal development process.

1. How to create the tags. These need to be printed and likely also need some form of protective coating.

2. How barcodes are attached to equipment. Some research needs to be done into what the best way to attach the codes to the equipment is.

Bibliography

[1] S. Gilbert and N. Lynch, 'Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services', *SIGACT News*, vol. 33, no. 2, pp. 51–59, Jun. 2002, ISSN: 0163-5700. DOI: 10.1145/564585.564601. [Online]. Available: https://doi.org/10.1145/564585.564601.

[2] A. S. Tanenbaum, *Distributed systems : principles and paradigms*, eng. Upper Saddle River, N.J: Prentice Hall, 2007, ISBN: 0132392275.

[3] P. A. Bernstein and N. Goodman, 'Concurrency control in distributed database systems', *ACM Comput. Surv.*, vol. 13, no. 2, pp. 185–221, Jun. 1981, ISSN: 0360-0300. DOI: 10.1145/356842.356846. [Online]. Available: https://doi.org/10.1145/356842.356846.

[4] R. Systems. (2020). Wise wms – warehouse management system, [Online]. Available: https://www.royal4.com/warehouse-management-system-solutions/wise-wms/ (visited on 05/03/2020).

[5] Infoplus. (2020). Infoplus warehouse manager, [Online]. Available: https://www.infopluscommerce.com/products/wms (visited on 05/03/2020).

[6] Brightpearl. (2020). Advanced inventory management for retailers and wholesalers, [Online]. Available: https://info.brightpearl.com/inventory-management-software (visited on 10/03/2020).

[7] Katana. (2020). Get in the driver's seat of your product-making business., [Online]. Available: https://katanamrp.com/ (visited on 10/03/2020).

[8] Cherwell. (2020). Itsm software that's easy to use, configure, and maintain, [Online]. Available: https://www.cherwell.com/products/it-service-management/ (visited on 12/03/2020).

[9] Wikipedia. (2020). Barcode, [Online]. Available: https://en.wikipedia.org/wiki/Barcode (visited on 15/03/2020).

[10] 'Information technology — automatic identification and data capture techniques — qr code bar code symbology specification', en, Standard ISO/IEC 18004:2015, 2015. [Online]. Available: https://www.iso.org/standard/62021.html.

[11] K. Krombholz, P. Fruehwirt, P. Kieseberg, I. Kapsalis, M. Huber and E. Weippl, 'Qr code security: A survey of attacks and challenges for usable security', in *Human Aspects of Information Security, Privacy, and Trust*. Springer, 2014, pp. 79–90.

[12] 'Information technology — radio frequency identification for item management — part 6: Parameters for air interface communications at 860 mhz to 960 mhz general', en, Standard ISO/IEC 18000-6:2013, 2013. [Online]. Available: https://www.iso.org/standard/59644.html.

[13] R. Zajonc, 'Mere exposure: A gateway to the subliminal', eng, *Current Directions in Psychological Science*, vol. 10, no. 6, pp. 224–228, 2001, ISSN: 0963-7214.

[14] A. S. Foundation. (2020). Apache couchdb® 2.3.1 documentation, [Online]. Available: https://docs.couchdb.org/en/stable/index.html (visited on 23/02/2020).

[15] Couchbase. (2020). Couchbase nosql database, [Online]. Available: https://www.couchbase.com/ (visited on 02/03/2020).

[16] R. P. Foundation. (2020). Raspberry pi 4, [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-4-model-b/ (visited on 10/05/2020).

[17] J. Löwy, *Righting Software, A Method for System and Project Design*, eng, 1st ed. Addison-Wesley Professional, Dec. 2019, ISBN: 0136524036. [Online]. Available: https://rightingsoftware.org/.

Appendix A

Delivery and Submission Form

INNLEVERINGSSEDDEL

	FAGM		REG DATO
INNLEVERES TIL:		UTLÅN	AVD KONTR.NR
INNLEVERES FRA:	REGNSK.NR		ARB ORDRENR
HJEMMEL			

POST	KATALOG ELLER PARTNR	BENEVNING	ENHET	MENGDE	MOTTATT/ KODE

	INNLEVERING GODKJENT:		FOR AVDELINGSSJEFEN:
DATO	FOR FORSYNINGSSJEFEN	DATO	AVDELINGENS FORSYNINGSOFFISER
	MOTTATT:		
DATO	AUT REPR FOR MAG/LAGER	BB - BRUKBAR FM - FOR MEGE VS - VANLIG SLITASJE	TS - TAPS OG SKADEM. OB - OPPGJØRSBL.

LIK GJENST. LEV.INN PÅ BILNR	POST DATO	BILAGSNR

UTLEVERINGSSEDDEL

	FAGM		REG DATO
UTLEVERES FRA:	UTLEV	UTLÅ x	AVD KONTR.NR
UTLEVERES TIL:	REGNSK.NR		ARB ORDRENR
HJEMMEL			

POST	KATALOG ELLER PARTNR	BENEVNING	ENHET	MENGDE	MOTTATT/ KODE
1					
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14	**SISTE PUNKT**	**SISTE PUNKT**		**SISTE PUNKT**	
15					

	UTLEVERING GODKJENT:	FOR AVDELINGSSJEFEN:	
DATO 0	FOR FORSYNINGSSJEFEN	DATO	AVDELINGENS FORSYNINGSOFFISER
	UTLEVERT:	MOTTATT:	
DATO	AUT REPR FOR MAG/LAGER	DATO	AUT REPR FOR AVDELINGEN

LIK GJENST. LEV.INN PÅ BILNR	POST DATO	BILAGSNR VHF

Appendix B

Code the of Proof of Concept

This PoC is developed using Rust[1], a modern systems programming language. Rust was chosen due to it's great cross-platform support, simple dependency management and ability to create statically compiled binaries. The latter part means that the PoC can be compiled into a single binary which can easily be copied to other compatible systems.

Instructions for how to build and run the project can be found below.

[1] https://www.rust-lang.org/

A simple PoC for equipment tracking

- Should run in any terminal (cmd, bash etc.), even old hardware.
- Enables simple handling by keeping items in storage and rented out.
- Very simple user interface which can be operated without a mouse.
- Extensible through component-based storage.

Running

To run the application the user needs to have `Rust` installed. Instructions can be found here.

When `Rust` is installed, simply run:

```
cargo run
```

Using test-data

To run the application with test-data, simply copy `test-data.json` to `data.json`.

```
cp test-data.json data.json
```

Todo

- ☒ Basic logging and user interface setup
- ☒ Store items with metadata and location
- ☒ Rent items to users
- ☒ Return items from users
- ☒ Add instructions and test data
- ☒ Store state to a file (simple persistent storage)
- ☐ Store event-sourced state to CouchDB (complex persistent storage)
  - ☐ Setup connection to CouchDB
  - ☐ Upload state changes to CouchDB (use `FlaggedStorage` from `specs`)
  - ☐ Reassemble state based on change events from CouchDB

Cargo.toml

```
[package]
name = "equipment-tracking"
version = "0.1.0"
authors = ["Ole Martin Ruud <dev@barskern.no>"]
edition = "2018"

[dependencies]
anyhow = "1.0.28"
arraydeque = "0.4.5"
crossterm = { version = "0.17", features = ["event-stream"] }
futures = "0.3.4"
log = { version = "0.4.8", features = ["std"] }
serde = { version = "1.0.106", features = ["derive"] }
serde_json = "1.0.52"
smol = "0.1.4"
specs = { version = "0.16.1", features = ["uuid_entity", "specs-derive", "serde"] }
tui = { version = "0.9", default-features = false, features = ['crossterm'] }
```

controller.rs

```rust
use crossterm::event::{Event as CEvent, KeyCode, KeyEvent};
use log::{debug, error, warn};
use specs::prelude::*;
use specs::saveload::{MarkedBuilder, UuidMarker};

use crate::model::menu::{
    RegisterItemStatus, RentItemStatus, ReturnItemStatus, Status as MenuStatus,
};
use crate::model::{Location, Metadata, State, TagId};

#[derive(Debug)]
pub enum Status {
    Exit,
    Continue,
}

pub fn on_event(state: &mut State, event: CEvent) -> Status {
    log::debug!("Recieved event: {:?}", event);

    match event {
        CEvent::Key(KeyEvent {
            code: KeyCode::Char('q'),
            ..
        }) if state.menu_status == MenuStatus::Main => Status::Exit,
        CEvent::Key(KeyEvent {
            code: KeyCode::Char(c),
            ..
        }) => {
            state.input.push(c);
            Status::Continue
        }
        CEvent::Key(KeyEvent {
            code: KeyCode::Backspace,
            ..
        }) => {
            state.input.pop();
            Status::Continue
        }
        CEvent::Key(KeyEvent {
            code: KeyCode::Enter,
            ..
        }) => {
            debug!("confirmed: {}", state.input);

            match state.menu_status {
                MenuStatus::Main => match &*state.input {
                    "1" => {
                        debug!("requested to register new item");
                        state.menu_status = MenuStatus::RegisterItem(
                            RegisterItemStatus::InsertItemId,
                        );
                    }
```

```
        "2" => {
            debug!("requested to rent item");
            state.menu_status =
                MenuStatus::RentItem(RentItemStatus::InsertRenter);
        }
        "3" => {
            debug!("requested to return item");
            state.menu_status = MenuStatus::ReturnItem(
                ReturnItemStatus::InsertItemId,
            );
        }
        _ => warn!("command not found"),
    },
    MenuStatus::RegisterItem(ref reg_status) => match reg_status {
        RegisterItemStatus::InsertItemId => {
            let id = state.input.clone();

            if state
                .world
                .read_storage::<TagId>()
                .join()
                .find(|tag_id| tag_id.0 == id)
                .is_none()
            {
                state.menu_status = MenuStatus::RegisterItem(
                    RegisterItemStatus::InsertItemName { id },
                );
            } else {
                error!("item with id '{}' already exists", id);
            }
        }
        RegisterItemStatus::InsertItemName { id } => {
            let _item = state
                .world
                .create_entity()
                .with(TagId(id.clone()))
                .with(Location::InStorage)
                .with(Metadata {
                    name: state.input.clone(),
                })
                .marked::<UuidMarker>()
                .build();

            state.menu_status = MenuStatus::Main;
        }
    },
    MenuStatus::RentItem(ref rent_status) => match rent_status {
        RentItemStatus::InsertRenter => {
            let renter = state.input.clone();

            state.menu_status = MenuStatus::RentItem(
                RentItemStatus::InsertItemId { renter },
            );
        }
```

```
        RentItemStatus::InsertItemId { renter } => {
            let id = &state.input;

            if let Some(item) = state.world.exec(
                |(entities, tag_ids): (
                    Entities,
                    ReadStorage<TagId>,
                )| {
                    (&entities, &tag_ids)
                        .join()
                        .find(|(_, tag_id)| &tag_id.0 == id)
                        .map(|(e, _)| e)
                },
            ) {
                if let Some(loc) = state
                    .world
                    .write_storage::<Location>()
                    .get_mut(item)
                {
                    debug!(
                        "item with id '{}' at {:?} rented to {}",
                        id, loc, state.input
                    );
                    *loc = Location::RentedTo {
                        renter: renter.clone(),
                    };
                } else {
                    warn!("item with id '{}' does not have a location", id);
                }
            } else {
                error!("there exists no item with id '{}'", id);
            }
        }
    },
    MenuStatus::ReturnItem(ref return_status) => {
        match return_status {
            ReturnItemStatus::InsertItemId => {
                let id = &state.input;

                if let Some(item) = state.world.exec(
                    |(entities, tag_ids, locs): (
                        Entities,
                        ReadStorage<TagId>,
                        ReadStorage<Location>,
                    )| {
                        (&entities, &tag_ids, &locs)
                            .join()
                            .find(|(_, tag_id, loc)| {
                                &tag_id.0 == id
                                    && !matches!(
                                        loc,
                                        Location::InStorage
                                    )
                            })
```

```
                                        .map(|(e, _, _)| e)
                                },
                            ) {
                                if let Some(loc) = state
                                    .world
                                    .write_storage::<Location>()
                                    .get_mut(item)
                                {
                                    debug!(
                                        "item '{}' returned from {:?}",
                                        id, loc
                                    );
                                    *loc = Location::InStorage;
                                } else {
                                    warn!(
                                        "item '{}' does not have a location",
                                        id
                                    );
                                }
                            } else {
                                error!(
                                "there exists no item with id '{}' which is not in storage",
                                id
                            );
                            }
                        }
                    }
                }
            }

            state.input.clear();
            Status::Continue
        }
        CEvent::Key(KeyEvent {
            code: KeyCode::Esc, ..
        }) => {
            state.menu_status = MenuStatus::Main;
            state.input.clear();
            Status::Continue
        }
        _ => Status::Continue,
    }
}

pub fn on_tick(_state: &mut State) {}
```

logging.rs

```rust
use anyhow::Result;
use arraydeque::{behavior::Wrapping, ArrayDeque};
use log::*;
use std::sync::Arc;
use std::sync::Mutex;
use tui::buffer::Buffer;
use tui::layout::Rect;
use tui::style::{Color, Style};
use tui::widgets::Text;
use tui::widgets::Widget;

pub fn init(level: LevelFilter) -> Result<TuiLogState> {
    let logger = Box::new(TuiLogger {
        logs: Arc::new(Mutex::new(ArrayDeque::new())),
    });

    let log_ref = TuiLogState {
        logs: Arc::clone(&logger.logs),
    };

    set_boxed_logger(logger)?;
    set_max_level(level);

    Ok(log_ref)
}

#[derive(Debug)]
pub struct TuiLogState {
    logs: Arc<Mutex<ArrayDeque<[Text<'static>; 256], Wrapping>>>,
}

#[derive(Debug)]
pub struct TuiLogWidget<'s> {
    pub state: &'s TuiLogState,
}

impl Widget for TuiLogWidget<'_> {
    fn render(self, area: Rect, buf: &mut Buffer) {
        use tui::style::*;
        use tui::widgets::*;

        let logs = self.state.logs.lock().expect("logs lock posioned");

        let block = Block::default().title("Logs").borders(Borders::ALL);
        let p = Paragraph::new(logs.iter())
            .block(block)
            .style(Style::default().fg(Color::White).bg(Color::Black))
            .wrap(true);

        p.render(area, buf);
    }
}
```

```rust
#[derive(Debug)]
struct TuiLogger {
    // TODO maybe use something else than a mutex here??
    logs: Arc<Mutex<ArrayDeque<[Text<'static>; 256], Wrapping>>>,
}

impl Log for TuiLogger {
    fn enabled(&self, _metadata: &Metadata) -> bool {
        true
    }

    fn log(&self, record: &Record) {
        if !self.enabled(record.metadata()) {
            return;
        }

        let (label, color) = match record.level() {
            Level::Error => ("ERROR", Color::Red),
            Level::Warn => ("WARN", Color::Yellow),
            Level::Info => ("INFO", Color::Blue),
            Level::Debug => ("DEBUG", Color::Magenta),
            Level::Trace => ("TRACE", Color::Gray),
        };

        let mut logs = self.logs.lock().expect("logs lock posioned");
        logs.push_front(Text::raw(format!(
            " {}: {}\n",
            record.target(),
            record.args()
        )));
        logs.push_front(Text::Styled(label.into(), Style::new().fg(color)));
    }

    fn flush(&self) {}
}
```

main.rs

```rust
use anyhow::Result;
use futures::{future, select};
use futures::{FutureExt, StreamExt};
use log::{debug, error, info};
use specs::prelude::*;
use std::io::{self, Write};
use std::thread;
use std::time::Duration;
use tui::backend::CrosstermBackend;
use tui::Terminal;

mod logging;

mod controller;
mod model;
mod view;

use crate::controller::Status;

fn main() -> Result<()> {
    // Setup terminal for application
    crossterm::terminal::enable_raw_mode()?;
    let mut stdout = io::stdout();
    crossterm::execute!(stdout, crossterm::terminal::EnterAlternateScreen)?;
    let backend = CrosstermBackend::new(stdout);
    let mut terminal = Terminal::new(backend)?;
    terminal.hide_cursor()?;

    // Setup executors and work pool (required for using block_on later)
    for _ in 0..4 {
        thread::spawn(|| smol::run(future::pending::<()>()));
    }

    let log_state = logging::init(log::LevelFilter::Info)?;

    let mut state = model::State::new(log_state);

    {
        use crate::model::{Location, Metadata, TagId};
        use specs::prelude::*;
        use specs::saveload::{
            DeserializeComponents, UuidMarker, UuidMarkerAllocator,
        };
        use std::fs::File;

        debug!("trying to load state from 'data.json'");
        if let Ok(file) = File::open("data.json") {
            let mut de = serde_json::Deserializer::from_reader(file);

            let (ents, mut markers, mut alloc, tags, metas, locs): (
                Entities,
                WriteStorage<UuidMarker>,
```

```
            Write<UuidMarkerAllocator>,
            WriteStorage<TagId>,
            WriteStorage<Metadata>,
            WriteStorage<Location>,
        ) = state.world.system_data();

        match DeserializeComponents::<anyhow::Error, _>::deserialize(
            &mut (tags, metas, locs),
            &ents,
            &mut markers,
            &mut alloc,
            &mut de,
        ) {
            Ok(()) => info!("loaded state from 'data.json'"),
            Err(e) => error!(
                "unable to read or parse data 'data.json' storage: {}",
                e
            ),
        }
    }
}

// Generate an infinite stream of tick events to update UI (doesn't need to be fused because it
// will never return `None`)
let mut ticks = futures::stream::unfold((), |_| {
    // 16 ms per frame means roughly 60 FPS
    smol::Timer::after(Duration::from_millis(16))
        .map(|time| Some((time, ())))
});

// Get input events from crossterm (must be fused to be used in select)
let mut terminal_events = crossterm::event::EventStream::new().fuse();

// Run main application
let app_result: Result<()> = smol::block_on(async {
    loop {
        select! {
            event = terminal_events.select_next_some() => {
                match event {
                    Ok(event) => if let Status::Exit = controller::on_event(&mut state, event) {
                        return Ok(());
                    },
                    Err(e) => log::error!("unable to get terminal event: {}", e),
                }
            }
            tick = ticks.select_next_some() => {
                state.world.maintain();
                controller::on_tick(&mut state);
                terminal.draw(|mut f| { view::draw(&state, f) })?;
            }
        }
    }
});
```

```
    {
        use crate::model::{Location, Metadata, TagId};
        use specs::saveload::{SerializeComponents, UuidMarker};
        use std::fs::File;

        let mut file = File::create("data.json")?;
        let mut ser = serde_json::Serializer::new(&mut file);

        let (ents, markers, tags, metas, locs): (
            Entities,
            ReadStorage<UuidMarker>,
            ReadStorage<TagId>,
            ReadStorage<Metadata>,
            ReadStorage<Location>,
        ) = state.world.system_data();

        let _ = SerializeComponents::<anyhow::Error, UuidMarker>::serialize(
            &(&tags, &metas, &locs),
            &ents,
            &markers,
            &mut ser,
        );
    }

    // Reset terminal after application exits
    terminal.show_cursor()?;
    crossterm::execute!(
        terminal.backend_mut(),
        crossterm::terminal::LeaveAlternateScreen
    )?;
    crossterm::terminal::disable_raw_mode()?;

    app_result
}
```

menu.rs

```rust
use std::borrow::Cow;

#[derive(PartialEq, Debug)]
pub enum Status {
    Main,
    RegisterItem(RegisterItemStatus),
    RentItem(RentItemStatus),
    ReturnItem(ReturnItemStatus),
}

impl Status {
    pub fn prompt(&self) -> Cow<'static, str> {
        match self {
            Status::Main => "command".into(),
            Status::RegisterItem(r) => r.prompt(),
            Status::RentItem(r) => r.prompt(),
            Status::ReturnItem(r) => r.prompt(),
        }
    }
}

#[derive(PartialEq, Debug)]
pub enum RegisterItemStatus {
    InsertItemId,
    InsertItemName { id: String },
}

impl RegisterItemStatus {
    pub fn prompt(&self) -> Cow<'static, str> {
        match self {
            RegisterItemStatus::InsertItemId => "register-item:id".into(),
            RegisterItemStatus::InsertItemName { .. } => {
                "register-item:name".into()
            }
        }
    }
}

#[derive(PartialEq, Debug)]
pub enum RentItemStatus {
    InsertRenter,
    InsertItemId { renter: String },
}

impl RentItemStatus {
    pub fn prompt(&self) -> Cow<'static, str> {
        match self {
            RentItemStatus::InsertRenter => "rent-item:renter".into(),
            RentItemStatus::InsertItemId { .. } => "rent-item:id".into(),
        }
    }
}
```

```rust
#[derive(PartialEq, Debug)]
pub enum ReturnItemStatus {
    InsertItemId,
}

impl ReturnItemStatus {
    pub fn prompt(&self) -> Cow<'static, str> {
        match self {
            ReturnItemStatus::InsertItemId => "return-item:id".into(),
        }
    }
}
```

mod.rs

```rust
use serde::{Deserialize, Serialize};
use specs::prelude::*;
use specs::saveload::{UuidMarker, UuidMarkerAllocator};
use specs::Component;

use crate::logging::TuiLogState;

pub mod menu;

pub struct State {
    pub input: String,
    pub menu_status: menu::Status,
    pub log_state: TuiLogState,

    pub world: World,
}

impl State {
    pub fn new(log_state: TuiLogState) -> Self {
        let mut world = World::new();
        world.register::<UuidMarker>();
        world.register::<TagId>();
        world.register::<Location>();
        world.register::<Metadata>();

        world.insert(UuidMarkerAllocator::new());

        Self {
            input: String::with_capacity(64),
            log_state,
            menu_status: menu::Status::Main,

            world,
        }
    }
}

#[derive(
    Component, Serialize, Deserialize, Hash, PartialEq, Eq, Clone, Debug,
)]
pub struct TagId(pub String);

#[derive(Component, Serialize, Deserialize, PartialEq, Clone, Debug)]
#[serde(tag = "type", content = "data", rename_all = "SCREAMING_SNAKE_CASE")]
pub enum Location {
    InStorage,
    RentedTo { renter: String },
}

#[derive(Component, Serialize, Deserialize, PartialEq, Clone, Debug)]
pub struct Metadata {
    pub name: String,
```

```
}
```

view.rs

```rust
use specs::prelude::*;
use tui::{backend::Backend, layout::*, style::*, widgets::*, Frame};

use crate::logging::TuiLogWidget;
use crate::model::{Location, Metadata, State, TagId};

pub fn draw<B: Backend>(state: &State, mut frame: Frame<B>) {
    // Calculate main vertical layout division
    let chunks = Layout::default()
        .direction(Direction::Vertical)
        .constraints(
            [
                Constraint::Min(10),
                Constraint::Length(10),
                Constraint::Length(1),
                Constraint::Length(1),
            ]
            .as_ref(),
        )
        .split(frame.size());

    {
        let storage_view = Layout::default()
            .direction(Direction::Horizontal)
            .constraints(
                [Constraint::Percentage(50), Constraint::Percentage(50)]
                    .as_ref(),
            )
            .split(chunks[0]);

        let (ids, metas, locs) = state.world.system_data::<(
            ReadStorage<TagId>,
            ReadStorage<Metadata>,
            ReadStorage<Location>,
        )>();

        {
            let items_in_storage = (&ids, &metas, &locs)
                .join()
                .filter(|(_, _, loc)| **loc == Location::InStorage)
                .map(|(id, meta, _)| {
                    Row::Data(vec![&id.0, &meta.name].into_iter())
                });

            let in_storage_table =
                Table::new(["Tag Id", "Name"].iter(), items_in_storage)
                    .block(
                        Block::default()
                            .title("Items in storage")
                            .borders(Borders::ALL),
                    )
                    .header_style(Style::default().modifier(Modifier::BOLD))
```

```
                .widths(&[
                    Constraint::Length(15),
                    Constraint::Percentage(100),
                ]);
            frame.render_widget(in_storage_table, storage_view[0]);
        }
        {
            let rented_items = (&ids, &metas, &locs)
                .join()
                .filter_map(|(id, meta, loc)| match loc {
                    Location::RentedTo { renter } => Some((id, meta, renter)),
                    _ => None,
                })
                .map(|(id, meta, renter)| {
                    Row::Data(vec![&id.0, &meta.name, &renter].into_iter())
                });

            let rented_items_table =
                Table::new(["Tag Id", "Name", "Renter"].iter(), rented_items)
                    .block(
                        Block::default()
                            .title("Rented items")
                            .borders(Borders::ALL),
                    )
                    .header_style(Style::default().modifier(Modifier::BOLD))
                    .widths(&[
                        Constraint::Length(15),
                        Constraint::Percentage(100),
                        Constraint::Length(15),
                    ]);
            frame.render_widget(rented_items_table, storage_view[1]);
        }
    }
    {
        let logs = TuiLogWidget {
            state: &state.log_state,
        };
        frame.render_widget(logs, chunks[1]);
    }
    {
        let text = [
            Text::styled(
                state.menu_status.prompt(),
                Style::default().modifier(Modifier::BOLD),
            ),
            Text::styled("> ", Style::default().modifier(Modifier::BOLD)),
            Text::raw(&state.input),
            Text::styled("_", Style::default().modifier(Modifier::SLOW_BLINK)),
        ];
        let input_widget = Paragraph::new(text.iter())
            .alignment(Alignment::Left)
            .wrap(false);
        frame.render_widget(input_widget, chunks[2]);
    }
```

```
    {
        let text = [Text::raw(concat!(
            "1 = [register new item]   ",
            "2 = [rent item(s)]   ",
            "3 = [return item(s)]   ",
            "q = [quit]   ",
            "esc = [back]",
        ))];
        let instructions_widget = Paragraph::new(text.iter())
            .alignment(Alignment::Left)
            .wrap(false);
        frame.render_widget(instructions_widget, chunks[3]);
    }
}
```

Appendix C

Projectplan and Agreement

Projectplan

Equipment tracking

Roald Strangstadstuen Johannes Möckel
Ole Martin Ruud

30.01.2020

Contents

Glossary

Proof of Concept A proof of concept is a realization of a certain method or idea in order to demonstrate its feasibility, or a demonstration in principle with the aim of verifying that some concept or theory has practical potential. 1

Radio-frequency identification A technology that uses electromagnetic fields to automatically identify and track tags attached to objects. 1

The Norwegian Home Guard The Norwegian Home Guard has 40,000 soldiers all over the country and they serve as a quick mobilisation force for the Armed Forces. 1

Acronyms

HV The Norwegian Home Guard. 1–3, 7, *Glossary:* The Norwegian Home Guard

IS delivery form. 1

NTNU Norwegian University of Science and Technology. 1, 7

PoC Proof of Concept. 1–3, 8, *Glossary:* Proof of Concept

RFID Radio-frequency identification. 1, 2, 4, 6, *Glossary:* Radio-frequency identification

US submission form. 1

1 Goals and limitations

1.1 Background

The communications section of The Norwegian Home Guard (HV) must always have control over the location of different types of equipment. Such equipment can be expensive, have a long replacement time, contain cryptographic keys, and more. As per today this control is maintained via the use of paper receipts called submission form (US) and delivery form (IS). This process is labor intensive as errors are not always caught early. When finally caught, errors then tend to take significant time to correct.

1.2 Goals

The main goal of the project is to analyze several digital solutions to track equipment and develop a Proof of concept (PoC) of one selected solution. The project will conclude with a recommendation to the client whether it is both profitable and valuable to further pursue the digitalization of the process.

1.3 Limitations

During the project phase, the group will not receive any funds or assets from the project owner. This is due to the fact that there are no funds delegated to the project. Norwegian University of Science and Technology (NTNU), however, does have access to some computer equipment that we have permission to use for our PoC.

The system as a whole will run inside an encrypted network which is separated from the internet. This means that both the report and the PoC can assume that the system is run within a secure and encrypted environment. Further this has the consequence that the system has to run without access to the internet.

As the system will run within an already existing infrastructure, the project will not consider the process of setting up such an infrastructure. However the PoC will have to be run within a simple test infrastructure that will be configured and setup by the group.

2 Scope

2.1 Subject areas

- Radio-frequency identification (RFID)
  - RFID is at the core of our project, as we intend to use it for the actual tracking of the equipment. RFID has multiple different standards we will have to explore in order to determine the feasibility of using it for equipment tracking in HV.
- Databases

  - Since we're working on a tracking system, databases are going to play an important role in the created solution. Especially since HV is going to have specific requirements, such as syncing different databases after prolonged times without connection.
- Portable Scanners
  - Since we are going to use RFID-tags to keep track of equipment, we are going to require some sort of device to scan them with. Determining if there are suitable devices already on the market or potentially creating one of our own will also be something we will need to look into.
- Programming
  - Since we are aiming to deliver a PoC we are going to need to program a solution that communicates with the other parts and facilitates the tracking. Such a solution would need to implement the needed business logic to accomplish the tracking, but might also include other features depending how much time we end up having to implement extra things.
- Usability
  - Another broader aspect we will have to consider is usability, as delivering a product which is difficult or unintuitive to use brings little value. Especially in a potentially stressful environment such as in HV. Usability considerations could include anything from making the user interface easy to use, to suggesting additional markings on the RFID-tags in order to differentiate individual units of similar equipment in a on-field situation.

2.2 Limiting

- We will not be considering network configurations, or other aspects of the infrastructure, as this is already present. Further the physical locations where the equipment tracking will be used is also out of scope.
- The project will not consider physical constraints of the tags such as what glue should be used or if they need to be shielded from sunlight. Such potential problems will be raised in the report, but we leave further investigation of them up to other projects, as this is not our expertise.
- We will not deliver a fully functioning system, purely a PoC. Further development will be required, as we are only exploring the viability of such a system and aiming to provide a demonstration of how such a system could be realized. If a digitalized system should be implemented based on our findings, then such a system would be created from the ground up.

2.3 Task description

We are going to create a PoC for a digitalized, RFID based equipment tracking solution for HV that could replace the current all-paper solution they currently

use. We will analyze how feasible such a system would be and do so by both analyzing the underlying technology as well as existing software.

The final product would consist of our analysis of the feasibility, a list of further potential problems that need to be investigated that are outside of our scope, and a PoC in the form of a short demonstration of how such a system could work. Our product would then be used by HV to further decide if they believe it is worth to further pursue such a digitalized solution or not.

3 Project Organization

3.1 Responsibilities & Roles

- **Project Owner:** Leif Tanum
- **Contact Person:** Espen Torseth
- **Supervisor:** Sony George
- **Record Keeper:** Ole Martin Ruud
- **GitLab Manager:** Roald Strangstadstuen
- **Meeting facilitator:** Johannes Möckel

The group has given each member 1 major responsibility;

- As the records keeper, Ole Martin Ruud is tasked with making and keeping notes and meeting reports, or ensuring someone else is keeping notes in his absence.
- As the GitLab manager, Roald Strangstadstuen is tasked with maintaining the GitLab group and all the repositories under it that are used to organize files, as well as enforce proper usage of it to keep everything organized.
- As the meeting facilitator, Johannes Möckel is tasked with keeping everyone updated on when and where meetings are held, to coordinate absences, and to get into contact with members who are absent from meetings.

3.2 Rules & Routines

Rules

- Keep up good communication with the group.
  - If one cannot attend a meeting, inform the others as early as possible.
  - Ask if something is unclear or confusing.
  - Speak up if you disagree or take issue with something.
  - Etc.
- Be flexible, if things need to be changed then try to accommodate.
  - Meeting times are subject to change if required.
  - Other changes might be necessary as time goes on.
- Keep track of, and frequently check the GitLab used for project management.
- The group rules are subject to change as needed.

  - Changes to the group rules require unanimous approval.

Routines

- Pair programming is encouraged, but not mandatory.
- All non-trivial tasks shall be linked to a GitLab issue.
- Time shall be tracked locally via a personally selected tool.
- All tasks completed in merge-requests shall link to their issues.
- All non-trivial changes shall be approved by at least one other member.
- Make good use of the GitLab Kanban board, tags, milestones, issues, etc.
- If a member misses a meeting by more than 15 minutes they are called by phone.
  - Unless the rest of the group is in a meeting with supervisor or project owner.

4 Planning, follow up and reporting

4.1 Main Divisions of the Project

The project will have 4 main phases; the first of which will be mainly focused on getting properly started, get the project agreement signed and the project plan made. We will also try to create a good group environment during this period and establish good work ethics.

The second part will consist of us researching background technology such as RFID and various database solutions, as well as finding and evaluating existing solutions for the problem we are working on.

Once we have evaluated existing solutions and have familiarized ourselves with the background technology we wish to use, we will begin on creating a proof of concept for a better solution. This part will involve us creating a semi-functional system with various features to demonstrate the viability of the system. We cannot decide exactly what parts we will be working on during this period, beyond basic functionality. This is because a final solution would consist of multiple different parts, such as equipment tracking, maintenance overviews, and similar. We will decide which parts we wish to work on to best demonstrate viability at the start of this part of the project, once we have done all the necessary research for it.

The final part of our project period will focus mostly on the production of the project report and the corresponding presentation. We will be writing parts of the report as we go, and in this final part of the project a first draft and later finished product will be created based on the material we have created up to this point.

4.2 Choice of System Development Model

For this project we chose the Kanban system development model, and we plan on using the following Kanban columns: *backlog*, *doing*, *review* and *completed*. We are using a GitLab board as our Kanban board, where open and closed issues represent our *backlog* and *completed* columns respectively. The *doing* and *review* columns are represented by tags. This is all sorted into a graphical Kanban board by the GitLab board. We may add further columns later on as needed.

The reason we chose Kanban as our system development model was because it is a agile system, and works especially well with potentially dynamic projects such as this, given a lot of the later parts depend heavily on the results of our research. Kanban also has no clear "leader" which is nice for a project with only 3 people where we would like all members to have equal responsibility. The waterfall model in comparison is very rigid and in our opinion not as well suited for this kind of project. Especially since we do not even quite know what we will be working on in phase 3 yet.

Another consideration we made when we chose the system development model we were going to use was the fact that we are still rather inexperienced with following such a model. Other models, such as Scrum, have a lot more overhead than Kanban, which would only further complicate things for us, without giving much value. Kanban on the other head has minimal overhead, is easy to understand and gives a easy overview over the state of the project at all times. This in turn helps preserve momentum and motivation.

We also plan to heavily encourage pair programming, although we are not going to be strictly enforcing it since we're only 3 members, and working together in the same physical location for the targeted 30 hours per week would be hard to accommodate. We have taken this practice from the extreme programming system development model, because we believe it will help improve code quality and help with coordinating between us.

Another part of a foreign system development model we plan to use is the "fail fast, fail early and fail often" mindset from the Lean model. This is because we have a big project, and limited time and manpower to complete it. We will need to figure out what works and what doesn't quickly, instead of obsessing over details or work on things that might not end up being used down the line. This way of thinking also resonated with us and was used by some of the group members in the past, so we believe it to be a natural part of the process in this project as well.

4.3 Status Meetings & Decision Points

We have weekly meetings with both supervisor and project owner on mondays, unless otherwise moved. Additionally there are group meetings for 6 hours on both mondays and wednesdays, the meetings with the supervisor and project owner are included in the 6 hour meeting time on mondays.

As we are following a Kanban based workflow, we will not have explicit decision points, but rather decisions are made through continuous discussions. This allows us to quickly change focus and direction if needed, even in the middle of a working week.

5 Documentation, standards and sources

5.1 Code

Documentation

Code documentation should adhere to established standards within any given programming language. An example of this would be doc-strings from PEP 257 (Goodger and Rossum, 2001) when working in Python. Generally, comments should be used for any non-trivial parts and any other places where a comment would improve readability.

Standards

In addition to comments, we will also follow the other standards part of any given programming language. In Python, this would mean following the standards set forth in the various PEPs. We are however allowing ourselves some leeway to omit certain standards in cases where we agree that a standard gives no benefit to our project and rather slows us down or otherwise gives us reason not to follow it.

An example of such an omitted standard would be the controversial line-length part of PEP 8 (Rossum et al., 2001). This standard states that lines should be kept to a length of 80 characters. This is intended both to reduce complexity and to allow code to fit more easily into most terminal windows. Our group, however, has decided to rather confine us to a line-length of 100 characters, with some leeway should one cross it by a few characters from time to time.

5.2 General

Since our project is going to require a lot of research into different technologies such as RFID we will be spending significant amount of times looking as standards of said technologies. This will be done via official standards whenever possible. Any other sources we use will be rigorously checked for validity. Our sources will be updated continuously as we progress in our project.

Other forms of documentation, such as meeting reports will be made and kept in a separate repository in the GitLab we are using for the project work. This includes everything from notes, meeting reports, and other miscellaneous things we want to keep record of. We believe this will help us keep our work more organized, especially since we are planning on keeping records of all meetings we have with the project owner, project contact, and supervisor. We also have

another repository for research related documents which functions along the same lines as the above.

5.3 Risk assessment

Missing information from project owner (varying consequence, fairly likely)

It is fairly likely that communication with HV is going to be rather slow, and that receiving answers to questions and gathering information is going to take a long time. We might even run into scenarios where we will be unable to get certain kinds of information at all, though that is less likely.

In the event where we do not get the information we may need to limit our project accordingly to accommodate for the lack of information, which would in turn lower the quality of the project. Beyond that it would however not stop us from completing the project. The severity of this would largely depend on what information is withheld.

Unforeseen complexity (medium consequence, somewhat likely).

We may run into a scenario where we realize that we are unable to fully complete the goals we have set ourselves due to unforeseen complexity of other difficulties that are holding us back.

In the case we run into such a scenario we would have to consider changing our priorities and potentially drop some of our goals to complete the project.

GitLab is discontinued (major consequence, highly unlikely)

Since we are using GitLab for almost all aspects of our project it would be a big problem for us if NTNU decided to discontinue it. Thankfully this is very unlikely since NTNU is actively increasing it's usage of the GitLab and explicitly created groups in it for the Bachelor projects, indirectly encouraging the use of GitLab in them.

In the unlikely event the GitLab is discontinued we would have to spend a lot of time moving all our organization to another GitLab.

Losing a group member (major consequence, highly unlikely)

Since we are only three group members, it would be a major problem for the group if one of us became unavailable due to illness, or otherwise. It would drastically reduce the amount of work we can do and move their responsibilities to another group member.

In the event we do lose a group member we would have to work with our supervisor, project owner and figure out a good solution which lets the remaining group members complete the project.

6 Implementation and plan

6.1 Gantt Diagram

Since a lot of our work is going to depend on what our research uncovers, it's difficult to create a good Gantt diagram for this project. We will be in a perpetual state of researching and testing things with a fail-early and fail-often mindset. Because of this we will be simultaneously working on the development of the PoC as well as researching the next steps for it and the technology associated with them. This means that the PoC development and researching period spans most of our gantt diagram.

Furthermore, it's difficult to split these into smaller pieces since we can't be sure exactly which parts will be feasible to develop ahead of time, as this is something our research will show us as we go. Because of this, we cannot split the development or research phases into smaller periods, since we at present do not know what topics we will be working on once we have begun development in earnest.

What we do know is that we intend to make notes, drafts and outlines for the report continuously as we are working, while the research and results are still fresh in our minds. This means that work on the report (marked thesis in the diagram) is going to be ongoing for the entire duration of the project. Because of this we are aiming for an early first draft, giving us significant time to rework, rewrite and improve it for the final product. Aiming for an early first draft also means that we can extend our expected deadline for the first draft if we end up behind schedule.

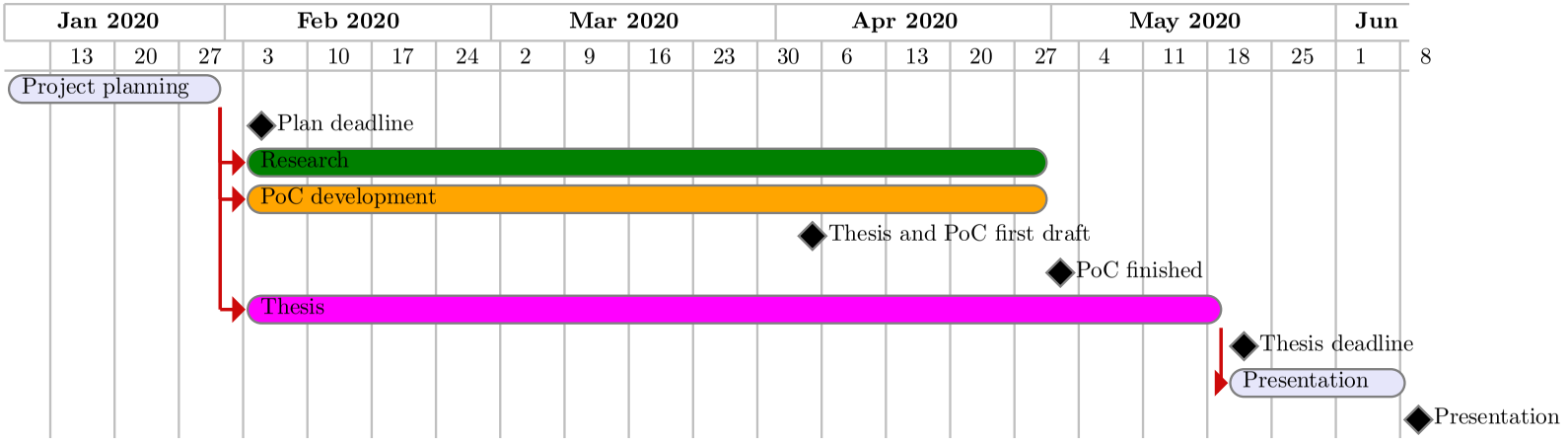


Figure 1: Gantt diagram

7 References

Goodger, D., Rossum, G. van, 2001. PEP257: Docstring conventions [WWW Document]. URL https://www.python.org/dev/peps/pep-0257/ (accessed 1.29.2020).

Rossum, G. van, Warsaw, B., Coghlan, N., 2001. PEP 8: Style guide for python code [WWW Document]. URL https://www.python.org/dev/peps/pep-0008/ (accessed 1.29.2020).

A Appendix

A.1 Project agreement

NTNU

Norges teknisk-naturvitenskapelige universitet

Vår dato Vår referanse

Prosjektavtale

mellom NTNU Fakultet for informasjonsteknologi og elektroteknikk (IE) på Gjøvik (utdanningsinstitusjon), og

Opplandske Heimevernsdistrikt 05 / G6-seksjonen

(oppdragsgiver), og

Roald Stragostadstuen, Johannes Möckel

og Ole Martin Ruud

(student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra 08.01.2020 til 20.05.2020.

Studentene skal i denne perioden følge en oppsatt fremdriftsplan der NTNU IE på Gjøvik yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra NTNU å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon, reiser og nødvendig overnatting på steder langt fra NTNU i Gjøvik. Studentene dekker utgifter for ferdigstillelse av prosjektmateriell.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. NTNU IE på Gjøvik står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av intern og ekstern sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Alle bestått bacheloroppgaver som ikke er klausulert og hvor forfatteren(e) har gitt sitt samtykke til publisering, kan gjøres tilgjengelig via NTNUs institusjonelle arkiv NTNU Open.

Tilgjengeliggjøring i det åpne arkivet forutsetter avtale om delvis overdragelse av opphavsrett, se «avtale om publisering» (jfr Lov om opphavsrett). Oppdragsgiver og veileder godtar slik offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og instituttleder/fagenhetsleder om de i løpet av prosjektet endrer syn på slik offentliggjøring.

Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode mv. som inngår som del av eller vedlegg til besvarelsen, kan vederlagsfritt benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av NTNU til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved NTNU og/eller studenter har interesser.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i NTNUs elektroniske eksamenssystem. I tillegg leveres ett eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med ett eksemplar til hver av partene. På vegne av NTNU, IE er det instituttleder/faggruppeleder som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og NTNU som regulerer nærmere forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene. Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale med oppdragsgiver, skjer dette uten NTNU som partner.

10. Når NTNU også opptrer som oppdragsgiver, trer NTNU inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene imellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

12. Deltakende personer ved prosjektgjennomføringen:

NTNUs veileder (navn): SONY GEORGE

Oppdragsgivers kontaktperson (navn): Espen Torseth

Student(er) (signatur): _____ dato 22.01.2020

_____ dato - 11 -

_____ dato - 11 -

_____ dato _____

Oppdragsgiver (signatur): _____ dato 17/1-20

Signert avtale leveres digitalt i Blackboard, rom for bacheloroppgaven.
Godkjennes digitalt av instituttleder/faggruppeleder.

Om papirversjon med signatur er ønskelig, må papirversjon leveres til instituttet i tillegg.
Plass for evt sign:

Instituttleder/faggruppeleder (signatur): _____ dato _____

Appendix D

Meeting Logs

Forventninger, regler og roller

Author	Ole Martin Ruud
Date	2020-01-08
Participants	
	Ole Martin Ruud
	Johannes Möckel
	Roald Strangstadstuen

Ønsket karakter

Vi er enige betydelig innsats og sikter på en B

Viktige tanker til skriving

- Rapporten skal skrives for å kunne forstås av en medelev
- Refleksjon (diskusjon rundt valg) er viktig for å vise forståelse
- Tanta til Roald kan lese gjennom oppgaven

Regler

Vi forventer at man

- sier ifra dersom noe ikke passer, man blir forsinket, man blir syk
- er fleksibel angående møtetid og andre deler
- ikke er redd for å være uenig
- kommuniserer godt med hverandre

Flere kommer senere ved valg av SU.

Roller og ansvar

- Ole Martin
  - Referent
- Roald
  - Issuemaster
- Johannes
  - Fasilitator

Neste møte 2019-01-09

- Sette opp miljøer (git repo, kalender, time tracking)
- Starte på mal for prosjektplan og rapport
- Kontakte Sony og Espen

Kartlegging med oppdragsgiver

Author	Ole Martin Ruud
Date	2020-01-09
Participants	
	Ole Martin Ruud
	Johannes Möckel
	Roald Strangstadstuen
	Espen Torseth

Agenda

- Hardware
  - Kjøpe?
  - Budsjett?
  - Hvem?
  - Hva?
  - Hvilke midler?
  - Begrensninger?
- Når kan vi se prosessen å sjekke ting inn og ut av lager?
- Begrensninger iforhold til oppbevaring av kode/rapport?
- QR vs. RFID vs. barcode - noen begrensninger fra arbeidsgiver?
- Fast møtepunkt? (f.eks. annenhver uke, mandag)
- Krav til utviklingsmodell?
- Forventninger til oss

Notater

- Mål: Analyse av fordeler/ulemper mm + POC (kanskje prototype?)
- Analyse av problemet er viktig; er dette rette veien å gå med tanke på pris, nyttighet, effektivitet?
- Tur til Terningmoen for å se på lager og nåværende løsning
- Se gjennom selve prossessen (IS og US)
- Mulig noe utstyr som kan brukes til POC
- Undersøke MicroPython og CircuitPython
- ESP32 - Bra batterilevetid, deep-sleep mm.
- NSN og/eller SAP-ID unikt identifiserer model/type utstyr
- Serienummer er unikt per NSN og SAP-ID
- Mulighet til å rydde opp i ettertid (historikk ved ID-basert utstyr) av store bevegelser
- Rotering av utstyr for å fordele slitasje
- Vedlikehold av utstyr, periodisk eller oppsatt tidspunkt
- Regelmessige notifikasjoner e.l.
- Utstyr
  - Telefoner identifiseres av tlf. nr.
  - Ugraderte USBer trackes i medieregisteret
  - SD-kort skal også i teorien trackes
- Fast møtepunkt mandager kl. 1230

- Hvordan dette henger sammen?
- Hvordan prosessen er?
- Hva kan forbedres?
- Hvordan komme langt nok ut (til endebrukeren)?

Status meeting with project contact

Author	Roald Strangstadstuen
Date	2020-01-13
Participants	
	Ole Martin Ruud
	Johannes Möckel
	Roald Strangstadstuen
	Espen Torseth

Saker og notater

- Skolens utstyr?
  - Ikke enda.
- Openstack?
  - Eigil / Lars Erik
- Infrastruktur per dags dato?
  - Det finnes.
    - Nyere windows kan vi forvente.
- Besøk
  - Får svar snart.
  - Gjøvik ting oppsett.
    - Godt for å se litt på bilde.
  - Grebe (innsatsstyrke)
    - Se på hvordan vi for eklest mulig hentet info vi trenger.
- Huske at vi driver med POC
  - Finne ut hvordan vi skal spare tid. (ikke nødvendingvis penger)
    - ESO-(oppsett)

1. Fra du kommer inn til du er klar.
   1. Tid fra folk kommer inn, til enheten er klar.
   2. Vi burde jakte hvor lang tid "ting" tar idag.
      1. Både i for og etter-kant.

- Sjemaer
  - Kommer snart.
    - Ekempel utfylt.
- Gitlab-en
  - Sendt epost.

Status meeting with project counselor

Author	Ole Martin Ruud
Date	2020-01-20
Participants	
	Ole Martin Ruud
	Johannes Möckel
	Roald Strangstadstuen
	Sony George

Agenda

- Discuss and cement the project background/goals/limitations
- Discuss how to write a generic plan (we have many parts which can be explored if given the time)
  - How to make a detailed plan when we are unsure about what we are going to do?
- Find out where to go next (we need to coordinate our efforts regarding the plan)
  - Set a concrete goal (a point on each part of the plan)

Notes

- Very generic task because we don't know what were doing when
- A plan is good no matter what
- Define some tasks despite leaving it open
- The plan should be easy to read for outsiders
- Limit which tasks we want to do and make an order
- Get clear requirements from project owner
- Risks involved with having a project owner that doesn't participate

Main project period

- Evaluate some existing solutions (1 week)?
  - End with a report of pros and cons

Status meeting with project contact

Author	Johannes Möckel
Date	2020-01-27
Participants	
	Johannes Möckel
	Roald Strangstadstuen
	Espen Torseth

Agenda

- Discussing access to the cyber-range room.
- Asking what translations we should use for HV and such in English.

Notes

- We will get access to cyber-range rooms soon™.
- Use the translations on the wikipedia page.
- Perhaps discuss if we should have a visual indication in addition to RFID
  - To prevent mixing of equipment.
  - Colors could work. Color blindness is not a problem.
- We can expect users to be comfortable with PCs. No hunt-n-peck people etc.
- Design should support functionality, it absolutely doesn't need to look "nice".
- Take a look at how visma and fike (accounting solutions) are naming things.
  - Names should be easily understandable. Make things intuitive.
- People are tracked via name, rank and department.
  - GDPR is of no concern, military gets exceptions to many things.
- Figure out what the minimum requirements are, hardware etc, and the associated cost.

Status meeting with project counselor

Author	Ole Martin Ruud
Date	2020-01-29
Participants	
	Ole Martin Ruud
	Johannes Möckel
	Roald Strangstadstuen
	Sony George

Notes

- Using numbers or numbers using letters is complicated
- Gantt looks okay but perhaps divide PoC development
- The report needs to be understandable for a stranger
- Discuss what we mean by PoC and demo - See glossary
- Add some feedback from the customer
  - Do some testing e.g. in a specific case

Status meeting with project contact

Author	Ole Martin Ruud
Date	2020-02-03
Participants	
	Ole Martin Ruud
	Roald Strangstadstuen
	Espen Torseth

Agenda

- Diskutere PoC dybde
- Datoer for øvelse

Notater

Datoer for øvelse

[*REDACTED*]

PoC

- LFR -> Trekloss for å vise størrelse (veldig grunnleggende)
- Redusere variabler (e.g. kostnader, praktisk gjennomførbart)
- Sette rammen for å kunne lage et greit eksempel
- Koding er fint i PoC'en

Status meeting with project counselor

Author	Ole Martin Ruud
Date	2020-02-05
Participants	
	Ole Martin Ruud
	Johannes Möckel
	Roald Strangstadstuen
	Sony George

Agenda

- Plan

Notes

- Some dates next week
- Start researching and read up on ISOs
- Now that we know what we want we should try to clarify the plan a bit
- Sony knows some people working with RFID that we could ask if we need something in particular (specific requirements) — redacted — — redacted —

Visit at HV (Terningmoen)

Author	Ole Martin Ruud
Date	2020-02-25
Participants	
	Ole Martin Ruud
	Johannes Möckel
	Roald Strangstadstuen

Notater

- Lett info om HV struktur
- SAAB ved Østen Almqvist har allerede en liknende løsning (må kontaktes)
- Personvernshåndtering; kun benytte ansattnummer og stilling/avdelig (kan være at også navn er greit, men gå utifra at det ikke er det)
- Hva slags enhetene må bestemmes (laptop, håndholdt enhet)
- Støtte for både sporing av enkeltenheter og antall enheter (HV må selv velge hva som skal brukes på hvilket utstyr)
- Sette forslag til grenser for historikk
- Vurdere strekkoder over RFID
- Ikke nødvendigvis krav til IS/US, men kan være greit
- Utfordringer:
  - Håndtere sett av utstyr med feil og mangler. Fjerne utstyr fra et innlevert sett med mangler.
  - GUI
  - Administrative det rundt systemet (tilgang, typer, nye klienter mm.)

Status meeting with project counselor

Author	Ole Martin Ruud
Date	2020-03-04
Participants	
	Ole Martin Ruud
	Roald Strangstadstuen
	Sony George

Agenda

- Visit at Terningmoen last week
  - Useful to see the location and ask questions
  - See how they operate
  - Clarified what is important to them
  - Learned about an existing system from SAAB
- Visit at Rena on friday
  - After talking to SAAB we will go see their system
  - A lot of the same specifications as us
  - They have experience with tagging equipment
  - What should we do if the system they have their perfectly fits HV's usecase?
- Some more progress has been done on writing the thesis itself but going slow

Notes

Skype was not cooperative so meeting was not completed.

Visit at SAAB (Rena)

Author	Roald Strangstadstuen
Date	2020-03-06
Participants	
	Ole Martin Ruud
	Roald Strangstadstuen

Goals

- Find out how SAAB tracks their equipment they lend out to the military.
- Question them about their experiences
- Look at how they deal with non-normal operation (service, broken equipment...)

Points we got from them

- They were most happy with barcodes as they are fairly easy to figure out. They can be printed on smaller tags as the reads are mostly able to read quite small barcodes.
- Their bar codes consist of a article and serial number (artikkel og serienummer). Where article denotes type of equipment and serial is a unique identifier for the item for that article number.
- Their barcodes use Code 128.
- In addition to creating bar codes for equipment, they also have special barcodes around the client to enable the user to scan tags to operate the system. This means that the user can control the system with only a barcode reader.
- RFID with UHF was looking promising. However they had problems with reading more than what they wanted. They also needed to inspect all the equipment regardless of the scanning when it was returned to start the repair process if anything was broken.
- QR-code was to big to be put on a lot of their equipment.
- Their system was based around exercises, however now they would like if a transaction was marked as an exercise.
- Their database with all the movements of roughly 350k items over 18+ years was about 1GB in size.
- They wanted more integration with third-party repair providers.

Status meeting with project counselor

Author	Ole Martin Ruud
Date	2020-03-18
Participants	
	Ole Martin Ruud
	Johannes Möckel
	Roald Strangstadstuen
	Sony George

Agenda

- Visit HV at Terningmoen
  - Useful to see the location and ask questions
  - See how they operate
  - Clarified what is important to them
  - Learned about an existing system from SAAB
- Visit SAAB at Rena
  - A lot of the same specifications as us
  - They have experience with tagging equipment
  - Proprietary system developed inhouse
  - Barcodes were preferred
    - RFID gave no apperant benefits
    - QR codes too big in many sitations
- Some more progress has been done on writing the thesis itself but going slow
- Change from development to research centered thesis
- Coding the prototype has been "delayed"

Notes

- Give Sony info about the previous events
- Next steps:
  - Sony will research how to make the interviews concrete
  - Sony will also read about meetings and make some comments
  - Continue writing the thesis content

Joint meeting with project counselor and project contact

Author	Ole Martin Ruud
Date	2020-03-25
Participants	
	Ole Martin Ruud
	Johannes Möckel
	Roald Strangstadstuen
	Espen Torseth
	Sony George

Agenda

- Update from Sony to Espen
- Current status

Notes

- Sony has updated and been updated from Espen that we are on track for solving the task
- Currently working on the report
  - Lack of need of meetings due to the fact that we know what we need to write, and now we just need to write it down on paper.
- Cross platform is nice-to-have and might be heavily beneficial as there are probably different hardware/software at different locations.
  - Mainly planed to run on a Raspberry PI, but should also support windows if location already have a server/computer in the storage facilities.

Status meeting

Author	Ole Martin Ruud
Date	2020-04-15
Participants	
	Ole Martin Ruud
	Johannes Möckel
	Roald Strangstadstuen

Notes

- Roald struggels to add more sections (feels like the topics are good but perhaps lacking some content)
- Should discuss economy

Planning the weeks ahead

- Write about economy for monday next week (Ole Martin).
- Johannes will merge his part.
- Send task to Roalds aunt 1. may.

Status meeting with project counselor

Author	Ole Martin Ruud
Date	2020-04-23
Participants	
	Ole Martin Ruud
	Johannes Möckel
	Roald Strangstadstuen
	Sony George

Agenda

- Read through draft thesis in advance and discuss/comment on it
- Handle issue #13 from Gitlab
- Plan forward

Notes

- Define the audience of the thesis.
  - We could add a disclaimer to the introduction about required previous knowledge.
  - **Audience should have basic technological competance, but no knowledge about distribution, databases and details**
- Chapter 1
  - Extend this part and improve the build up of the motivation.
  - Give a hint to the reader about the requirements of the system.
  - Perhaps make a link from research questions directly to results/discussion chapters which discuss that question.
  - Add a small chapter about the security which explains why we *mostly* don't care about it.
  - Mention dissemination/contribution in the introduction.
    - Write a small paragraph explaining how the knowledge will be transferred to the military.
- Chapter 2
  - Imagine that somebody reading the thesis is someone without knowledge.
  - Usage of ITIL, WMS should be explained a bit better before common usage.
  - Move interview part from SAAB to results chapter, because it's a part of the result of applying our method (even though it is also in a sense background knowledge).
  - Expand a bit more about the SAAB system (it is a valuable context).
  - Explain tagging/scanning in chapter 2 and reference it in a later chapter.
- Chapter 3
  - Why are we choosing the following methodologies? Make a small introduction.
- Chapter 4
  - Add analysis of existing systems in regards to the requirement.
  - Link sub chapters to the research questions.

Notes from Johannes

good structure

1 is good intro

good background, but maybe we should consider presenting it to someone who is not really into the topic. f. eks. we explain about different existing stuff like wms chapter 2 last paragrapth is interview, but that just suddenly apears there. we never talked about it before it's presented later in method maybe move to result? we have no context here have some more about Saab system, since it's never grought up before

chapter 4 we discuss other rfid stuff, this is also mentioend in background.

requirements should be earlier motivation need to be presented better, out of which research questions come

In 3.1 we're directly going to qualitative interviews why we are using them but before presenting 3.1 we should state WHY we are uisng this method more overarching exlaination as to why we chose thiese methods in particular, opposed to others

4.1 what we're missing is the link with the research questions 1.2.1 should link with the different section here

for the reader it should be easy to see where the questions are adressed. blob diagramm

chapter 1 dissemination

Status meeting with project owner

Author	Ole Martin Ruud
Date	2020-05-06
Participants	
	Ole Martin Ruud
	Johannes Möckel
	Roald Strangstadstuen
	Espen Torseth

Agenda

- How to deal with the interview part as we did not really conduct a proper interview

Notes

- Many different forms of interviews
  - Write a summary of the interview and ensure that they agree with the contents
- The main product of the thesis is the understanding of the target area
- Consider making a visual PoC which shows the use of the system to add to the thesis as a digital addition
- Become comfortable with the presentation tool (e.g. Zoom or Teams)

Feedback from project counselor

Author	Ole Martin Ruud
Date	2020-05-12
Participants	
	Ole Martin Ruud
	Johannes Möckel
	Roald Strangstadstuen
	Sony George

Notes

- The thesis is on the shorter side, we could add more
- We could leave the orange parts describing the sections if we modify them a bit
- Introduction - Chapter 1
  - Extend the introduction
  - Give some idea about the scale of equipment (size, difficulty)
  - Explain replacement time better
  - Make a PoC based on what? List requirements in the beginning to make it clear
  - Explain the contribution of this thesis
- Background - Chapter 2
  - Distributed systems is a bit out of nowhere, needs introduction or reason
  - There is no sign of a link to the glossary from keywords
  - Are there any specific solutions which are relevant for each category of systems? Examples which show that we spent time on analyzing systems
  - Scattered requirements leads to a lack of background for making statements
  - Onsite visit with Saab
  - Move Saab system mention to results
  - Make connections to requirements
- Method - Chapter 3
  - Change qualitative interview to onsite visit
  - Change order of definition of methods to the order that they were executed
  - Mention how we prepared/planned for the visits/interviews
    - Gain experience about the location and processes
    - Gain experience about what scanning solutions they use
  - Interviews are probably the biggest contribution to the thesis
  - Break down the energy spent in the different parts of the thesis/project
- Results and discussion - Chapter 4
  - Write a small introduction to the chapter
  - Don't repeat tagging sections
  - Ensure that tagging and scanning is not a repeat of background but rather discussing and arguing
  - Clarify that Saab technologies part is the result of the interview
  - Make references to research questions and requirements
- Basic requirements in chapter 1, but then refactor and redefine requirements after interviews/visits
  - Same for RFID (why did it change)

Johannes notes

1.1 short, a bit more open (easier to follow for those without understanding.)

Requirements need to got to the beginning (above hypothesis)

limitations (security etc.)

we should say something about dstributerd systems.

Saab needs to be introduced before

make reason in WMS

for existing examples, give examples

interview -> onsite visit

move Saab from existing solutions out

instead of repeating tagging, move the details to results from background

onsite visits section

clarify 4.2.1

Feedback from project counselor

Author	Ole Martin Ruud
Date	2020-05-16
Participants	
	Ole Martin Ruud
	Johannes Möckel
	Roald Strangstadstuen
	Sony George

Agenda

- Ask whether we should move discussion of existing solutions in background to results chapter.
- Definition of hypothesis based on requirements?
- Requirements analysis
- Rough comments from Sony about current version

Notes

- Move discussion part of existing solutions to results
- Use requirements in hypothesis
- Ensure glossary entries are displayed properly
- Add images for bar code, QR codes and RFID (example of Saab barcode)
- Some lacking information
- Add small introductions to all major chapters explaining what we will do in the following chapter
- 4.4. Mention why we didn't consider centralized database storages (PostgreSQL)
- 3 Methods
  - Planning qualitative interview explanation
- Flow graph of visualization of what we did
- We see/experience should be explained more

Last meeting with project owner

Author	Ole Martin Ruud
Date	2020-05-18
Participants	
	Ole Martin Ruud
	Johannes Möckel
	Roald Strangstadstuen
	Espen Torseth

Agenda

- HV vs the HV vs the Home Guard
- Dissimination
- Any particular feedback points?
- Our presentation is the 08.06.2020 at 11:20

Notes

- Remember to add appendicies (project plan etc.)
- Change HV to the Home Guard
- Dissimination looks good
- Ensure we answer the research questions in the results and discussion:
  - Some questions might not get enough of a basis to answer, thats permittable
  - Some questions might not be answerable (economy?)
  - Some questions might be not refined enough
- Ensure we use gls for ALL acronyms/glossary entries
- Hypothesis looks good and reference to previous sections
- Split current limitations into limitations and assumptions
- Add references to requirement analysis
- Check usage of logo with Leif

Last meeting with project counselor

Author	Ole Martin Ruud
Date	2020-05-20
Participants	
	Ole Martin Ruud
	Johannes Möckel
	Roald Strangstadstuen
	Sony George

Agenda

- How should reference norwegian words (artikkel nummer + serie nummer)
- Should we explain how we generated barcodes, qr-codes, rfid?
- Should we mention Saab explicitly in method or just “loosly”?
- Should we add the source code of the poc? (ca. 600 lines)

Notes

- Reference norwegian words through parantesis and italic
- Add a footnote detailing that we made images ourselves.
- Only mention Saab loosly (as currently done)
- Add source to appendix

J. Möckel, O. M. Ø. Ruud, R. Strangstadstuen

Distributed Equipment Tracking in Unreliable Network Environments

NTNU
Norwegian University of
Science and Technology