Akuttjus

Group 1

TDT4290

Authors:
Thor Martin Abrahamsen
Are Gotteberg Haartveit
Eirik Tvedt Jensen
Petter Bakkan Johansen
Morten Kartevoll
Sarah Serussi

November 12, 2015

NTNU
Norwegian University of
Science and Technology

Abstract

Doctors and other health personnel face a variety of situations with high risk almost on a daily basis. There is high risk in surgery, in the handling of a spreading infection, and even when treating patients with medicine. In order to minimize these risks hospitals have developed large amounts of procedures for the sole purpose of mitigating risks when handling patients. The issue, however, arises when doctors are forced to use cumbersome systems to find information about the relevant procedures for a given situation. Especially for situations which are quite similar to each other, or for situations that rarely occur, the need to look up information often arises, which leads to doctors spending too many of their work hours looking for the information or even in some cases documenting and practicing based on misinformation.

The customer has asked the team to develop a knowledge support tool to be used at the St. Olavs Hospital in Trondheim. The tool is intended to assist the medical personnel at the Emergency Room with the correct use of laws for a specific set of situations. The tool needs to be simple, quick to use, and intuitive, while retaining all the necessary information to cover the situations a doctor faces when dealing with difficult patients.

With this in mind, the group has created a tool that helps doctors in their decision-making when faced with patients that are either suicidal, violent, inflicting harm on themselves, or non-consenting to treatment the doctor deems necessary. In order to realize this tool, the group have created an application that will be available for use on smart phones, tablets and on PC.

The team's most important goals are to improve decisions in the aforementioned situations and to help ensure that doctors exercise their legal rights correctly while making sure that they do not overstep their bounds, thus protecting patients' rights as well. These goals need to be reached in a high usability alternative to the current more cumbersome systems.

Acknowledgements

On behalf of the team we wish to thank our advisors, Pekka Abrahamsson and Soudabeh Khodambashi, for their guidance, and support throughout this entire project. We also wish to thank our contact at St. Olavs Hospital, chief physician Florentin Moser, for being available and answering all our questions, as well as giving insight into the hospital and the ER facility. He also helped us understand the patient situations relating to our project.

We would like to express our gratitude towards the following people: Jostein Dale, a chief physician at St. Olavs Hospital, for giving us valuable feedback about our solution, the course staff of TDT4290 for granting us the possibility to receive feedback on the midterm report, every person who participated in the prototype test and gave us invaluable observations and Merete Blokkum for providing feedback on how information relating to the laws in the application was phrased and displayed. Lastly we would like to extend our thanks to Norwegian University of Science and Technology (NTNU) for giving us an office where we could work and for providing general supplies.

Outline

This chapter contains the goal of the report as well as an outline of the chapters, with a short explanation of their contents.

Goal of the report

The goal is to show the process from the initial project's problem description, the planning that was done before and during the sprints, to the finished product and evaluation of the project. The report will document the different phases, from the pre-study phase to the last sprint, sprint 4.

What this report covers

This section gives an overview of the chapters that are covered in this report.

Chapter 1: Introduction
Contains an introduction to the problem that is to be solved, the project background and the project goal.

Chapter 2: Preliminary study
Describes the research that was done in the preliminary study phase, as well as the current, and desired situation.

Chapter 3: Planning
Documents the planning that was done during the project, from the overall estimation of working hours and the actual hours spent, to the estimation and actual hours spent in each sprint.

Chapter 4: Requirements
Presents the requirements for the product that was to be implemented, with explanatory use cases.

Chapter 5: Methodology
Introduces the chosen methodology for this project, as well as other methodologies that could have been used.

Chapter 6: Risk Analysis
Presents the risk associated with the project and the product that was to be created, as well as an analysis of the strengths, weaknesses, opportunities and threats related to the project.

Chapter 7: Technologies
Contains the different tools and technologies that were used in the project.

Chapter 8: System design
Presents the architecture of the product's software with explanatory views.

Chapter 9: Sprint 1
Describes the planning and process within sprint 1, with a backlog that contains a list of the tasks that were done in this sprint.

Chapter 10: Sprint 2
Describes the planning and process within sprint 2, with a backlog that contains a list of the tasks that were done in this sprint.

Chapter 11: Sprint 3
Describes the planning and process within sprint 3, with a backlog that contains a list of the tasks that were done in this sprint.

Chapter 12: Sprint 4
Describes the planning and process within the last sprint, with a backlog that contains a list of the tasks that were done in this sprint.

Chapter 13: Testing
Presents the tests that were done in the project, including user tests and prototype testing of the product that was implemented.

Chapter 14: Evaluation of the product
Discusses the evaluation of the finished product, as well as a presentation of the requirements that were fulfilled, and a section about further work that could be done on the project.

Chapter 15: Conclusion
Presents the lessons learned throughout the project as well as concluding remarks about the project.

Chapter 16: Reflection
Reflects about the process of the project, as well as the challenges that were faced in the course of the project.

Contents

List of Tables

List of Figures

1 Introduction

As a part of the Master's program at the Department of Computer and Information Science at NTNU, the course TDT4290 Customer Driven Project (CDP), is attended by fourth year students [30]. The students that attend this course are divided into groups where each group is assigned a customer who decides upon a project to provide to the group. The purpose of the collaboration between students and customers is to give the students practical experience of how it really is to work on a bigger project that will result in a product for a real customer.

The group was given the task of developing a knowledge support tool that will be used in the Emergency Room (ER) at the St. Olavs Hospital in Trondheim. The tool is meant to help doctors that are working there to better their documentation in cases where they have treated patients that are either suicidal, violent, harming themselves, or non-consenting. It might also be used as a part of the informal education and maintenance of doctors' routines of handling difficult patients in emergency situations.

1.1 Project Description

The project task is to create a knowledge support tool in the form of an application (app) for the ER at the St. Olavs Hospital in Trondheim. The product is intended to assist the medical personnel at the ER with the correct use of laws for a specific set of situations. These situations involve cases where the patient is either suicidal, violent, harming themselves, or non-consenting. In some of these cases it might be difficult to observe a clear separation between the different categories, especially in cases where the mental state of the patient influences the situation. This often leads to doctors practicing and documenting the wrong laws, which is what the app is meant to mitigate in order to better preserve both the patients' and the doctors' rights.

From the initialization of the project the customer had a clear idea of the content and seemed confident that there would barely be any changes to the content, at least for the next five to ten years. This is mainly due to the fact that laws regarding health care have been added or changed only once or twice in the last five years, and that a new law often takes several years to be deducted. However, after having found additional content and extending the use of the application the customer, in cooperation with the development team, settled on developing functionality that would let administrators add, alter, or delete content of the application. The name that was decided for this product was "Akuttjus", which translates to "Emergency law" in English.

1.2 Project Sponsor

Figure 1.1: Project sponsor St. Olavs Hospital

The customer is the Department of Emergency at the Clinic of Emergency Medicine and Prehospital Care at St. Olavs Hospital in Trondheim. The team's contact is chief physician Dr. Florentin Moser.

As the local hospital of Sør-Trøndelag, St. Olav covers 302,000 inhabitants. As well as being the local hospital, St. Olav also has regional and national tasks for counties like Møre og Romsdal, Nord-Trøndelag, and Sør-Trøndelag.

There are several institutions located in Sør-Trøndelag that specialize in somatic and mental health care. The main portion of these locations are situated in Trondheim: Brøset, Lian, Østmarka and Øya. In addition there are three District Psychiatric Centres, where two of them are located in Trondheim and the last one in Orkdal. The clinical activity also consists of a few hospitals, pinpointed in Orkdal and Røros, and the Rehabilitation Centre in Hysnes.

Besides being a hospital for the general habitations of Sør-Trøndelag, St.Olavs Hospital is integrated with NTNU in Trondheim. Whereas patient treatment, research and education are built-in functions.

1.3 Involved Parties

As mentioned earlier, this project is a result of the learning material from the mandatory course CDP. Even though this is not a traditional project with companies on each side of a contract, there are still a few stakeholders in this project; the students, the supervisors and the customers.

1.3.a The project group

The project group consists of the students that are in the same group in the CDP course. These students are all fourth year students at NTNU's Computer Science Master's programme.

Experience of the group

The group consists of fourth year students from the Computer Science program at NTNU. Although there have been many types of teamwork during the first three years at NTNU, there have not been many projects that have been as extensive as this one. Thus, the team has had little to no experience working on a big project.

Team member	Role	Experience
Sarah Serussi	Project Leader, Customer Contact	Design, Java
Thor Martin Abrahamsen	Test Leader	Web development, Django
Morten Kartevoll	System Designer, Scrum Master	Software development, Photoshop
Eirik Tvedt Jensen	System Architect	Git, Python
Petter Bakkan Johansen	Quality Assurance (QA) Responsible	Game development, User Experience (UX)
Are Gotteberg Haartveit	System Architect	Software architecture, HTML5

Table 1.1: Team members and their roles

1.3.b The customers

The customers of this project are the ones that created the task that has been assigned and those that hopefully will use the end product that is produced during the course of this fall semester.

- Department of Emergency at Clinic of Emergency Medicine and Prehospital Care at St. Olavs Hospital in Trondheim
- Chief physician Florentin Moser.

1.3.c Product owner

The product owner is the one initiating the project and, on behalf of his organization, outlays requirements that would lead to his vision of the end product. He is the main stakeholder and responsible for all executive decisions. In this case the product owner is the customer contact, Florentin Moser.

1.3.d The supervisors

The supervisors of the course CDP are the ones responsible for having an overview of the students' work and progress of the project given. They guide the students through weekly meetings with the student groupps.

- Pekka Abrahamsson
- Soudabeh Khodambashi

1.4 Project Background

At the ER the doctors encounter many different types of patients. Some of these patients may oppose treatment, even when it is crucial that they receive immediate attention. There exist laws for when the hospital personnel can, when they shall, and when they can not give medical treatment for such a patient. After a decision like this is made it needs to be documented in the patient journal, referring to the given law.

According to the customer, Florentin Moser, the hospital personnel generally have problems looking up laws that are applicable to a situation using the tools they have available. As a consequence the documentation is sometimes done incorrectly.

The systems which are used today are way too complicated and cumbersome to use, and additionally contain an immense amount of information. According to the customer, doctors and medical personnel can use up to 15 minutes, or even more, to find a simple form.

1.5 Project Goal

By creating the application "Akuttjus", the team and the customer especially, have high hopes that this tool will help medical personnel that are working in the ER, by being able to make decisions more easily, and that the doctors will be able to document correctly after an incident as well as having a useful tool for learning.

A specific goal of this project is to create a tool to educate and help hospital personnel documenting correctly when dealing with patients who behave problematically. This is done by presenting the applicable laws for a given situation. As well as educating and helping the medical personnel with documenting correctly, the application's goal is also to drastically reduce the amount of time used to find laws and forms for a specific situation.

Another goal for the team is to acquire real life experience with an authentic customer. In addition the objective was to function as a group, and document the life cycle of the project as well as learning more about the technical aspects of making software.

1.6 General Terms

This product is intended to be used by the staff at the ER in St.Olavs hospital. Nevertheless, the laws referenced in the product applies for all hospitals in Norway. By excluding location specific procedures, the product can be used nationwide.

This product specifically covers patients who are suicidal, violent, self-harming or non-consenting. After searching for similar applications in different application stores and visiting a variety of sites on the Internet, there was no application or site that directly addressed the customer's requirements in this project. Nevertheless we found similarities in the functionality we are planning to use. Our research indicates that there does not exist any direct substitute for our product.

1.7 Scope

The project started August 25th. and ends with a demonstration of the final product November 18th. The deadline for submitting the report is the 12th of November, and so the estimated amount of work is calculated for the period, 25th of August to the 12th of November. The team consists of six members and each team member is expected to work approximately 25 hours a week. This yields 1710 person-hours during the 11,4 weeks of the project.

1.8 Summary

In this chapter the following parties have been introduced: the customer St. Olavs Hospital, the team members, and the supervisors. Hospital personnel find it problematic to document the correct law. Therefore, one of the goals of this project is to reduce the amount of time it takes for a person to locate the law for a given situation. Although the group of patients this tool will be used on is relatively small, the tool is applicable for all hospitals nationwide.

2 Preliminary study

The preliminary chapter is about understanding the task given by the customer. In order to better comprehend the assignment it was divided into smaller parts, and each of these are analyzed in order to come up with a desired solution. By looking at today's situation and comparing it to the wanted solution, several key points to the development process were identified. These key points were then used to analyze frameworks and technologies that could provide a possible solution. Several possible solutions are discussed in order to find develop the evaluation criteria.

2.1 Problem description

The problem that needs to be solved is the amount of wrong documentation by doctors working at the ER, and to help medical personnel to make decisions faster and to take the decisions based upon their knowledge of which laws apply in the different situations. This problem will be solved by making a knowledge support tool that will be used in the ER at the St. Olavs Hospital in Trondheim.

The "Akuttjus" tool will help the doctors to document the procedures and laws used after making a decision about a certain patient. There are many different types of patients coming to the ER, and this tool will help the doctors to categorize them into patient groups by the correct measures to be taken after an incident. The tool will contain the different laws corresponding to the various types of patients. It will also be used as a encyclopedia tool for the doctors where they can find which laws that are linked to which patient group.

The tool will mainly be used after an incident has occurred, as the severity of most of the situations that arise in the ER are high, but it can also be used during a situation if the necessity of immediate attention is low.

2.2 The situation today

In the St. Olavs Hospital ER Division there are a number of different software programs that are used on a daily basis. Among others, there are programs for adding and writing patient journal files, software for getting relevant reports, and software for retrieving forms that are necessary for documenting a situation.

One system that is used today is called Electronic Quality System (EQS) [2]. This program helps to manage, secure and develop quality in different types of work processes. It contains documents about guidelines, procedures and planning which can help the doctors to document their work. The EQS has an enormous database that includes reports, guidelines and procedures. This makes the system quite extensive. Without any intuitive navigation or search function, the system can quickly become too comprehensive to use. By using the EQS, a doctor can spend up to 15 minutes just searching for the correct form to fill out. In the ER in particular there is a need for easier and quicker access to information.

2.3 The desired situation

The most important aspect of this project is to implement a tool that will help doctors acquire information fast and easy. The goal is to reduce the number of incorrectly documented cases, by giving hospital personnel easy access to guidance. A desired situation is one where doctors, nurses, and medical students can use the tool as practice to more frequently remember the current law for a given situation, and that they have a tool to rely upon in situations where they are unsure what to do.

2.4 The possible solutions

There are a few possible solutions to solving the problem that the ER at St. Olavs Hospital has. One option is to develop a software program for a computer and another is to implement an application that can be used on any platform. By developing a software program for a computer, the access to information via mobile phones or tablets will be non-existing. By implementing an application, there will be easier access to information through several devices. There are several types of applications that can be implemented for this task, namely *web-based*, *native* or *hybrid* [1].

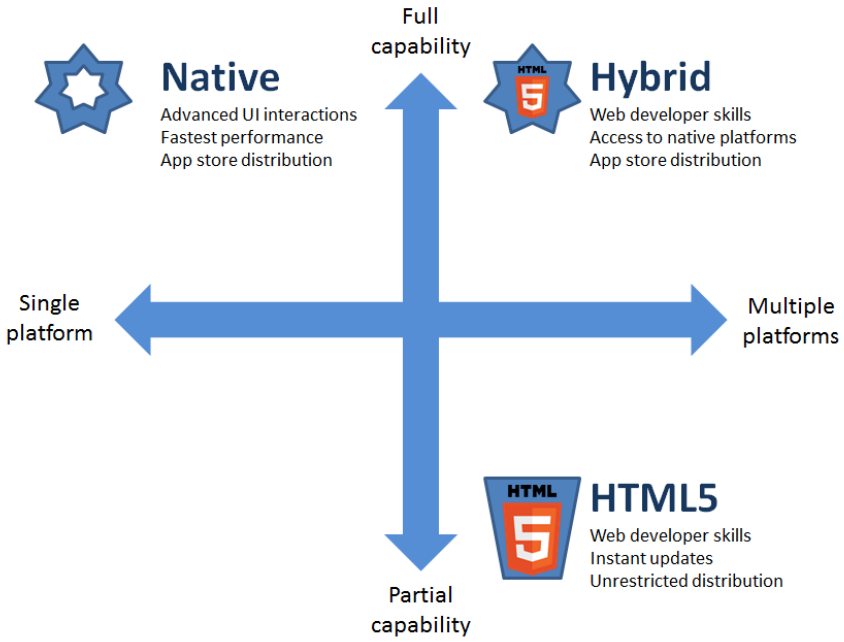


Figure 2.1: Pure Native vs. Hybrid Native vs. Mobile Web

2.4.a Web-based applications

Web-based applications are client-server software where the client runs in a web browser. The so called web application was introduced in the Java language in the Servlet Specification version 2.2. by James Duncan Davidson and Danny Coward [9]. Web applications makes access to the app easy since no software installation is needed. It is convenient since the app is developed independently of different web browsers, but it requires a stable Internet connection.

2.4.b Native applications

A native application is an app that is developed for a specific device or a particular platform. This enables the developers to optimize the performance according to the given device, but by making native applications there are many different programming languages that need to be used in order to fit most operating systems.

2.4.c Hybrid applications

A hybrid application is one typically developed in a web-language and built, or compiled, into native apps for the desired platforms or devices. This enables access to the device's hardware, like camera and accelerometer, and is developed in one language and could be adjust for a specific device after compilation.

2.5 Evaluation criteria

For the patient group that are being considered in the project, the customer stated that doctors in the emergency room often assume the wrong laws both when providing and documenting treatment. The main reason for this is, according to the customer, that the information about the correct laws is difficult and time consuming to obtain in their current system, especially in emergent events. Even though all doctors study these laws throughout their education very few prioritize to remember these and would rather keep their focus on the practical aspect of treating patients.

For the solution, the main evaluation criteria will be to reduce the difficulty of the, now time consuming, task that is to retrieve the correct laws, and thereby increase the correctness of the laws used in practice and documentation.

The customer expressed early interest for the tool to be accessible on as many devices as possible. Therefore there was early on decided upon to make an app that would fit mobile phones, tablets and computers. There are primarily three different types of apps; native, web-based and hybrid.

Even though the Internet connection in the ER is relatively stable, there can be periods without any Internet access, and it is unfortunate to limit the application by only being used if the user has access to the Internet. For this reason, and the fact that a web-based app must work on several different types of browsers, which can lead to big differences in the user interface, the development team chose not to make a web-based app.

Because of the customer's wish of making the tool applicable to every platform, the option of making the application native was quickly abandoned, namely because the time frame given would not be sufficient to make apps for all possible mobile operating systems. After researching for different frameworks, a hybrid application seemed to be the most promising.

2.6 Similar applications and technologies

There is almost an app for every desired task in today's technology world, but after researching for similar applications online the group was unable to find one that had similar functionality to the "Akuttjus" application.

Although no application was found that did exactly what the group wanted the app to do, there are some applications that could relate to the desired functionality in the "Akuttjus" app. Aid to Capacity Evaluation (ACE) is one of those. It is a tool that is used for evaluating if a patient is competent to give consent to a decision about his/hers own medical treatment. The application will not be a tool for deciding if a person is competent to give consent, but in order to use the application optimally, a doctor needs to determine if his/her patient is able to make such a decision.

There exists some apps that are made explicitly for hospitals or doctors. One app that is worth mentioning is called VTE [12].

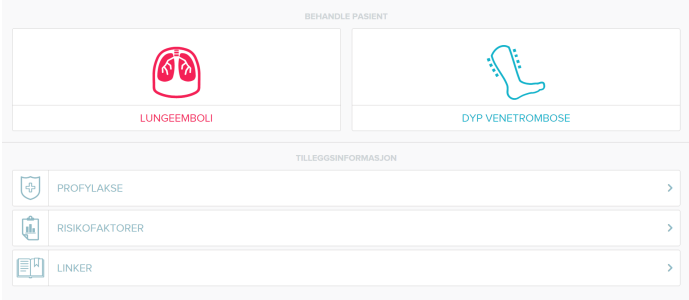


Figure 2.2: VTE front page

This is a tool that is used for assessment, diagnostics, and treatment of the disease Venous Thromboembolism, which is used by medical personnel only. This application is developed in collaboration with the St. Olavs Hospital and is made for the purpose of helping the doctors make decisions about certain patients that have this disease. This app has some similarities with the "Akuttjus". They are both apps that are developed for and in collaboration with medical personnel, they assist the users of the application by providing knowledge, and they include crossroads with decisions which guide the doctors in the right direction. The pros and cons of VTE will be taken into account when designing the structure and the graphical elements of "Akuttjus"

Name	Pros	Cons
VTE	- Good design - Easy to use(on computer and tablet) - The idea of the structure is good - Summary of chosen elements	- Not mobile-friendly - To many levels of information (from beginning to end)

Table 2.1: Pros & cons with VTE

2.7 Standards

This sections outlines some conventions to guide all team members to keep the same standard. The aim is to increases the readability of the code and keep a consistency of design throughout the application.

2.7.a Design standards

For the application to be distributed in the *AppStore* [19], the design must follow Apple's Guidelines for Human Interaction [18]. This application will be developed for more platforms than iOS, and Apple's guidelines gives an industry approved standard for usability and layout. It outlines design strategies and the use of user interface elements.

2.7.b Code standards

The aim for explicitly defining a coding convention is to ease the maintenance of the application. The lifetime of the app is long, and after the implementation is done by the students it is critical that other engineers can continue to develop it.

Naming convention

Camel Case will be used for all naming of variables, functions, classes and interfaces [27]. The phrase are written without white space and the next words initial character is capitalized. e.g. `functionName` and `ClassName`. All names should be descriptive and informal. By naming variables with their content, functions, and methods with what they do, the readability of the code will be elevated. Every class and function of an interface should begin with a comment describing how to use it. Every constant throughout the application should be named with all letters in uppercase. e.g. `CONSTANT`.

2.8 Summary

Hospital personnel have problems with documenting the correct laws in the current situation. A major part of this problem is rooted in systems that are currently used and the fact that most of these are rather old, like the EQS. The wanted situation is one where anyone working at the hospital can rely on the knowledge tool the group is creating and where the application is used for educational purposes as well as a knowledge tool.

There are several conceivable solutions, but the decision fell upon a hybrid application. After thorough analysis, the team could not find an application which possessed the wanted functionality, but the VTE app has a similar structure to what the group would like to implement. The development methodology for this project will be mainly Scrum, because it suits the problems and possible solutions best.

3 Planning

In order for a project to succeed, planning is crucial, because without a structured timetable the efficiency of the work might be drastically reduced. This chapter contains the schedule for the work that needs to be done and the methods of implementing these plans. The project organizational roles will also be presented.

3.1 Organizational demands

The purpose of this project is to teach students how to run a project from start to finish. As a result of this, there are few explicit demands from the faculty and the course lecturers. There are however some guidelines that need to be followed.

- The project groups need to attend all course lectures. It is acceptable to miss one lecture.
- If there exist women in the project group, they will be the project manager.
- Each team member are expected to work 25 hours per week.
- There will be weekly meetings with the group's advisor.
- Every week the group needs to deliver a status report for its advisor.
- There will be bi-weekly meetings with all the team leaders and advisors.

3.2 Project organization

This section includes the different roles and responsibilities each team member has. These roles reflect which skills each member can contribute the most to. Although the roles are set, they are not finalized. Which means that participants of the project can work on different themes and not only the roles specified in this section. Further the section includes information about the different types of meetings. To ensure good communication, that roles are followed, and that everyone shows up to meetings, discussions, and general working hours, we created a set of rules. These are listed in appendix G.

3.2.a Roles and responsibilities

Except from the role of team leader, which was set by the organizational demands, the team members were assigned roles based on experience and own choice. The different roles and responsibilities are listed below.

Project leader - Sarah Serussi

The main role of the project leader is to have an overview over the project, and solve conflicts if any occur. In addition the project leader should also manage the teams resources and prepare meeting agendas. Lastly this person shall keep the moral of the group at a high standard.

Customer contact - Sarah Serussi

Firstly the customer contact shall have a good and organized dialogue with the customer, as well as forwarding and sharing all mail and its content. Meetings with the customer should be arranged by this person. The customer contact is the only one to have direct communication with the customer.

Advisor contact - Sarah Serussi

The advisor contacts main role is to maintain the communication with the advisor. Further this person should deliver a weekly status report to the advisor.

Scrum master - Morten Kartevoll

The scrum master should help the project team reach its goal at the end of each sprint. Maintaining the scrum methodology and promote self organizing are also important tasks. In addition to promote self organizing the scrum master should also aid the project team in events to ensure that progress is made.

2nd in command - Morten Kartevoll

2nd in commmand is the project leader's right hand. This person should relieve the project leader of excessive workload, and shall step into the role as project leader when needed.

User interface designer - Morten Kartevoll

In addition to have the main duty of choosing how the user interface is going to be, the designer should also create and gather icons, logos, and other graphics needed.

Quality assurance responsible - Petter Bakkan Johansen

The main role of the quality assurance responsible is to ensure that the quality of the final report and product is as high as it can be. This includes checking that rules are respected and standards are used correctly.

Test leader - Thor-Martin Abrahamsen

The test leader have the main responsibility for testing. This includes ensuring that test plans are executed, as well as delegating testing tasks to the rest of the implementation team.

System architects - Are Gotteberg Haartveit, Eirik Tvedt Jensen

The system architects main job is the architecture of the software. They are also in control of the software code, and the delegating tasks that needs to be done related to coding. Lastly they should make sure that everyone respects the chosen pattern.

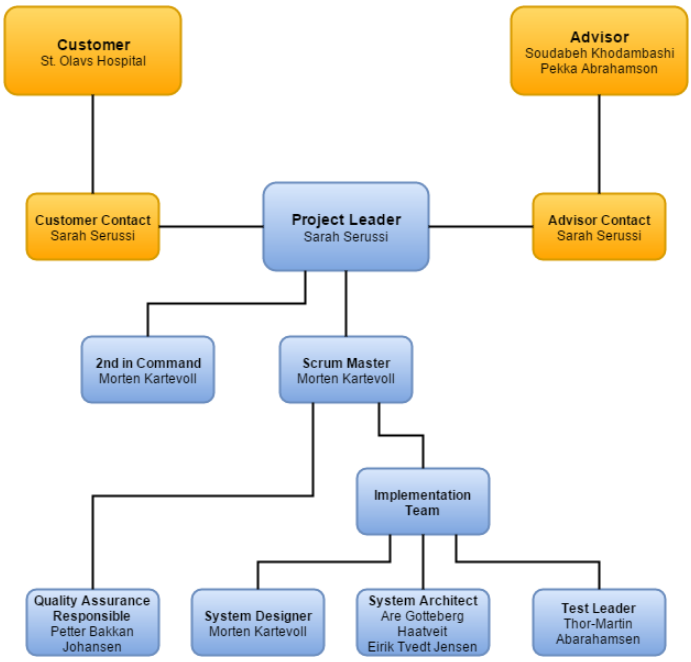


Figure 3.1: Project Roles

3.3 Quality assurance

In order to prevent mistakes and to avoid an unnecessary amount of workload, it is important to have quality assurance continuously throughout the project. This section explains the concrete measures that have been taken in order to ensure the quality of the report and the implementation.

3.3.a Time of response

In general, the time of response is the time it takes from an inquiry is made until the beginning of a response. In this context, the time of response is the time it takes for one of the involved parties

to pose a question or a demand until one of the other involved parties delivers an answer.

Description	**Time of response**
Approval of minutes of customer meeting	24 hours
Feedback on phase documents the customer would like for review	48 hours
Approval of phase documents	48 hours
Answer to a question	24 hours
Providing requested documents	48 hours

Table 3.1: Time of response table

3.3.b Routines for producing work with high quality

In order to ensure quality of the produced items within documentation and implementation, one team member has been assigned the responsibility of making sure every item is checked by at least one person that has not been involved in the respective work. In this project, the group will mostly be working in pairs, such that the work produced will constantly be checked by one other team member. If there are some tasks that are completed by only one person, the quality assurance person will assign one team member to check their work. In addition, the documentation that can be affected by the customer's requirements and desires will be approved by the customer in order to ensure the correctness of it.

3.3.c Calling for a meeting

For every meeting with the customer, the team should discuss and plan what will be reviewed for the next meeting. The customer will receive an e-mail no later than 72 hours before the meeting will take place, and because the customer has a less flexible timetable than the students in the team, the customer will set the time and place of the meeting.

The weekly meetings with the advisor will be decided either by e-mail at least 48 hours before it will take place, or the next meeting will be decided at the end of the previous meeting. A status report and relevant deliveries will be sent to the advisor at least 24 hours before the meeting will take place. The time and place will be agreed upon via e-mail.

3.3.d Agenda for a meeting

In order to have a structured agenda for the meetings with the advisor, customer and the team, there will be used a template for each meeting. This template contains information about who is present, questions to ask, and other notes that will be discussed. The templates can be found in appendix B.

3.3.e Minutes of a meeting

There will be written a summary of all the meetings with both the customer, advisor and the group. All decisions, actions and clarifications are important to document to make it easier to review them later in the project. In order to make sure there are no misunderstandings between the team and the customer, the team and the advisor, or within the group, there will be made an oral recap at the end of each meeting. In the recap there will be a discussion to confirm the correctness of the content.

3.4 Phases

There are six phases that are documented in this report, excluding the final phase that lasts from the 12.11.2015 to the 18.11.2015 which contains the work that is done regarding the final presentation of the team's application. Each phase contained the task of writing on the report in addition to the description mentioned in table 3.2.

Our project start began with an introduction phase. There was held a kick-off for the groups where the main goal was to get to know each other. The group made team rules, objectives, project goals, and a team name, Tiaco.

In the second phase the group focused on preliminary studies. The project task needed to be understood correctly, and there needed to be done research on similar products as well as the tools needed to be used in the project. In addition to doing research and documenting it, there were also some meetings with both customer, advisor and the group.

After the preliminary studies finished, the real work began. In the first sprint the team started making three different prototypes and conducted usability tests within the group and with the customer. There was high focus on determining which flow and structure the application should have, and which content the app should contain.

The second sprint consisted of implementing the revised prototype and its functions, as well as starting with the implementation of the application. The functionality of the app was implemented in the beginning this sprint, while the actual layout and styling was started at the end of the second sprint.

In the third sprint, most of the time was used to finish the implementation of the application. The team also started implementing the administrator view at this time.

The fourth, and final sprint, consisted of finishing the implementation of the app, both main view and administrator view, finishing the styling of the application, conducting usability tests with medical interns, and finalizing the report.

Phase	Description	Week
Introduction	Getting to know each other, Project setup	35
Preliminary studies	Planning, Report setup, Research	36-37
Sprint 1	Prototyping, Usability tests	38-39
Sprint 2	Implementation	40-42
Sprint 3	Finalizing main view	43-44
Sprint 4	Finalizing admin view, User testing	45-46

Table 3.2: Project phases

3.5 Work breakdown structure

A Work breakdown Structure (WBS) is a decomposition of a project into smaller tasks. By decomposing a large project into packages it becomes easier to get an overview of what needs to be done, thus an overwhelming task might still be achievable.

The team's WBS is visualized in table 3.3. The estimated hours in this table is based upon the organizational demand that states that each team member of the group are expected to work 25 hours per week, for 11,4 weeks (excluding the week before the presentation), which gives a total of 1710 hours. The team found this expectation to be unrealistic, because the team members had other major project in other courses. Therefore the team estimated the total amount of hours as 100 hours less than expected, which equals 1610 hours.

3.5.a Explanation of the tasks in the WBS

This section explains the different tasks in table 3.3.

- **Research:** The research task consists of the team's research at the start of the project. It includes research on technologies that is considered to be used in the project, as well as background information about the St. Olavs Hospital, and similar existing software.
- **Meetings and planning:** The meetings task consists of group meetings, advisor meetings, team leader meetings, and customer meetings, as well as course lectures. The planning task consist of the planning that was made before the start of sprint 1, as well as the planning that is made at the beginning of each sprint. The reason that the end date is set to the 12.11.2015, is because of the planning that will be made for the presentation.
- **Learning new technology:** The task of learning new technology consists of the time spent on getting to know the tools that were used during the project, such as Meteor, LaTeX, HTML, and Ionic.
- **Design:** The design task consists of the work that was done with the prototyping and the decision making of how the application would be styled.
- **Implementation:** The implementation task consists of the programming that was done on the app, both front-end and back-end, as well as the styling of the applications. The testing is also included in the implementation task, which consists of the test that were executed

between sprint 1 and sprint 4. It includes prototype testing, user testing, and integration testing.

- **Documentation:** The documentation task consists of the work that was done with the report.

Task	From date	To date	Estimated hours	Actual hours
Research	25.08.15	14.09.15	40	48,5
Meetings and planning	25.08.15	11.11.15	330	250
Learning new technology	04.09.15	22.10.15	70	86
Design	03.09.15	31.10.15	170	148
Implementation and testing	30.09.15	09.11.15	470	452,5
Documentation	02.09.15	12.11.15	530	592,5
Total	**25.08.15**	**12.11.15**	**1610**	**1577,5**

Table 3.3: Work breakdown structure with hours

Task	Preliminary study	Sprint 1	Sprint 2	Sprint 3	Sprint 4
Research	26	23,5	0	0	0
Meetings	73	61,5	65	27,5	23
Learning new technology	23,5	15	34,5	0	0
Design	28,5	44,5	27	40	0
Implementation	0	18,5	112	188,5	133,5
Documentation	61,5	145	150,5	34,5	201
Total	**211,5**	**308**	**389**	**290,5**	**357,5**

Table 3.4: Work breakdown structure for each phase with actual hours

3.6 Gantt Diagram

A Gantt diagram is used when illustrating the project shedule. The first Gantt diagram was created by Karol Adamiecki in 1896, but he did not publish his work until 1931. 'Gantt' comes from the first person to publish such a tool: Henry Gantt, which published this around 1910-1915. [11] This diagram is suppose to graphically display both start and end dates of elements in the project. In order to make the Gantt diagram, a scheduling program called Gantt project was used [7.2.e]. The Gantt diagram can be found in a larger format in Appendix F.

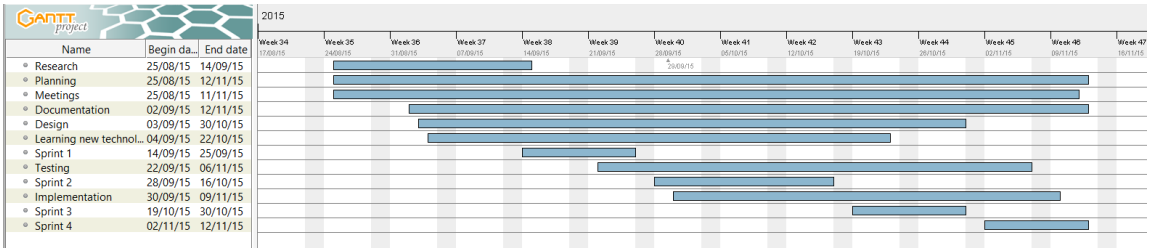


Figure 3.2: Gantt diagram

3.7 Milestones

A milestone is used to mark a specific point along the project lifetime. These points do not interfere with the duration of the project, as they are only major progress points that is important to reach at the given time. This project's milestones are listed in the following order.

Milestones	**Finished date**
Preliminary studies are finished	13.09.15
Show the first prototype	22.09.15
First sprint is finished	27.09.15
Skeleton code is ready	01.10.15
Show second prototype	09.10.15
Second sprint is finished	18.10.15
Third sprint is finished	01.11.15
Main view of application is finished	02.11.15
User testing with medical interns	04.11.15
Implementation is finished	09.11.15
Fourth sprint is finished	12.11.15

Table 3.5: Milestones in the project

3.8 Summary

In this chapter the organizational demands have been presented, as well as a detailed project organization description. The different roles have been covered by the team members, and some team members have several responsibilities. An overall view and a more descriptive outline of the project's phases is included.

4 | Requirements

This chapter describes the system requirements. They are divided into functional and non-functional. These are explained through use-case diagrams. Both high-level and low-level use case diagrams will be used. A flowchart of how the customer wanted the flow of the application will be provided, explained, and presented graphically.

4.1 Flowchart from customer

For the first customer meeting, the customer came prepared with a flow chart of his vision for the application. The flow chart contained four categories with accompanying lanes of the flow from start to finish. The path between the categories and the rest of the steps were filled with the data that was to be encompassed in the text fields.

The customer expressed that the flow chart might change, and that there was no need to focus on the textual content, but that the main focus was the flow of the application.

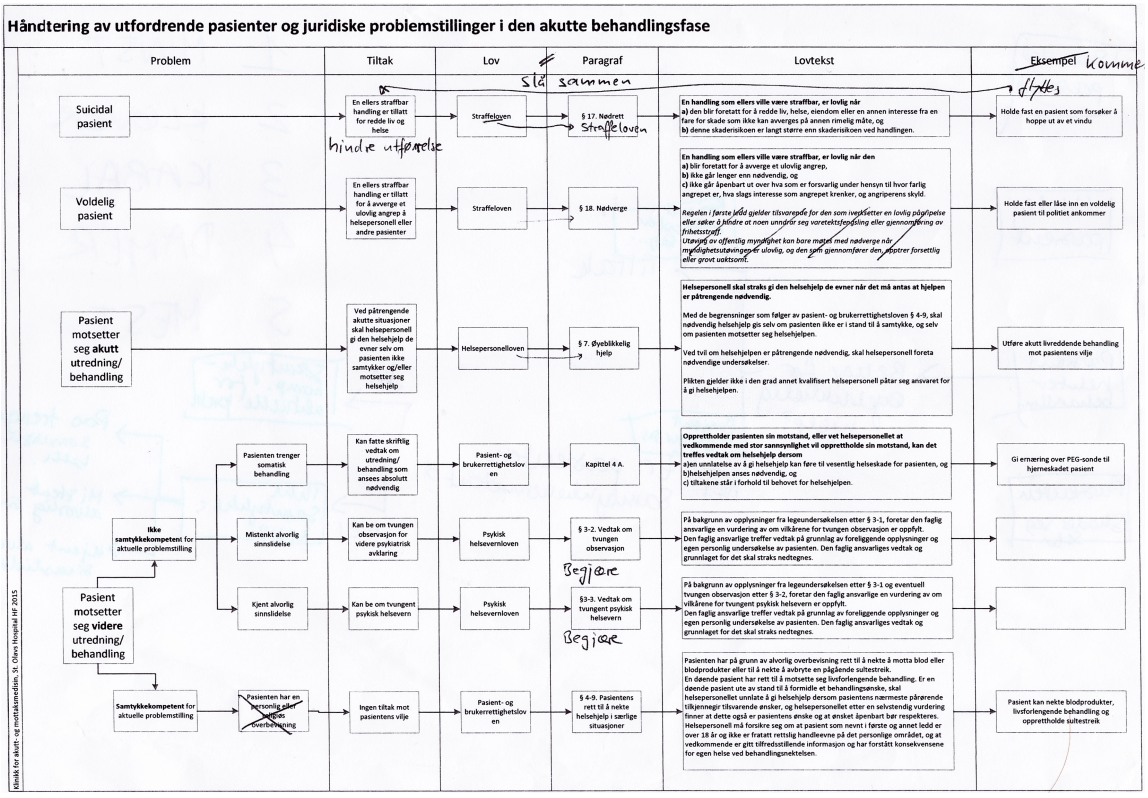


Figure 4.1: Flowchart from customer

After expressing our ideas to the customer the flowchart was quickly changed to a more programmable solution. These ideas included combining two of the categories (”Pasienten motsetter seg akutt behandling” and ”Pasienten motsetter seg videre behandling” which translates to ”The patient resists life threatning treatment” and ”The patient resist further treatment”) into one common category (”Pasient nekter behandling” which translates to ”Patient resists treatment”).

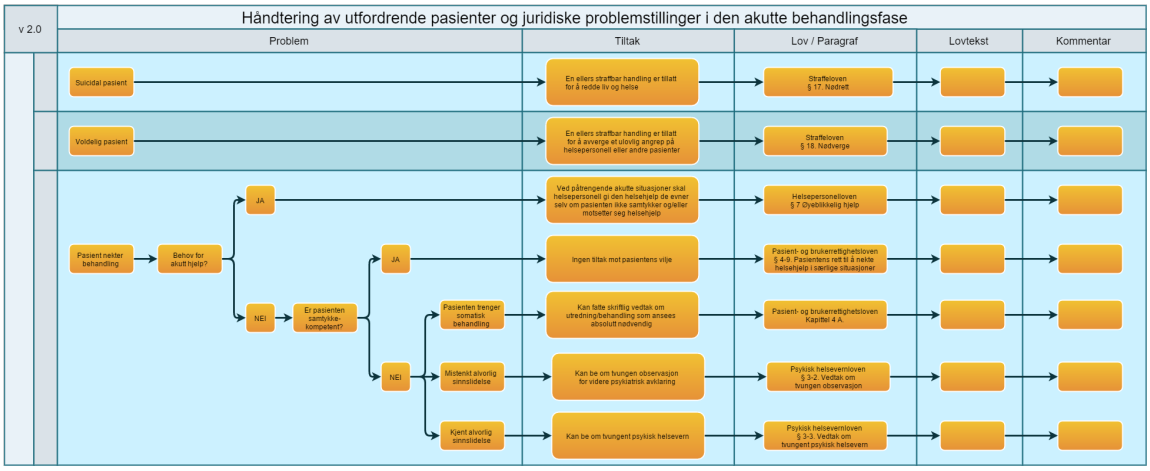


Figure 4.2: Edited flowchart from customer

4.2 Methods for requirement analysis

The flow chart that was given to the group played a major part in defining the requirements for this project. By studying the flow chart the group was able to think from a programming perspective of how the application should be structured. In addition to forming requirements from the flow chart, there were also held an interview with the customer to detect other requirements that could not be extracted from the flow chart. During the first interview, the customers ambitions were clearly expressed, among others that he wanted the application to run on multiple platforms, such as smart phones, tablets, and computers. The interview in its wholeness can be found in Appendix H.

As well as having regular meetings and interviews with the customer, the team were invited to a tour of the ER at the start of the project. The team got a tour of the different rooms and entrances, which was useful to understand how and when the patients are admitted at the reception. By observing how the doctors interact with patients, it was clear that in order to have the possibility to use the application during an emergency situation, it needed to work very fast.

The customer meetings were occasionally combined with a user test of the application, where the customer was the test subject. During the first user test, the requirement of constraining the amount of clicks, was discovered, which is described in requirement F9.

Regular meetings with the team's advisor were also used to discuss requirements. The requirement of having an administrator view for the application was conferred with the advisor, and added as a requirement during sprint 2, in chapter 10.

4.3 Functional requirements

The functional requirements describe the specific behavior of the product that is to be made. These are listed below.

ID	Description	Priority
F1	The applications must have a start menu (with decided amount of categories)	High
F2	The application must show a summary of the outcome in the end of each iteration	High
F3	The application should have visual feedback when pressing a button	Low
F4	The application must have a "Return" button which will take the user one step back	High
F5	The application must have a consistent design and user interface	High
F6	The application will have a "Home" button that takes the user back to the start screen	Medium
F7	The application will give an exit prompt before an administrator deletes an item	Medium
F8	There should be an option in settings to "Turn off exit prompts"	Low
F9	The application will strive to display relevant content in as few clicks as possible	Medium
F10	The application will have a back-end	Medium
F11	The application will have an administrator view	Medium
F12	The application will be able to be used offline for smart phones and tablets	Medium

Table 4.1: Functional requirements

4.4 Non-functional requirements

Non-functional requirements specify the criteria that a system should have in terms of evaluating the system's quality attributes.

ID	Quality Attribute	Description	Priority
NF1	Performance	The application must not take longer than 5 seconds to boot	High
NF2	Performance	The application must have an up time of at least four nines	High
NF3	Performance	The application must at most use one minute to navigate the user from start to documentation	High
NF4	Performance	The system will be scalable	Medium
NF5	Modifiability	It will be possible to change the data in any text box	Medium
NF6	Modifiability	It will be possible to add categories to the application	Medium
NF7	Modifiability	No data will be hard-coded	Medium
NF8	Reliability	The application must link a certain situation to the correct laws that apply	High
NF9	Reliability	The content in the application must not be ambiguous	High
NF10	Usability	The application must have an easy and intuitive navigation system	High
NF11	Usability	The administrator of the application will be able to add and edit content	Medium
NF12	Usability	The language in the application must be understandable	High
NF13	Usability	The application must be responsive	High
NF14	Usability	The user will not use more than 1 hour to learn the basics of the application	Medium

Table 4.2: Non-functional requirements

4.5 Use case

Ivar Jacobson described textual, structural, and visual modeling techniques for specifying use cases in the year of 1986. [24] Use cases consist of a conceptual diagram accompanied by a textual description. The purpose is to describe series of actions performed on the system.

4.5.a Actors

An actor is someone performing the actions on the system. Actors are divided into two categories: **users** and **administrators**.

User

A user is anyone that access the system. This would be *end users*, *nurses*, *doctors*, and so on.

Administrator

An administrator is a user with extra privileges in form of access to the system. Some functionality is only accessible to administrators, described in the high-level use case below.

4.5.b High-level use case

A high-level use case is where each separate low-level use case are combined into one general case that shows the overview.

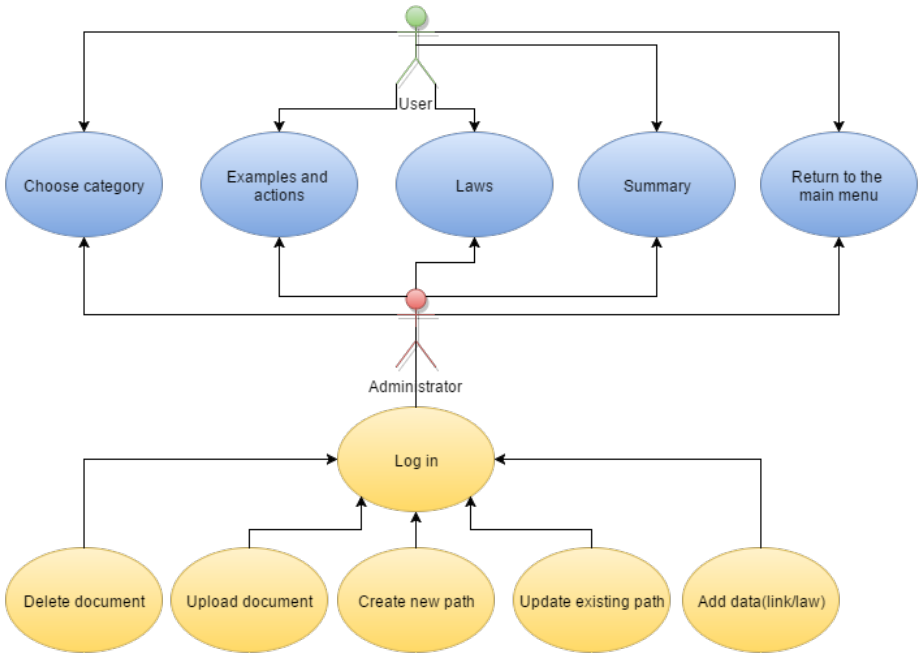


Figure 4.3: High-level use cases

4.5.c Low-level use cases

The low-level use cases present one single requirement for the system.

Figure 4.4: UC1 - Choose category

Use case name:	**UC1 - Choose category**
Actors related:	User
Trigger:	A person and presses one of the categories in the main menu
Objective:	Categorize the patient
Pre-conditions:	The application is running
Response:	A set of choices in the category
Normal event flow:	1. User chooses category that suits the situation
Variations in the event flow:	N/A
Scenario:	A user wants to categorize a patient so he can decide what he can do with the situation, so he chooses category that suits the situation

Table 4.3: UC1 - Choose category

Figure 4.5: UC2 - Examples and actions

Use case name:	**UC2 - Examples and actions**
Actors related:	User
Trigger:	A person presses the action button
Objective:	Check if the situation fits the examples and what actions to take
Pre-conditions:	The application is running and a category is chosen
Response:	The user is navigated to the action screen
Normal event flow:	1. User chooses action and presses the button and is navigated to the action screen.
Variations in the event flow:	1.a. Press the return button from the law screen or change the tab.
Scenario:	A user wants to check what actions to take with a categorized patient so he navigates to the action screen.

Table 4.4: UC2 - Examples and actions

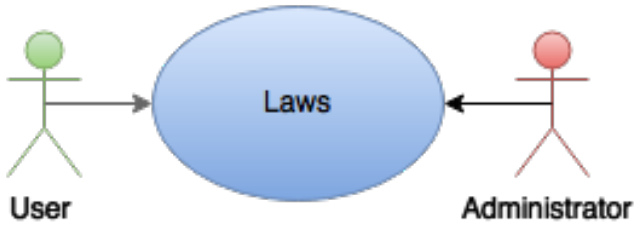


Figure 4.6: UC3 - Laws

Use case name:	**UC3 - Laws**
Actors related:	User
Trigger:	A user presses the law button
Objective:	Check what laws(s) relates to the situation
Pre-conditions:	The application is running and a category is chosen
Response:	The user is navigated to the action screen
Normal event flow:	1. User chooses action and presses the button and is navigated to the action screen
Variations in the event flow:	1.a. Press the return button from the current page if the law page was previous accessed, or change the tab to law
Scenario:	A user is unsure what laws are related to the current situation and checks with the law page.

Table 4.5: UC3 - Laws

Figure 4.7: UC4 - Documentation

Use case name:	**UC4 - Documentation**
Actors related:	User
Trigger:	A user presses the Documentation tab
Objective:	Get an overview of what has been done in the last iteration, and find information about documentation
Pre-conditions:	The application is running and a category is chosen
Response:	The user is navigated to the documentation page
Normal event flow:	1. User maneuver to the documentation tab. 2. User can now see in what category the patient is, law paragraph, related documents, and useful documentation information
Variations in the event flow:	N/A
Scenario:	After a doctor is finished reading what the laws and actions relate to the current patient, the doctor wants to take a look at what he should document, and access to relevant documents/schemes.

Table 4.6: UC4 - Documentation

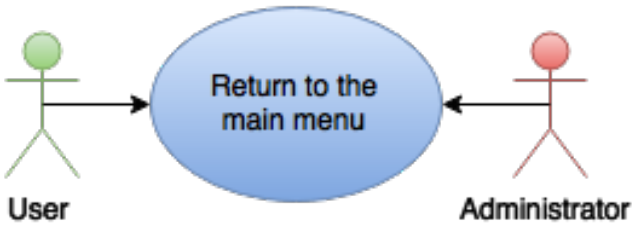


Figure 4.8: UC5 - Return to main menu

Use case name:	**UC5 - Return to main menu**
Actors related:	User
Trigger:	A user presses the main menu button
Objective:	Return to the initial state
Pre-conditions:	The application is running and the user is anywhere but in the main menu
Response:	The user is navigated to the main menu
Normal event flow:	1. User presses the home icon. 2. The application navigates the user back to the main menu
Variations in the event flow:	N/A
Scenario:	The user wants to return to the main menu for a new iteration and presses the home icon

Table 4.7: UC5 - Return to main menu

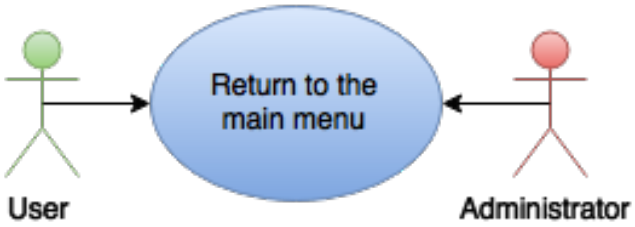


Figure 4.9: UC6 - Exit prompt

Use case name:	**UC6 - Exit prompt**
Actors related:	User
Trigger:	A user presses the home icon when in the summary
Objective:	Check if the user really wants to return to initial state
Pre-conditions:	The application is running and the user is in the summary
Response:	An exit promt is displayed on top of the summary
Normal event flow:	1. The home icon is pressed. 2. An exit prompt becomes visible. 2.a. Press cancel to stay in summary. 2.b. Press exit to return to main menu
Variations in the event flow:	N/A
Scenario:	A user accidentally presses the home icon when reading the the summary, and does not want to discard the summary (wants to cancel the abort).

Table 4.8: UC6 - Exit prompt

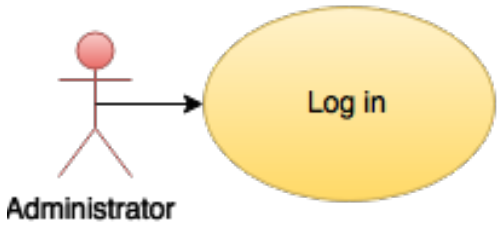


Figure 4.10: UC7 - Login

Use case name:	**UC7 - Login**
Actors related:	Administrator
Trigger:	An administrator logs on the system
Objective:	To get access to additional functionality
Pre-conditions:	The application is running and the user in settings
Response:	The administrator logs into the system and get access to delete and add documents in the summary screen
Normal event flow:	1. The administrator navigates to the login area. 2. After writing the username and password the administrator gets access to the additional functionality
Variations in the event flow:	N/A
Scenario:	As an administrator, I want to login to the system and get access to additional functionality

Table 4.9: UC7 - Login

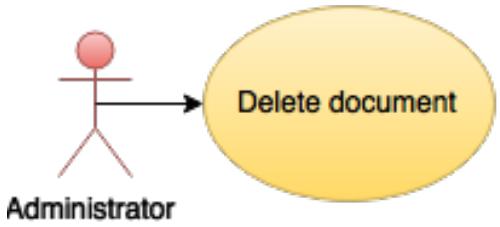


Figure 4.11: UC8 - Delete document

Use case name:	**UC8 - Delete document**
Actors related:	Administrator
Trigger:	An administrator deletes a document in the summary screen
Objective:	To remove documents that are not up to date
Pre-conditions:	The application is running and the administrator has logged into the system
Response:	The document which is deleted will be removed from the summary
Normal event flow:	1. The administrator navigates to the summary he wants to delete an document in. 2. The administrator presses delete on the document which must be removed. 3. Exit prompt
Variations in the event flow:	N/A
Scenario:	As an administrator, I want to be able to delete a document which is outdated.

Table 4.10: UC8 - Delete document

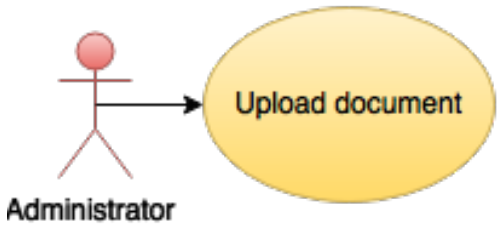


Figure 4.12: UC9 - Upload document

Use case name:	**UC9 - Upload document**
Actors related:	Administrator
Trigger:	An administrator uploads a document in the summary screen
Objective:	To add new documents which are needed in a specific situation
Pre-conditions:	The application is running and the administrator has logged into the system
Response:	There will be a new document in the summary page
Normal event flow:	1. The administrator navigates to the summary he wants to upload an document in. 2. The administrator presses upload. 3. Correct files are chosen and uploaded
Variations in the event flow:	N/A
Scenario:	As an administrator, I want to be able to upload a document which is up to date in the specific stituation

Table 4.11: UC9 - Upload document

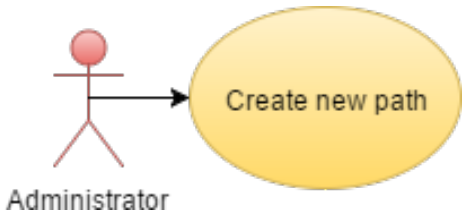


Figure 4.13: UC10 - Create new path

Use case name:	**UC10 - Create new path**
Actors related:	Administrator
Trigger:	An administrator creates a new path using the flow creation tool
Objective:	To create a new categorization of a patient which includes examples, laws, and documentation
Pre-conditions:	The application is running and the administrator has logged into the system
Response:	A new category of patients in main screen, and the subsections under it(examples, laws, documentation, and potential subcategories.
Normal event flow:	1. The administrator chooses the flow creation/update tool 2. The administrator follows the guide for creating a new path. 3. The administrator creates the path by pressing a create button
Variations in the event flow:	N/A
Scenario:	As an administrator, I want to be able to create a new path which categorize a set of patients. I want to be able to add laws, examples, documentation, and the potential sub path which is needed to categorize a patient.

Table 4.12: UC10 - Create new path

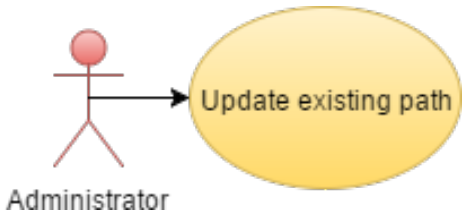


Figure 4.14: UC11 - Update existing path

Use case name:	**UC11 - Update existing path**
Actors related:	Administrator
Trigger:	An administrator updates an existing path with the flow creation tool.
Objective:	To update an existing path to ensure the correct laws, categorization, and documentation is used.
Pre-conditions:	The application is running and the administrator has logged into the system
Response:	An existing path or element is updated.
Normal event flow:	1. The administrator chooses the flow creation/update tool 2. The administrator follows the guide for updating an existing path 3. The administrator updates the path by pressing a update button
Variations in the event flow:	N/A
Scenario:	As an administrator, I want to update an path so that it is up to date with the current laws, documentation, examples, and path

Table 4.13: UC11 - Update existing path

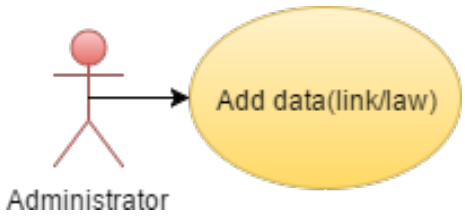


Figure 4.15: UC12 - Add data(link/law)

Use case name:	**UC12 - Add data(link/law)**
Actors related:	Administrator
Trigger:	An administrator adds new content in the law or link page
Objective:	To add new content to links and the relative law page
Pre-conditions:	The application is running and the administrator has logged into the system
Response:	A new element will be added to the link or the relative law page
Normal event flow:	1. The administrator chooses the add content tool 2. The administrator follows the guide for adding new content 3. The administrator adds the content by pressing add content
Variations in the event flow:	N/A
Scenario:	As an administrator, I want to be able to add new laws or links to the respective pages.

Table 4.14: UC12 - Add data(link/law)

4.6 Summary

This chapter listed 12 functional requirements and 14 non-functional requirements. It identifies two actors on the system, a user and an administrator, and present nine use cases for them. There is a clear difference between the actors in the use cases. The first collection are viewed from the user's desires, where the focus lies on the requirements specification. The last use cases focuses on maintenance, and are viewed from an administrator's perspective.

5 Development Methodology

This chapter presents some of the known development methodologies in software development and the chosen methodology for this project. Each subsection contains a short description of the theory behind the respective methodology with some advantages and weaknesses.

5.1 Waterfall model

The waterfall model is a sequential design process, which is used in software development processes and was first formal described in an article from 1970 by Winston W. Royce. [34] This model flows steadily downwards through stages like analysis, design, testing, implementation, and maintenance, hence the name waterfall. In this model, each phase of a project needs to be finished before starting with the next one. This ensures structure, but not flexibility.

One of the major advantages with the waterfall model is time and effort saved in a project lifetime. Because of its concrete structure the possibility of going backwards is non-existent, and the time that might have been used to analyze the previous stage is saved. Although time and effort might be saved, the central idea of the model may lead to redesign, redevelopment, and retesting, and therefore increasing costs. There are variations of the waterfall model which correct flaws that the pure model struggles with, such as waterfall with sub projects, waterfall with overlapping phases, and waterfall with risk reduction.

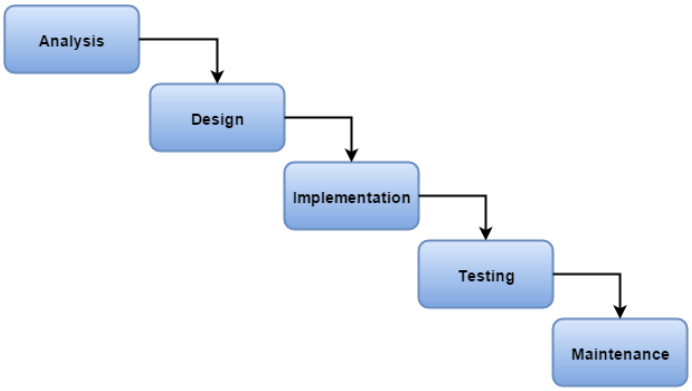


Figure 5.1: Waterfall model

5.2 Scrum model

Scrum is an iterative and incremental agile software development methodology for managing product development. The concept of this methodology was described in The Manifesto of Agile Software Development in 2001, even though the method itself had been around for years. [8] This written material gathered all the best practices in one place. Flexibility defines the scrum model and is a complete product development strategy, where the team assigned to the project works to reach a common goal as a unit. This methodology encourages close physical co-location or active on line participation from every member in the team. In addition daily face-to-face meetings is recommended for disciplining team members.

One of the key principles with scrum is its focus on maximizing the scrum team's competence to swiftly respond to a customer's emerging requirements. In other words the scrum approach accepts that a problem may not be fully understood or defined, but focuses more on quick response to changes. Although Scrum ensures effective use of resources such as money and people available to the project, problems may emerge when people are not committed to it. In large teams the efficiency may decrease, as Scrum relies on every member of the team to do their part.

There are several roles in the scrum framework. These roles consist of a scrum master, a development team, and a product owner. Even though there may exist other roles in a real project, scrum does not define other roles than the ones encountered in this section.

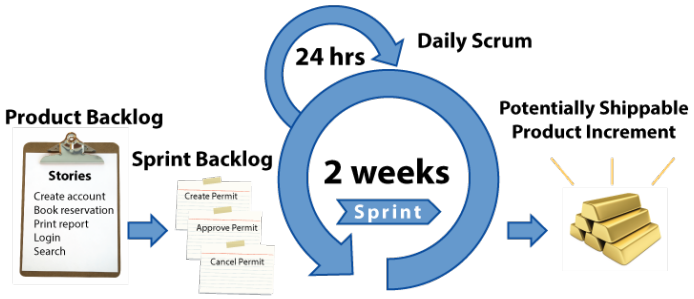


Figure 5.2: Scrum model [33]

5.2.a Product owner

The main representative of the stakeholders is the product owner, which thereby is the customer's communication channel to the scrum team. The main idea for the product owner is to compose a list of items wanted in the product. Also the product owner should not be involved in the teams development of the technical aspects.

5.2.b Development team

The development team's goal is to deliver potentially shippable parts when each scrum period ends. Development teams are multifunctional and self organized, where the individuals work in the following categories: analysis, design, implementation, testing, technical communication, and documentation.

5.2.c Scrum master

The scrum master's role is to help both the development team and the product owner. Helping the development team to reach its goals and maintain the scrum methodology, promote self-organizing, help the product owner manage the product backlog, and aid the development team events to ensure progress, are some of the tasks the scrum master strive for.

5.3 Kanban

Kanban is an agile development methodology where the focus lies on just-in-time delivery [35]. There exists a work queue where the tasks are sorted after priority. After finishing a task, the team members pluck the next task that is on top of the backlog. This methodology is inspired by Toyota Production System and Lean Manufacturing In Kanban there needs to be an assigned person, usually the product owner, that continuously updates the backlog with the most important tasks.

The Kanban approach is very flexible. The tasks that are in the product backlog may be re-organized whenever the priority of a different task is higher or lower. This approach does not describe specific process steps, thus changes are constantly made to the system.

One advantage with Kanban is that it is very receptive to change. This is useful in a project where either the requirements or deadline are often altered or if the priorities of the different tasks change. However if the team is undisciplined, the freedom that Kanban comes with might be taken advantage of.

5.4 Development methodology decision

After reviewing the different types of development methodologies there were three candidates for our decision, where the Kanban and Scrum methodologies are agile, and the Waterfall method is rigid. In today's modern software development market the agile development methods are more often used.

Two of the agile software developments main strengths are flexible response to change and good customer interaction. The latter is easier achieved when a prototype is developed in the early phases, and is later adjusted to the customer's wishes. By prototyping first, both the customer and the development team will get a clear understanding of the product's functionality and design.

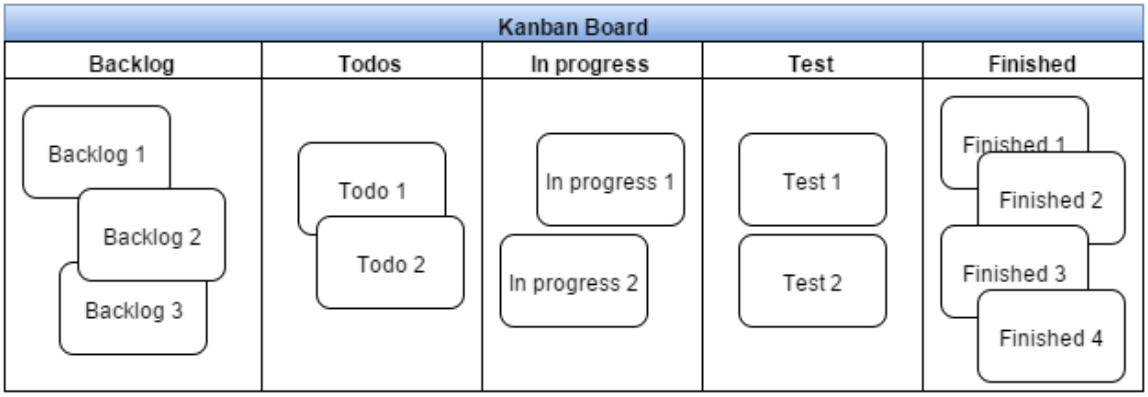


Figure 5.3: Kanban

	Pros	Cons
Kanban	- Extremely flexible - Small amount of time used in planning - Extremely simple to understand	- Often requires behavioral change - Estimation
Waterfall	- Time and effort - Budget is easy to measure - Structure	- Changes in the scope - Rigidity - Inflexibility
Agile	- Flexible - Quick response to customer - Ensures effective use of resources - Continuous improvement	- Highly skilled members is required - Planning - Can be time consuming because of flexibility

Table 5.1: Pros & cons with methodology

The product has especially high sensitivity to customer feedback as the main goals of the application needs to be ease and speed of use. One of the main motivations behind the customer request was the cumbersome systems already in place for finding correct procedures in the hospital (e.g. EQS).

The product needs to be built around expert instruction, because the information involved is very sensitive to error. Since the program deals with laws and patient treatment, the repercussions for a doctor being guided incorrectly could be severe. If this type of information should change,

our program must be changed with it, which could potentially be a risk for the waterfall type model.

The team expects to be able to deliver a few iterations of the product for refinement before the time limit expires. This lets us utilize the advantages of scrum and agile methodology more than if we only had time for the initial prototype, and thus no steps of refinement. The Kanban methodology lets us exploit flexibility in the project which is useful when documenting the process. Based on these reasons we have chosen to use the agile development methodology Scrum, with a hint of the Kanban methodology.

5.5 Summary

The Waterfall model is a contrast to both Scrum and Kanban. It is a rigid methodology which focuses on clearly defined stages that must be done in a sequential order. Scrum is an iterative and flexible methodology with certain rules, while Kanban has essentially the same flexibility, but without strictness.

6 | Risk Analysis

In every project there is a set of risks that may or may not occur throughout the project and in the planning phase these risks were identified and quantified. Table 6.1 and Table 6.2 contains each of the identified risks with associated consequences, probability, actions and strategy, deadline, and the person responsible for enforcing actions. Each consequence and probability is graded from high to low, where high represents a high severity of the consequence or a high probability. The deadline represents an optional date when the actions and strategies that are designed to mitigate the consequences of the risk should be set in motion.

6.1 SWOT analysis

A Strengths, Weaknesses, Opportunities, Threats (SWOT) analysis is a planning method which is used to evaluate strengths, weaknesses, opportunities and threats. Granting what Jackson et. al (2003) said, the SWOT analysis' intentional use was to get an overview of the internal and external factors at the beginning of the project. [23] Later, the analysis was used to map which parts of the project should be relied upon the most, and which opportunities and strengths can help the progress of the project the most, such as properties of the project team. Threats are related to external factors outside of a project, for instance the unavailability of an important tool or framework.

- Strengths: properties of the project or the team that grants an advantage over others.
- Weaknesses: properties of the project or the team that may cause a disadvantage compared to others.
- Opportunities: external elements that the project or the team may benefit from.
- Threats: external elements that could cause problems for the project or the team.

6.1.a Strengths

The two biggest strengths of the team are communication and motivation. A big factor that makes the team's communication work very well is the fact that all members of the team have Norwegian as their native language, which makes the individuals in the team able to understand the Norwegian way of behaving. The team members are comfortable with writing and speaking English, which makes it relatively easy to write reports in English, and to communicate with the team's English speaking advisor.

Figure 6.1: SWOT analysis

As the members of the team have not worked with a real customer before, the motivation of making a bigger effort in such a subject than in other subjects where the work will not be published or released, is high. Considering the probability of an escalation in the project, from one ER the every ER in Norway, the willingness to perform well is great. In addition the possibility to learn about customer relationship and gaining experience from a realistic project increases the interest for the project. Since this is a project where the groups are assigned their own offices, the professionally of the subject is higher than what the team members are used to before. This boosts the team's motivation, and enhances the ability to work efficiently and effectively.

Another strength of the team is that all team members lives 15 minutes from the team's office at NTNU, which makes the possibility to discuss problems and solutions in person painless and easy. In addition, the customer's work place, the St. Olavs Hospital, is located only 15 minutes from the team's office, which enables communication and meetings to be easily arranged.

Considering that every person in the team is attending their fourth year on their masters degree, the technical background and knowledge every person carry implies that everyone can participate in both coding and writing the report. This makes the team flexible, and even though a person may become ill, another person may always substitute for this person. Every different specialization path in NTNU's computer science masters degree, excluding Algorithms and Computers, are represented in the group, the team will have huge capability to find solutions to a broad spectre of problems.

6.1.b Weaknesses

The biggest weakness in the team is the lack of experience in considerable big projects. Although this might be the case, the team is getting highly relevant information through lectures about team work, and the entire group, individually, has been part of smaller projects where valuable techniques and knowledge can be share among the team.

Another weakness is the difference in courses each member is participating in which will elude the difficulty when planning working hours and meetings. This is a side effect of the diversity in specializations and can develop into a problem if proper planning is not made. The solution to this problem is straight forward: make a plan with the group, and be flexible where and when to work.

Since a common grade is given, a difference in people's expectation in the matter of which grade will be received may cause some people to settle for a lower grade than others. This can also lower the final quality of the project product. To conquer this risk, the team discussed which expectations every person in the group had, and made a strategy from the result.

6.1.c Opportunities

There are a lot of external elements that could benefit both the team members and the project. Firstly the excitement from the customer contact must be taken into account, because this will motivate the team, in addition the customer will advertise and promote our project to the rest of the community. Additionally, with the customer's enthusiasm the project will be give more meaning.

Since our customer contact is a doctor at St.Olavs Hospital, we can use this network to provide ourselves with the appropriate test persons. Consequently, we see this as a fulfilling opportunity to test the functionality and composition of our application. A goal of this project is that every emergency room in Norway is supposed to use the application we are developing. This gives us a potentially relatively large user group, and will possibly drastically decrease the amount of wrongly documented journals and other documents.

Up-and-coming tools and new frameworks, which qualifies for our solution, will also be appealing, since there are not many consequences for the team members other than the grade.

6.1.d Threats

A threat that might influence the implementation is the threat of changes to the requirements. Some requirements are minor and some are major. If there is a change in a requirement that may lead to huge alterations in the application, then it might become a big threat to the project. In order to prevent this, it is important to communicate frequently with the customer regarding feedback of the application.

The lack of relevant people to do testing for the application might be a threat in such a way that the final product might only be customized for a few people, and not usable for the majority of the potential end users. The importance of testing the application through user testing is high. In order to prevent the lack of relevant people to do testing, the team needs to invite the right people to test, and enough people, far ahead of the testing date.

Another threat is the aspect of unfamiliar tools which the team will, with a significant probability, encounter. Even though working with an unfamiliar framework is not a major threat by itself, it is important to take time to familiarize ourselves with a certain framework to make sure it is used correctly. As always the threat of losing data because of unnecessary mistakes made by a team member can occur. Therefore, it is important to make duplicates, and to back-up the work that is done.

6.2 Explanation of risk analysis table

This section will explain the tables 6.1 and 6.2 in the following section. The letter after the number in the first column clarifies whether the current risk is an external or internal risk. In column three the following letters have this specific meaning.

- Consequence
  - L: Minor
  - M: Significant
  - H: Critical
- (P)robability
  - L: Low
  - M: Medium
  - H: High

6.3 Summary

In this chapter a SWOT-analysis is presented as well as 17 risks related to the project. Two of the biggest strengths of the team are communication and motivation, while two of the biggest weaknesses are lack of experience, and the big workload from other subjects which demands a great amount of time. Some of the risks that are considered most crucial are; changes made to the requirement specification, lack of knowledge, failure to complete sprint task, and uneven workloads.

Nr	Risk factor	Consequences	P	Actions and strategy	Deadline	Responsible
1I	Prolonged illness/absence	M: Increased workload on the other members	M	Allocate workload on other members	48 hours	Group leader and the absent member
2I	Lack of knowledge	H: Unable to complete the product	M	Make sure the team knows or learns the tools that are needed.	Continuous	All
3I	Conflicts within the team	M: Work flow may suffer	L	Resolve conflicts as a group	Within the day of occurrence	Group leader
4I	Miscommunication within team	M: The team wastes time doing irrelevant work	M	Make sure everyone knows what their task is at all times.	Continuous	Group leader
5I	Uneven workloads	M: Member may not be able to complete all of his/her work	H	Try to divide the remaining tasks and use SCRUM-methods to better estimate workload for the next sprint	Within the same sprint	Group leader and SCRUM master
6I	Lack of motivation	M: A team member's potential workload and quality of work is not met	L	Regularly check on each team member	Continuous	Group leader
7I	Unable to contact members involved	M: Organizational problems	L	Keep on working without the missing member	Continuous	Group leader
8I	Poor project estimation	H: Not meeting project deadline	L	Increase workload or change requirements	Continuous	QA responsible
9I	Failure to interpret the assignment	H: Redundant work may be done. Fail to meet customer needs	L	Maintain regular contact with customer and evaluate major design ideas with the entire group	Continuous	All
10I	Poor choice of framework	H: Product not completed or not as specified by customer	L	Do a good research on available tools at the start of the project.	Continuous	All
11I	Poor leadership	M: The team's productivity and motivation may suffer.	L	Second in command takes a more leading role	Continuous	All

Table 6.1: Internal risk analysis

Nr	Risk factor	Consequences	P	Actions and strategy	Deadline	Responsible
1E	Lack of access to workplace	M: No place for meetings and working in group. May hurt communication and work flow	L	Have an alternative place to meet	Continuous	Group leader
2E	Conflicts with the customer	M: Feed	L	Resolve conflicts as a group	Within the day of occurrence	Group leader
3E	Loss of data	H: Important data may be lost and a lot of work will have to be redone	L	Keep data stored in cloud based services and maintain a backup of the most important data	Continuous	All
4E	Changes and/or additions to the specifications	M: May alter the implementation	H	Reevaluate backlog order and schedule	48 hours	System architects
5E	Hardware and Software failure	H: Need to acquire new equipment or software, which can blow the budget	L	Make use of external resources	48 hours	All
6E	Software version updates	L: May slow down the time it takes to use a specific software	H	Spend time to understand new functionality or syntax	Continuous	System architects and programmers

Table 6.2: External risk analysis

7 | Technologies

Because a knowledge support tool is heavily dependent on content there are no pre-existing solutions that provide what we need. However, the need to navigate through information easily and display it in a clear, concise manner is a very common challenge in the app industry, so we can borrow heavily from standards, templates and frameworks that have been developed to aid app developers.

7.1 Programming tools and frameworks

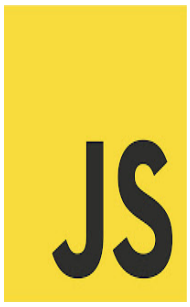
This section describes the different programming tools and frameworks that are used in the project.

7.1.a Cordova

Apache Cordova is a mobile application development framework created by Nitoby. After Adobe systems purchased Nitobi in 2011 they rebranded this framework as PhoneGap, and later released an open source version called Apache Cordova [45]. This open source version supplies a set of APIs [43] that lets a developer access native functionality on a mobile device while using HTML5, CSS and JavaScript, enabling hybrid development. Because it functions seamlessly with many of the frameworks and libraries for app development and has stable releases it is an easy choice.

7.1.b JavaScript

JavaScript was originally developed in 10 days in May 1995 by Brendan Eich, while he was working for Netscape Communications Corporation. [42] As a dynamic high level language, JavaScript which is often combined with HTML and CSS in order to be the essence of World Wide Web content production; JavaScript is used to add dynamic elements to a web page, HTML is used to structure the content, and CSS styles it. JavaScript is an object oriented language where objects are created by adding methods and properties to them at run time. This programming language can be used to add behavior such as pop-ups, give feedback to the user, and give suggested results while typing in a search box.



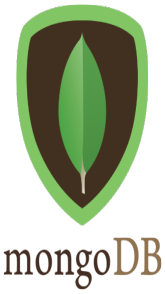
7.1.c HTML5

HTML stands for Hypertext Markup Language, which is used for structuring and presenting content on a web page. HTML5 is the fifth version of HTML, published in 2014 [6]. This language has a wide variety of functions that works on all type of platforms. It can be used to show animation, music, movies and other applications.

7.1.d CSS

CSS stands for Cascading Style Sheets [5]. This style sheet language is used for defining the visual presentation of a web page. CSS allows the designer to specify fonts, colors, and layouts. The style sheet can be integrated into a web page or it can act independently as its own file. This file can change the styling of all the pages that are linked to the css file, which makes the manipulation of the web pages a lot easier by only having to change one single file to change many pages.

7.1.e MongoDB



MongoDB was first created in 2008 by a company called MongoDB Inc. [44] In contrast to a relational database, MongoDB is a NoSQL database [10]. MongoDB is document-oriented which means that it consists of mapping between documents by using dynamic schemas. These documents contain fields, binary data and sub documents. In MongoDB, the data is stored as objects, which makes the database especially suited to be used along with an object oriented programming language.

7.1.f Front-end: Ionic

For front-end development there are a variety of framework options for hybrid development. The team has tested basic application implementations and read documentation for several of the Cordova friendly frameworks, and favour Ionic for its ease of use, stability and wide range of features.

Ionic is a development framework for building interactive hybrid mobile apps using HTML5 and JavaScript [7]. Ionic is a front-end user interface framework that handles the look and feel of an application. It comes with built-in native-style mobile elements and layouts, which makes it easy to style the application for a native look.

7.1.g Back-end: Meteor

For the back end of our application, we need only basic functionality to update and reference law texts and potentially an admin login. However, we need to be able to update this information from the cloud, and run the application offline without losing all functionality.

Meteor is a JavaScript app development platform which has a full-stack framework for mobile and web development [14]. The platform supplies a simple and sturdy database through MongoDB use, a collection which is easy to update from an online server. This functionality is extended to manage offline situations through GroundDB, which is MongoDB with the necessary offline functionality added [16]. The Meteor platform has a large database of possible additions and extensions which gives us few constraints relating to the implementation of our design and possible features. Templates and implemented UI elements can also be used from the database, which lowers development time.

7.1.h The chosen solution: Meteor with Meteoric

The natural way to combine Meteor and Ionic is with Meteoric which integrates Ionic into Meteor [47]. This solution retains all Meteor functionality, while adding Ionic without the overhead of integrating the two manually. All additions and extensions available to Meteor are expected to work with Meteoric with or without seamless integration, so the low development time and large amount of freedom is retained.

7.2 Management tools

There are many types of tools that can be used for managing a project. In order for the project manager to have control and overview of the work that needs to be done, it is important to be

organized. This sections presents the management tools the group has chosen to utilize throughout the implementation.

7.2.a Google drive

Google Drive is a cloud based service made by Google. [37] This service allows the user to synchronize, upload, and share files with other users. In addition Google Drive includes an office suit that permits collaborative editing of documents, spreadsheets, presentations, drawings, forms, and other tools. Google Drive was used when writing the meeting agendas, status reports, general text to the report, making models used in the report, and documenting hours used in spreadsheets.

7.2.b Trello

Trello is a web-based project management application originally made by Fog Creek Software. [20] When using Trello one organizes projects into boards. These boards are then divided into categories or list which contains cards. The cards are tasks that are suppose to move from one list to another with drag-and-drop to see progress in the project. Trello has the possibility to tell you who's working on what, what's being worked on, and where in the process something is. Further, Trello is a collaboration tool where everyone can edit the different parts, which makes this management tool suitable for projects like this one.

7.2.c Slack

Slack is a team collaboration tool co-founded by Stewart Butterfield, Eric Costello, Cal Henderson, and Serguei Mourachov. [26] It divides communication into channels, direct messages and private groups. This way the team can customize the platform for communication and maintain an overview of progress across individual team tasks. Slack provides a series of integrations and acts as an information portal. The team uses the GitHub integration, which posts activity form the project repositories, (Git and GitHub is explained in section 7.4.a) and the Trello integration, to post activity from the teams Trello board.

7.2.d Teamweek

Teamweek is a management tool used for planning a project's life cycle [41]. The Teamweek calender is a Gantt chart and includes milestones, sprint lengths, minor and major tasks, and corresponding dates. It uses color coding to categories the different tasks, which makes it easier to group the various parts. This tool was used only as a internal reminder of how long a specific sprint, or what the estimation of a task were.

7.2.e GanttProject

GanttProject is a tool used to create gantt diagrams, and is created by Dmitry Barashev. [40] This technology allows the user to create tasks and milestones with priorities, cost, color, and fill pattern. Hierarchical trees can be created, and the different levels can show a summarized cost of the lower levels.

7.3 Design tools

When working with a customer which highly appreciate easy and clean design, it is important that these parts will be taken into consideration. This section will involve the tools used when designing for instance logos and icons. In addition, the tool used for making the prototype will also be presented.

7.3.a Axure RP

Axure was founded in May 2002 by Victor Hsu and Martin Smith, and is a prototyping software that can swiftly make prototypes. [36] The application generates HTML web sites that contains the form and wireframe elements that are set up by the user. The interface is drag-and-drop based, and is also extended with wireframing.

7.3.b Photoshop

Photoshop is a photo manipulation tool created by Adobe [21]. This tool has allowed our team to create icons and logos used in the application as well as the website for our company.

7.4 Extra tools

This section consists of tools and technologies which did not fit any of the other sections.

7.4.a Git

Git is version control system mainly used for software development [21]. Speed, data integrity, and support for distributed, non-linear work flows, is the significant parts of this distributed revision control system. Every Git working directory is a repository, which is completely developed, is independent from network access, as well as includes a complete history of respective directory. It also has full version-tracking capabilities. GitHub is a web interface to manage repositories in Git.

7.4.b LATEX

LaTeX is a word processor an will be primarily used for writing the report and creating relevant tables [15]. As opposed to the typical word processor Microsoft Word LaTeX does not use formatted text. It uses plain text and relies on markup tagging conventions. This defines a general structure of the document. To manufacture a file which is fit for printing or to be digitally displayed, a Tex distribution tool is used, for instance TeXlive or MikTex.

7.5 Summary

In this chapter we have presented the tools that are used in the project, as well as presenting the main decision of the framework that has been chosen; Meteor with Meteoric. Meteoric integrates Ionic into Meteor, which enables us to access the functionality of Meteor along with the benefits of Ionic.

8 System Design

This chapter will present the architecture and the design of the system. The architectural drivers are listed and intended patterns are described. The architecture is based on the architectural 4+1 model to display a definite analysis of the system. The model is divided into five different views, logical, process, development, physical, and scenarios, where each view represents various viewpoints in the architecture.

8.1 Architectural drivers

Software quality attributes

Usability is the most important quality attribute of this application because the technological knowledge among intended the user group is relatively low. To achieve this the team will use design principles and test the application several times on medical personnel to get constructive feedback.

Business constraints

The final solution should be delivered on a set deadline date for evaluation and grading. After this date the customer owns the product.

Technical constraints

As demanded by the customer the application should be available on a majority of mobile devices and browsers

8.2 Architectural pattern

This section contains the architectural patterns the team planned to use in this application.

8.2.a Model-View-Controller pattern

The MVC pattern [3] breaks functionality into three components: a model, a view, and a controller that mediates between the model and the view. The model is a representation of the data storage and it contains the application logic. The view is the components the user sees interacts with.

8.2.b Client-Server pattern

A client-server pattern [4] is basically requests from a client to a server and responses from the server back to the client. In this architecture the team have decided to implement the controllers partly on the client-side to achieve offline possibility. The controllers regarding authentication and validation will be placed on the server to prevent attacks.

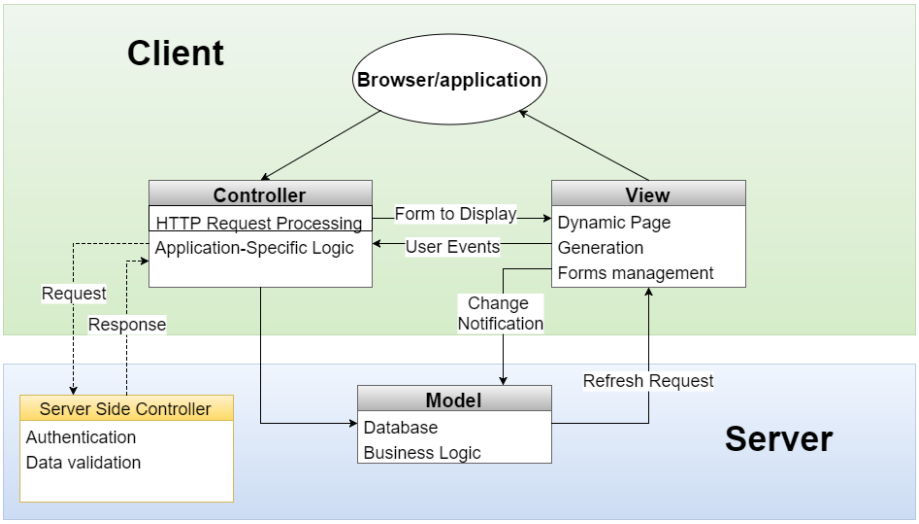


Figure 8.1: MVC with client-server pattern

8.3 4+1 Views

The 4+1 model is a description of the architecture in software-intensive systems, where multiple, concurrent views are used. Using these concurrent views, which are designed using an architecture-centered, scenario-driven, iterative development process [25], allows the possibility to devote attention to the different concerns of every person involved with the architecture. In addition to the various concerns the views separately manages the functional and non functional requirements.

A software architecture is a high-level structure of the design and implementation of the software which is being developed. The authors of 'Foundations for the Study of Software Architecture' [32], Perry and Wolf, made a formula which alleviates the performance requirements and considerable functionality of the system, in addition to satisfy some non-functional requirements such as reliability, scalability, portability, availability, modifiability, and usability. The formula is as follows: Software architecture = Elements, Forms, Rationale, 'That is, a software architecture is a set of architectural (or, if you will, design) elements that have a particular form.' [32]

The formula is applied independently on every view, meaning in every view there will be defined a collection of elements to use, forms and patterns which is used is captured, and a connection between the architecture and the requirements is obtained through the rationale and constraints.

To illustrate the the differences in the views a blueprint with a specific notation for each view is used. The 4+1 view model is rather flexible due to the fact that other tools, notations, and design methods can be utilized. This is primarily useful for the process and the logic decompositions.

The physical view is omitted as we do not use any special hardware for the application. Web and browser details are also omitted, as they are not the focus of our implementation.

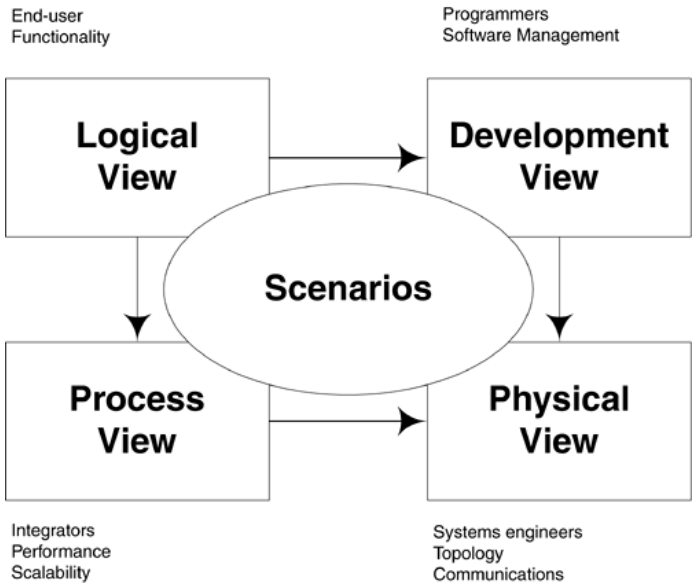


Figure 8.2: 4+1 View
[25]

8.3.a Logical view

The logical view depicts which services the system provides to its end-users. It provides a static overview of the entire system and focuses on the functionality and the interrelationship between parts of the system. This view gives an overview of the functional requirements that the system should fulfill.
The diagram 8.3 is a modified version of a class diagram, attempting to give a clear representation of how information and functionality is supplied to the common user and administrators.
8.4 shows all the views the common user will see in the application and how they are structured. The structure of the views is made to be familiar for mobile users, while addressing customer feedback.
8.5 shows all views available to an administrator of the application. The structure closely follows that displayed to the common user for ease of use. This is especially important because the administrator is not expected to have any additional experience or education in software development or the application in general compared to the common end user.

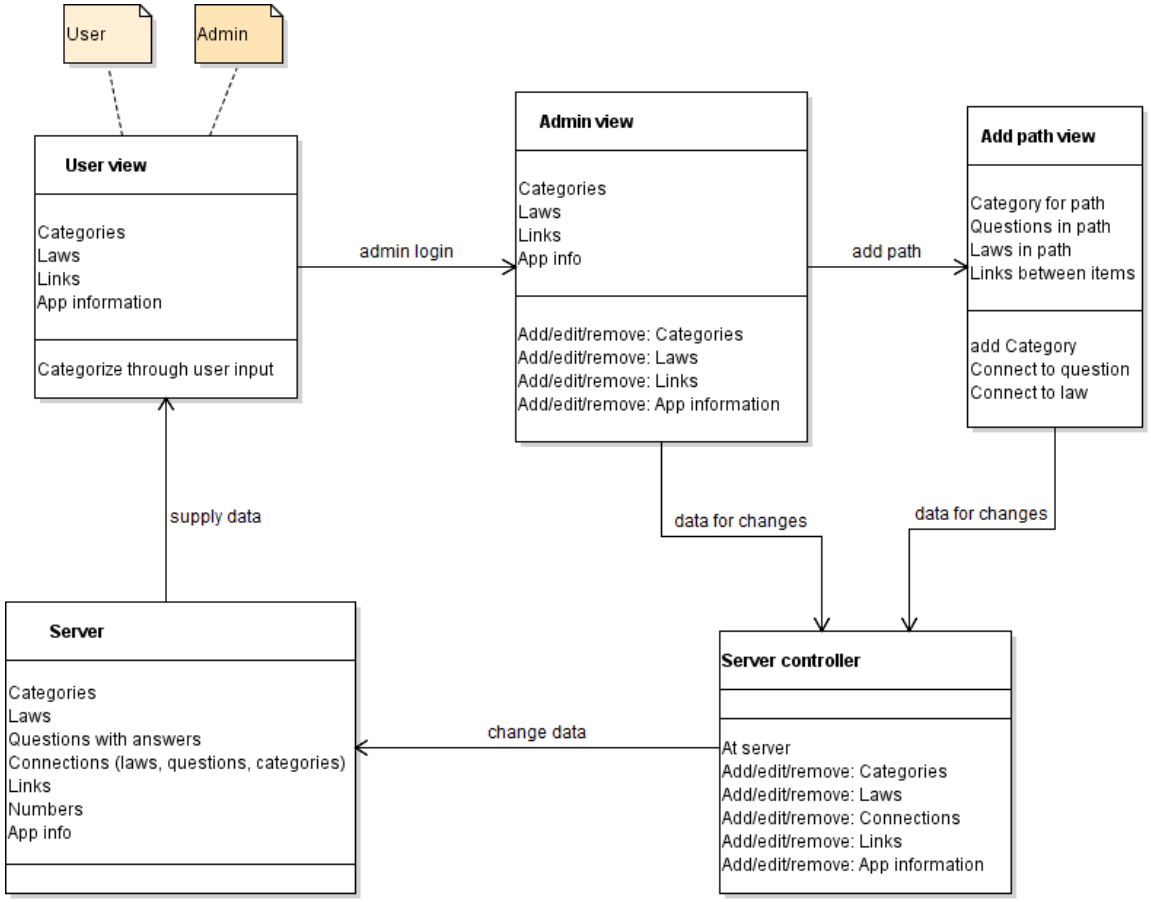


Figure 8.3: Logical view of the application

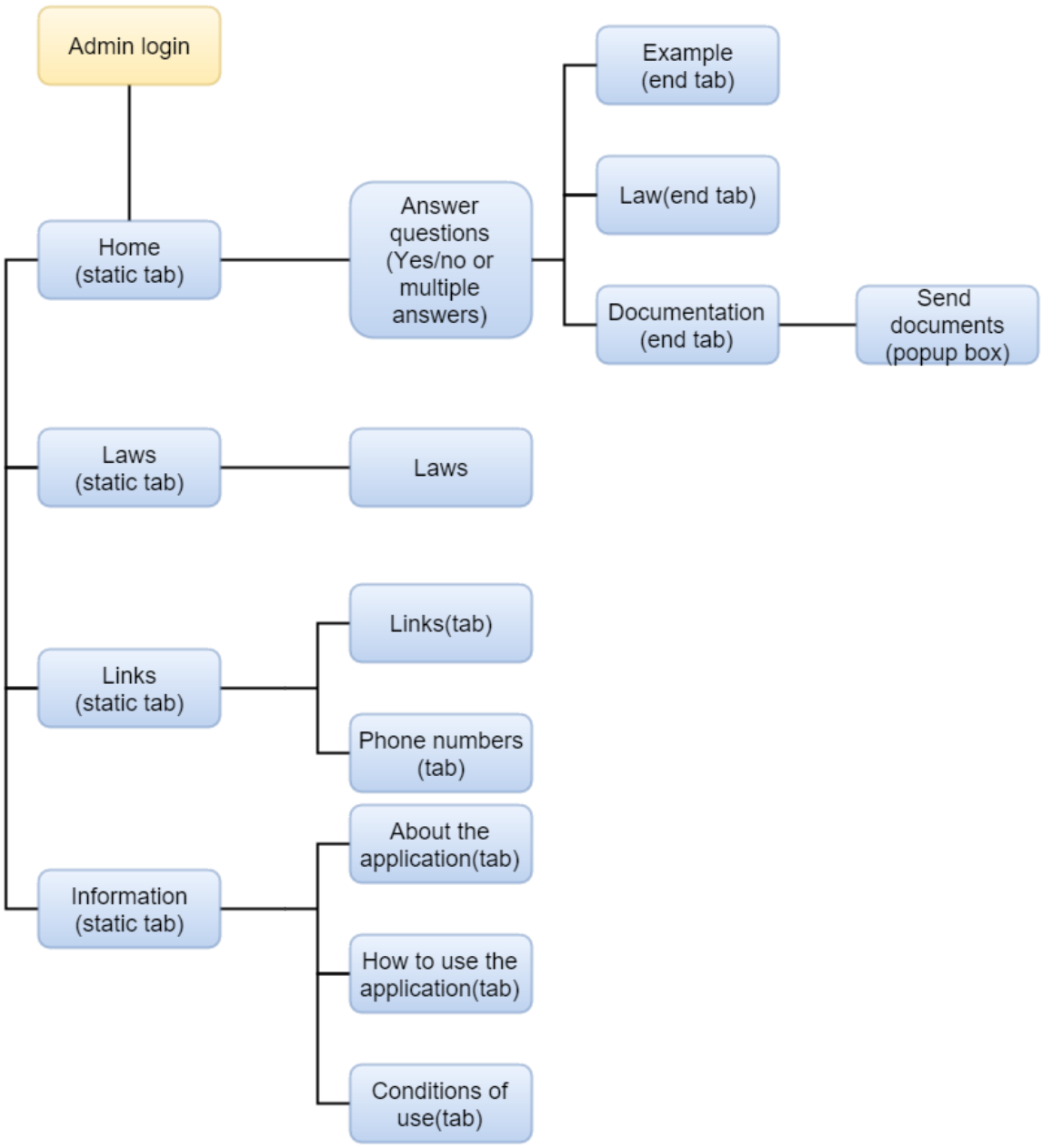


Figure 8.4: Logical view of the application

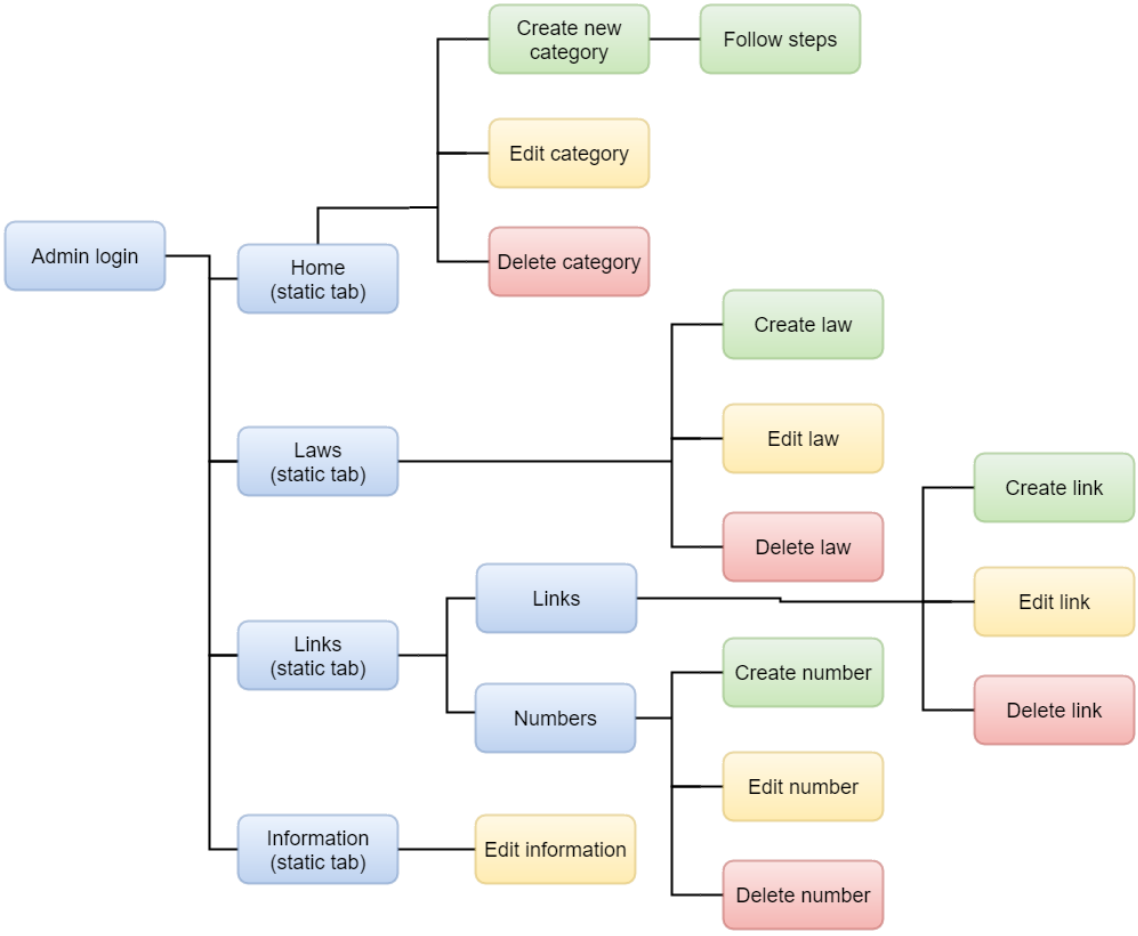


Figure 8.5: Logical view of the application

8.3.b Process view

The purpose of the process view is to be able to show how the system works at run time. It displays the way the dynamic aspects of the system function, and how the system processes communicate. The diagram depicted in Figure 8.6 displays the logic used for the addPath page in the application. All Yes/No questions can be followed by an indefinite number of Yes/No questions, an indefinite number of multiple choice questions, or a law which is the end of any path. The application does not support connecting from multiple choice questions to Yes/No questions.
Similarly, Figure 8.7 shows the logic used for user navigation of the front page. As above, Yes/No

can lead to any number of Yes/No questions or any number of multiple choice questions, or a law. Figure 8.8 shows how user input from an administrator is handled for database changes. The security for our server is simple. The only way the server can be edited from a client is through calling server methods, which are only called if the user is signed in. Encrypted connections are assumed, otherwise the information can obviously be accessed.
The sequence diagram Figure 8.9 shows the calls for a typical db change in the application. This diagram assumes no error in the database calls, as errors are sent directly to Meteor alerts and displayed to the user if applicable, as is typical for the Meteor framework.

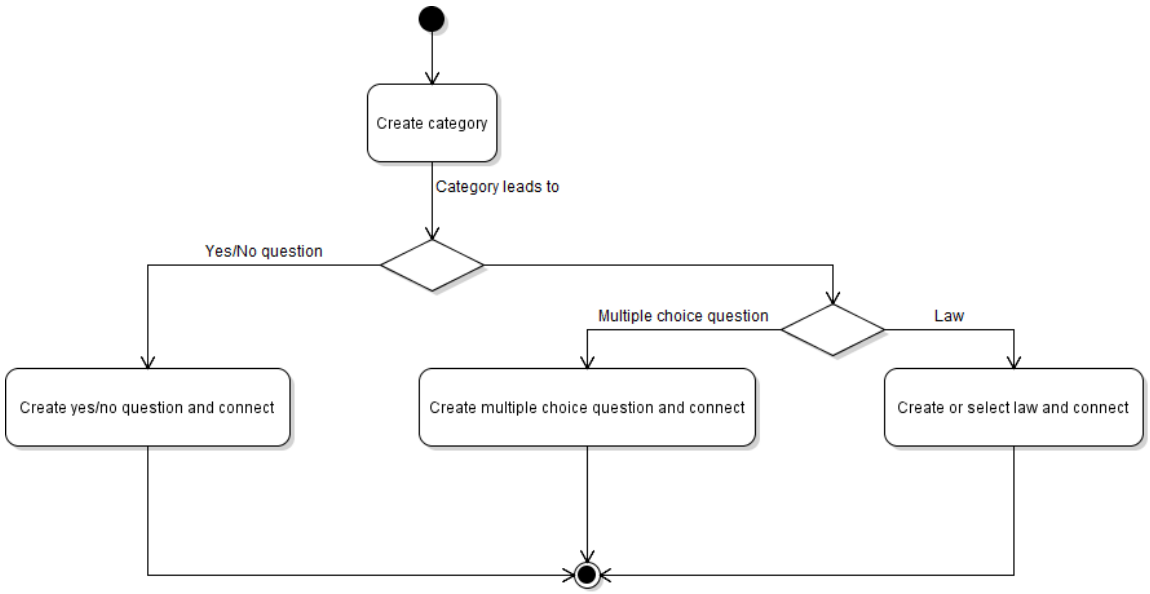


Figure 8.6: Activity diagram for adding a path

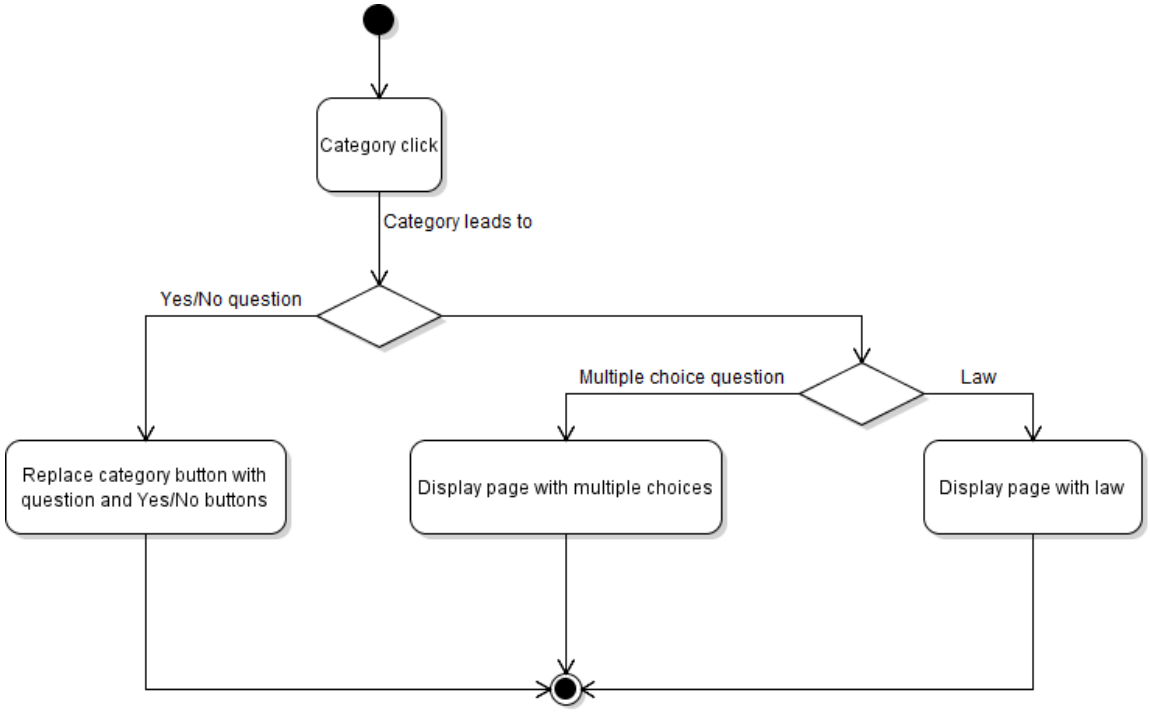


Figure 8.7: Activity diagram for category buttons

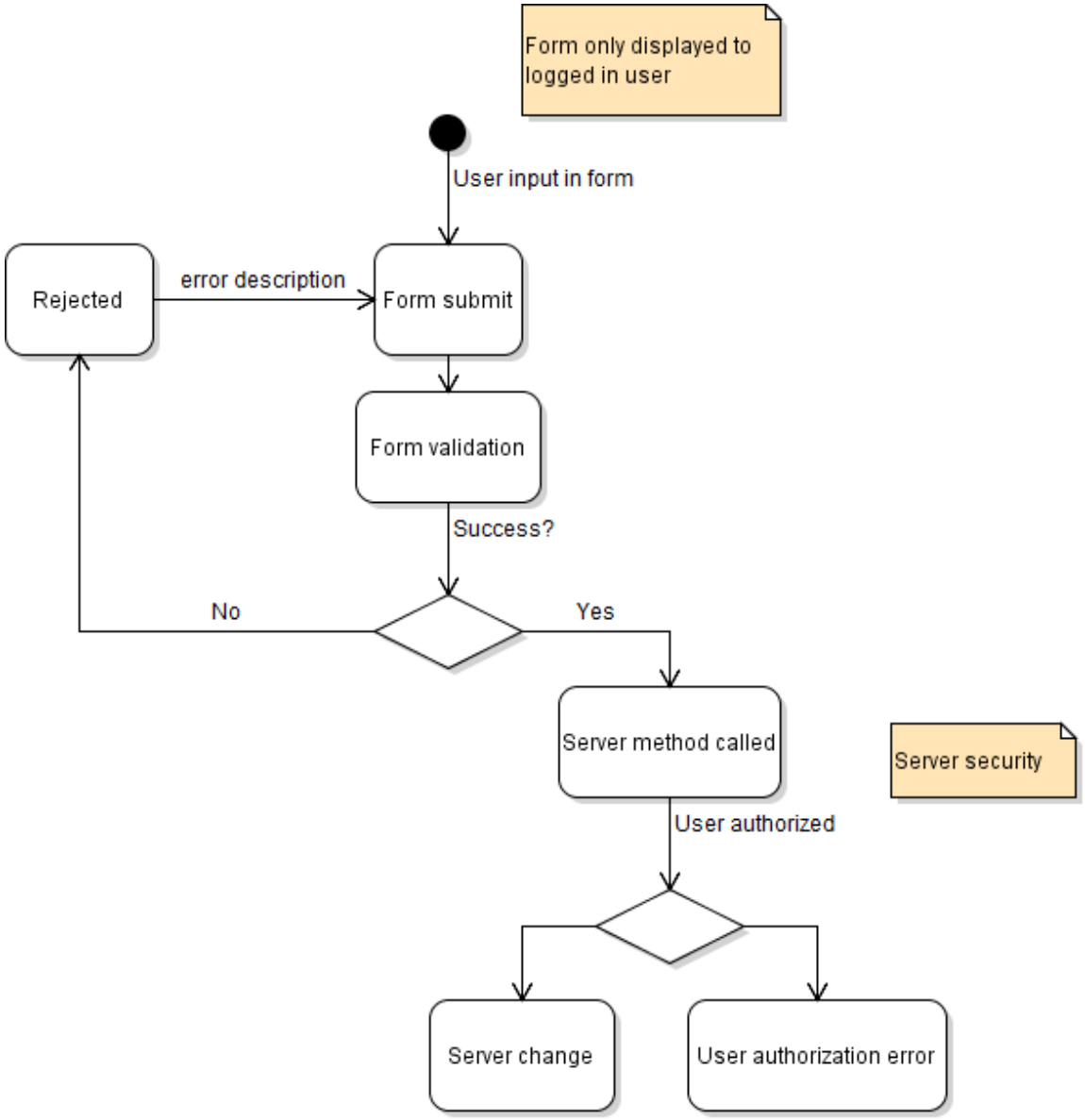


Figure 8.8: Activity diagram for editing database

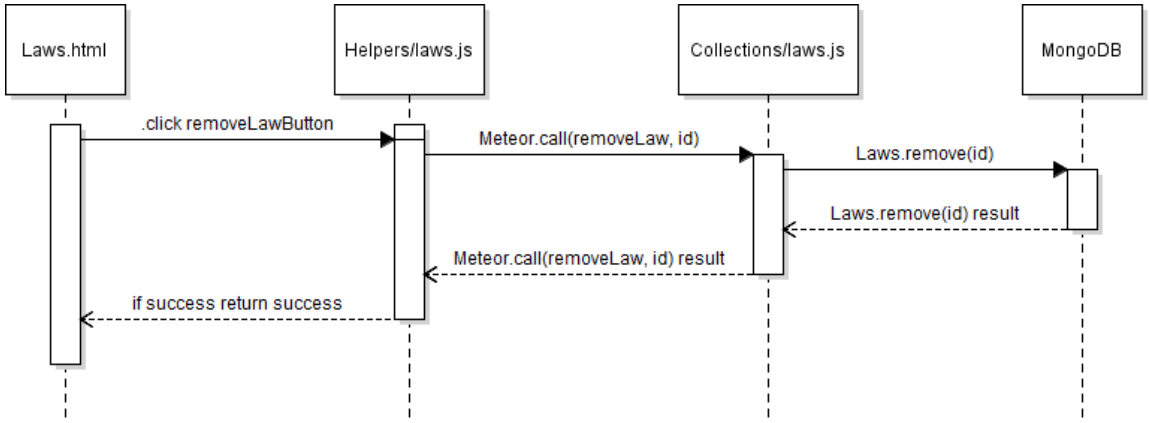


Figure 8.9: Sequence diagram for a database law removal

8.3.c Development view

The development view is used in software management, and focuses on dividing the system into smaller system components, modules and sub-systems. The view is also known as the implementation view, as it shows the structure of the implementation of the system.
Figure 8.10 gives an overview of the structure of our application, which is the typical Meteor structure. Meteor uses HTML, CSS and JS exclusively, even server-side.

SCSS is a CSS extension that is responsible for all styling of text and UI components in our application. The structuring of information in each view of the application is done in HTML and JS through the use of Spacebars. In Spacebars, HTML templates are used which have associated JS helpers that serve as the controller and supply information from the model for dynamic portions of the page. The HTML templates are reactive, they update based on changes in the helpers associated with each template. Separating content into different views is done through templating and a JS router. The router displays templates and sets the data context for different parts of the application, either accessed through URL redirection or simply swapping templates on the current page when a function is called, for instance on a button click. The server is accessed through the JS template helpers that request information which the client has been allowed access to, or through calling secured server side methods to make changes in the database. Meteor uses MongoDB by default, which is a document-oriented database where we store all the information we wish to be able to manage. The information is info to be displayed to the user that can get changed. Paths and laws are stored here. Setting up and adding items to the MongoDB is done through server-side JS code, which contains methods for creating and changing collections of documents in the database.

Figure 8.11 shows the file structure for the application. Code under lib is shared between client and server, and contains code for the routing between views in the application and the user account setup. The user account setup is shared because the framework requires this to set up server side connections to the database of users and apply configuration options to the client. Server code under server is server bootstrapping, where initial database information is added, a server side

method for sending emails, and publications which let information in the database be accessed by clients. Docs contains relevant document and form files for the patient categories in the application. Client contains the client side code for the application. That is, styling of UI components, HTML templates, helpers for each template, and global methods and variables.

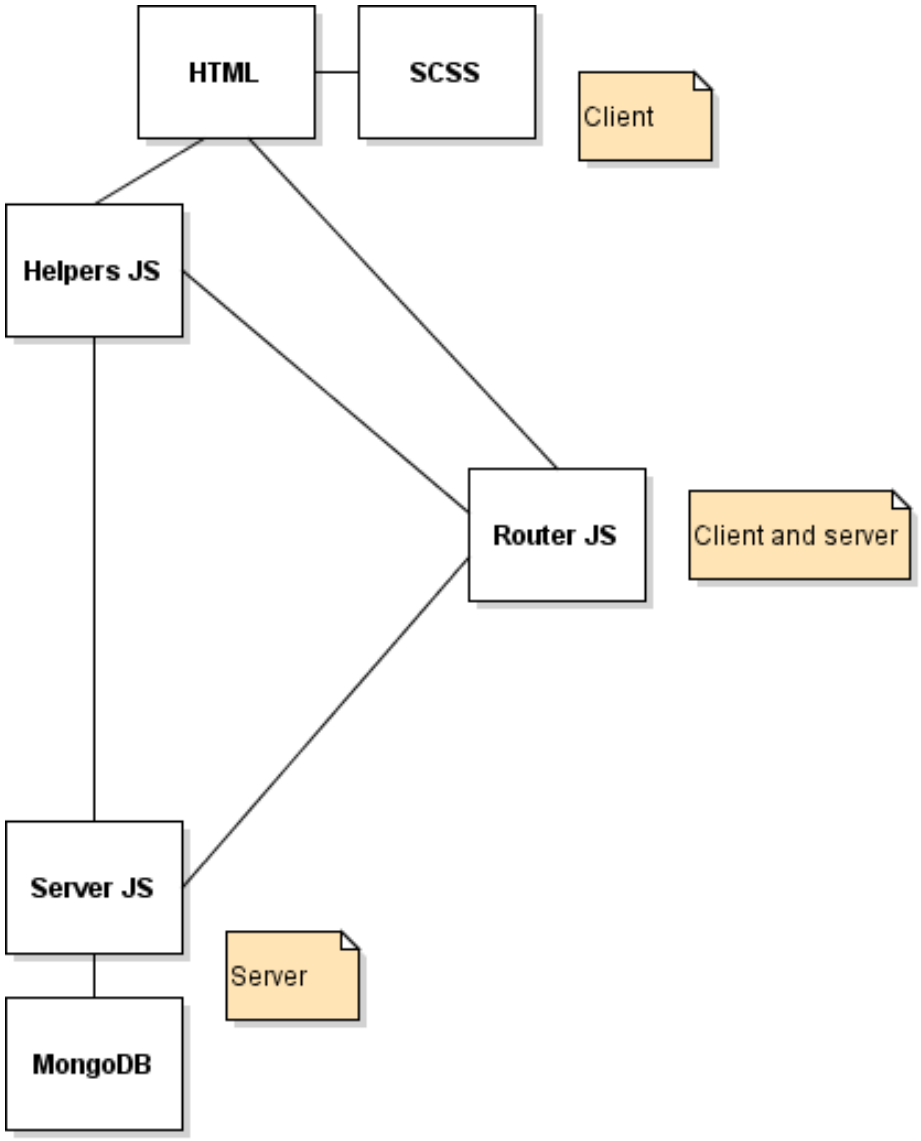


Figure 8.10: Overview of implementation, the typical Meteor setup

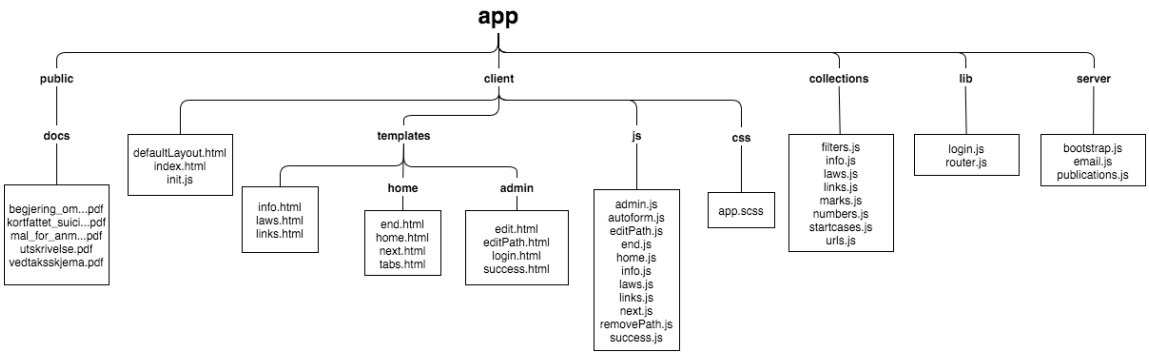


Figure 8.11: The file structure of the application

8.3.d Scenario view

The scenario view is an abstraction of the most important requirements, where one uses either scenarios or use-cases to show them. In this project the team has chosen to utilize use-cases to represent the system's requirements, as shown in section 4.5.

8.4 Design principles

The team has studied and applied several design principles to follow best practices for user interactions with digital systems. Some of them are explained in this section.

Gestalt principles

The gestalt principles [39] stems from perception-psychological explanations of how we perceive and organize visual impression. There are six principles:

- Grouping / Proximity
- Line / Continuity
- Mental completion
- Similarity in shape
- Similarity in color
- Foreground / Background

Don Norman

Don Norman published a book in 1988 called "The design of Everyday Things" [38]. Here he introduces four concepts about design, which are:

- Affordance
- Constraints
- Deedback
- Mapping

Affordance relates to what action a subject signals. Like a button wants to be pushed, and might stand out. **Constraints** is about clearly indicating actions that are not allowed. If a button should not be pushed it might be grayed out. **Feedback** means that actions should indicate that they execute, like a button changes color as it is pushed. **Mapping** is concerned with how the user interprets objects and actions from other applications. That the "back" button on the application have a left arrow maps to the users knowledge of other uses of a left arrow.

Nielsens 10 heuristics

Jakob Nielsen studied several systems and published in 1995 10 general principles for interaction design [29]. They are called "heuristics" because they are broad rules of thumb and not specific usability guidelines. The heuristics are:

- Visibility of system status
- Match between system and the real world
- User control and freedom
- Consistency and standards
- Error prevention
- Recognition rather than recall
- Flexibility and efficiency of use
- Aesthetic and minimalist design
- Help users recognize, diagnose, and recover from errors
- Help and documentation

8.5 User Interface

The resulting interface can be seen in appendix J. This is a user guide for the application and includes screen shots for every page of the application along with a textual description og the components.

8.6 Architectural rationale

The choice of architectural structure for our application is closely tied to Meteor and web development best practices. MVC is a natural, integrated and almost necessary pattern to use with Meteor. It also enables the responsiveness and reactivity we require to make a user friendly application. Reasons for choosing Meteor are documented in section 7.1. Database access is also done in typical Meteor and web app fashion, calling server side methods for access. We chose to use the Meteor default MongoDB because it has good enough performance for our application, has short implementation time when used with Meteor, and because we managed to structure the information we needed with the document-oriented database pattern. The capacity of the system depends entirely on the servers it is deployed on, but the information in the database is very limited in size, so we have had no database performance issues and do not expect there to be any. We are not required to supply a server for deployment, which is the true performance bottleneck of the back-end.

8.7 Summary

This chapter describes how usability is the most important quality attribute. The other architectural drivers are the deadline as a business constraint and the technical constraint is that the application should be available on a majority of mobile devices and browsers. It describes how the group intend to apply the MVC and client-server pattern, and shows views of the architecture from the 4+1 model. It lists some design principal the team has studied and concludes with an architectural rationale.

9 | Sprint 1

In this chapter the first sprint will be explained in more detail. The first sprint has focus on the initial implementation and which functionality the application should have. This will be clarified in the prototyping, and the implementation section. In addition to information regarding the actual application, a detailed description of the testing, as well as the results, will be presented. Finally there will be an analysis of the sprint, where the team consider what went well and what could be improved in the next sprint.

9.1 Sprint Planning

Based on the initial planning phase that was performed before the start of sprint 1 the team had created a sprint backlog containing the tasks the team had decided to complete within the first sprint. Since most of the analysis was completed before the start of the sprint, the team could start creating content based on what was found in the planning phase. The main goals of the sprint were to design, implement, and test the initial prototype, as well as set up and test the chosen programming environment. The deadline for designing and implementing the prototype was set on the 21.09.15 since the group was going to test it on the customer on the 22.09.15. This was the only milestone for this sprint.

9.1.a Sprint Duration

The total duration of the first sprint lasted two weeks. The first sprint began on the 14th of September, 2015, and ended on the 27th of September, 2015.

9.1.b Sprint Goal

The initial goal of the first sprint was to make and finish a prototype to show to the customer. In order to do this, some planning was done by the team and there were discussions about how we wanted the application's flow to be like. After some consideration in the first couple of days of the sprint, the team decided to make several prototypes to cover more options for the flow of the application. The readjusted goal became to make and finish three distinctive prototypes.

9.2 Sprint Backlog

This section contains a list of the sprint backlog items that were either initiated or completed in the first sprint. The tasks were picked out of the backlog based on what needed to be completed first. The visualization of the burndown of the tasks in the backlog is displayed in Figure 9.4.

No.	Description	Priority	Estimated Time	Used Time
01	Documentation	high	100	145
02	Sketch prototype #1	high	6	2
03	Create prototype #1.1 in Axure	high	10	9
04	Create prototype #1.2 in Axure	high	10	14
05	Create prototype #1.3 in Axure	high	10	12.5
06	Setup programming environment	high	6	2
07	General tools setup	medium	6	3
08	Test meteoric	low	10	12.5
09	Test axure prototypes	medium	14	8
Total			**196**	**208**

Table 9.1: Backlog for sprint 1

9.3 Prototyping

The initial prototype that was developed for the customer was based of the flowchart presented in the planning phase, more specifically in section 4.1. As discussed in section 4.1 the group had different interpretations of the flowchart and in the end the team was not able to settle on an interpretation. This in turn led to the team developing three different prototypes for the customer. The motivation behind this was to present different types of implementations and get feedback on the pros and cons of the different implementations. This feedback would then serve as the basis for the next prototype iteration.

9.3.a Prototype 1.1

Prototype 1 was made as a naive interpretation of the flowchart where a transition (arrow) between content boxes indicated a change of view in the application. That is, each content box serves as its own view and the user is only able to make transitions to views, or boxes, that are directly linked to itself. In figure 9.1 the leftmost screen displays the start view where the four main categories from the flowchart are presented. The middle and rightmost screens represents the law and summary views. These views are directly linked and the user is able to navigate from law to summary and vice versa. From the law view the user is also able to navigate back to the example view, which is the view that lies between the start and the law view. The summary view is the last view in the flowchart and is only linked to the law view and can only maneuver to this view. All the views, except the start screen, also contains a home button that takes the user all the way back to the start view. As this is a very naive interpretation of the flowchart it has several elements that the group found to be problematic. The main one being that it is deeply nested and the movement within the application is quite static.

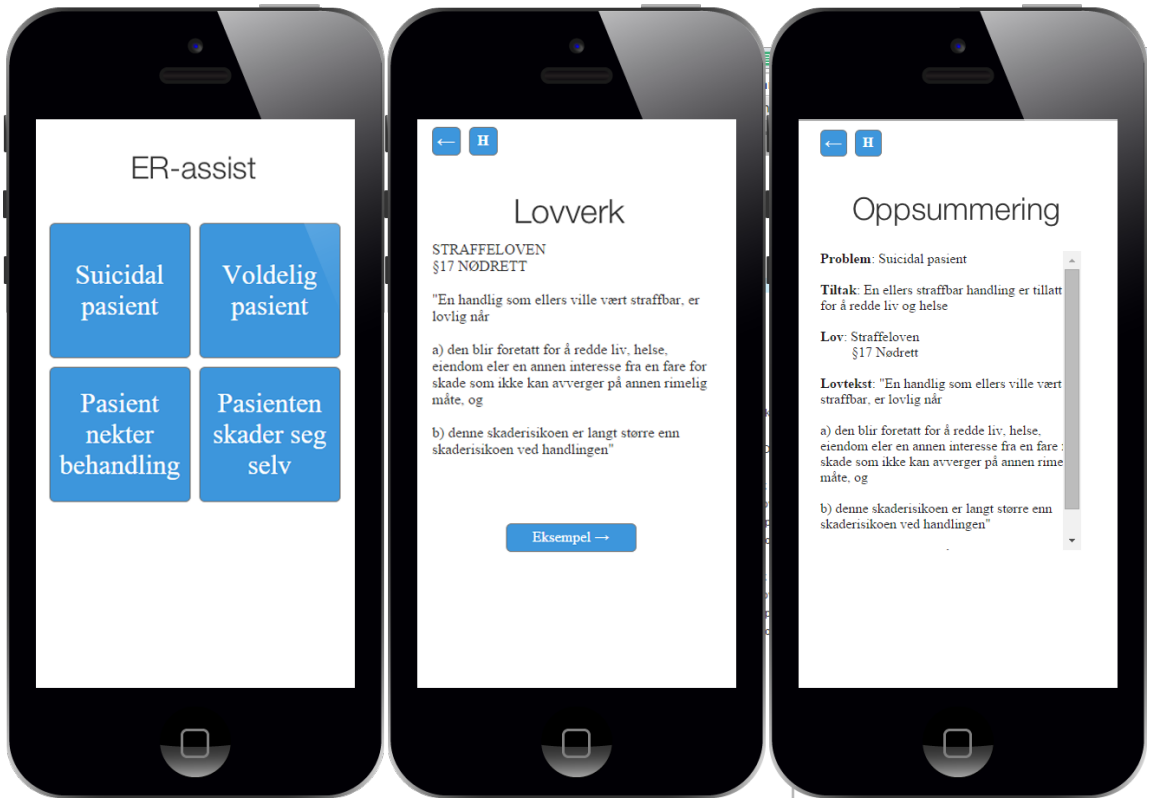


Figure 9.1: Prototype 1.1

9.3.b Prototype 1.2

Prototype 2 was also developed in a naive way, quite similar to prototype 1. The main difference was the interpretation of the category where the path in the flowchart was determined by a set of questions. Instead of having one view for each question with associated answers, the button for this category was transformed into a dynamic button that, after being pressed initially, prompted the user with the questions and answers so the user would not have to leave the start view in order to access the questions. The dynamic button is displayed on the leftmost screen in figure 9.2.

The dynamic buttons were also used on the categories that directly continues to the example view. For these categories the drop down of the dynamic buttons allowed the user to choose if he or she wanted to jump directly to the example or law view. The motivation behind using this kind of buttons was to reduce the amount of clicks, and thus the time, it took to maneuver to the law view, which was considered to be the most used view.

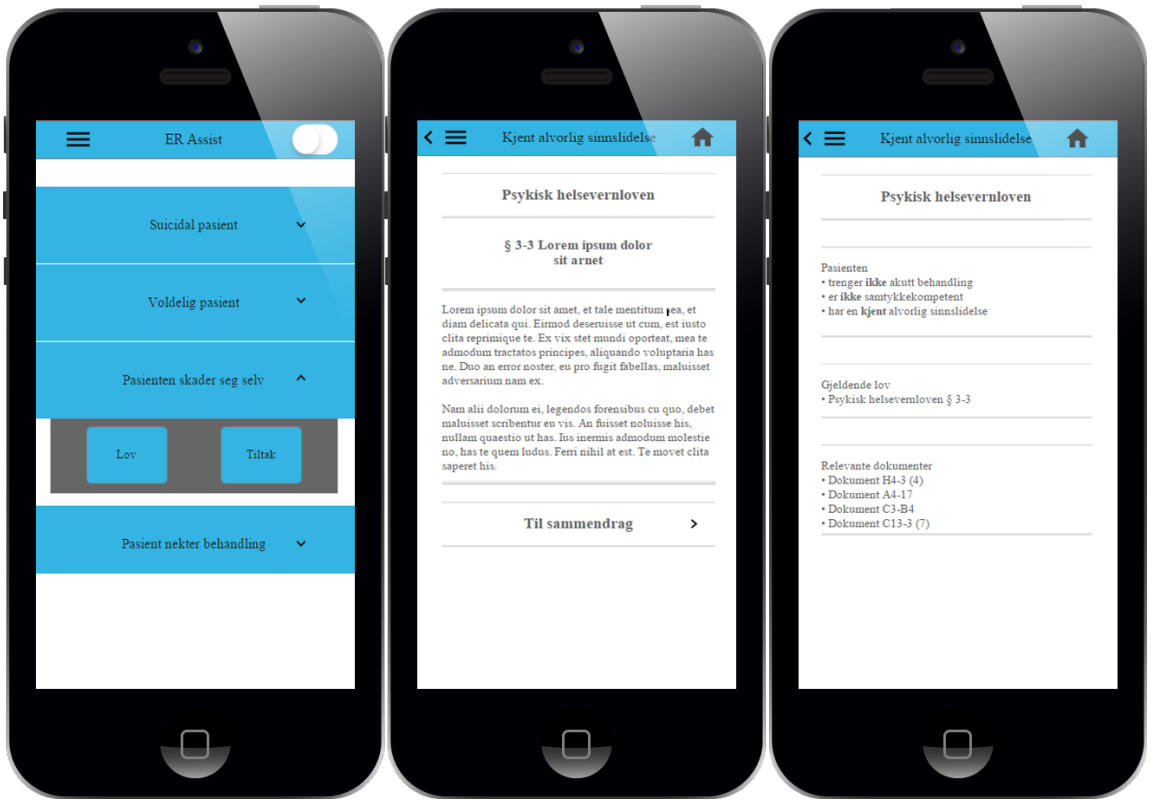


Figure 9.2: Prototype 1.2

There were also some other minor alterations compared to prototype 1, like the side menu and the layout of the views, but this was not the functionality that the group wanted to test in the first user testing so these alterations was not emphasized in the user testing and the customer was informed about this during the test.

Prototype 2 also had some issues when it came to the flow and maneuverability of between the views, and the dynamic buttons for the categories without questions could potentially be more confusing than helpful. If the user chose to go directly to the law view, and then wanted to maneuver back to the front view, he or she might want to press the back button only to find out that it only lead them to the example view. It was considered to alter the path when the user chose to go to the law view first, but this would only lead to another case of inconsistent design.

9.3.c Prototype 1.3

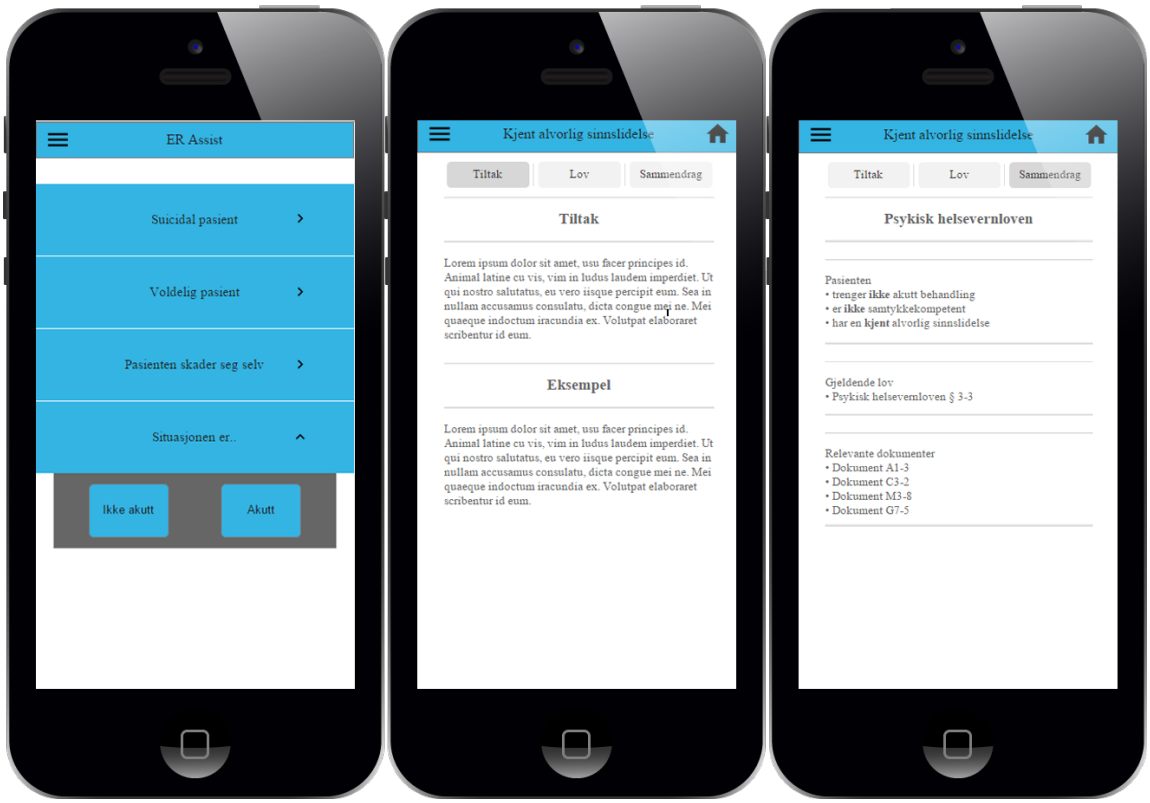


Figure 9.3: Prototype 1.3

Prototype 3 was originally just an improved version of prototype 2. The dynamic buttons for the categories that did not lead to any questions were changed back to basic buttons, and they now lead the user to a tab view where each of the three original views, example, law, and summary, now

where tabs in one view, as displayed in the middle and leftmost screens of figure 9.3. Using tabs instead of separate views provides the user with more information about his or hers location and what options they have in order to move around inside application. The user can now freely move between examples, law, and summary in one seamless view.

9.4 Implementation

Most of implementation was halted until after the first user test, which in turn meant close to the end of the sprint. Only the main framework was set up at this stage, and the most of the team members started to familiarize themselves with Meteor and Meteoric in the first sprint.

9.5 Testing and Results

Before the testing began, the team discussed which prototype they thought were the worst and the best based on which is the most user friendly with criteria such as readability, complexity, flow and number of clicks. The team thought that the prototype that was the least user friendly was prototype number 1.1, and the prototype that was most user friendly was number 1.3.

The first user testing with the customer was held on the 22.09.2015. There were two doctors present, Florentin Moser and Jostein Dale. They both tested all three prototypes and gave us valuable feedback. The testing started with prototype number 1.2, then prototype number 1.1 and ended with prototype number 1.3. A reason for this order of testing was because we did not want to show the customer the easiest type first, and then show prototypes with increased complexity, so that the test person would not be biased of the subsequent prototypes. Another reason was that we wanted the customer to be pleasantly surprised when the third prototype was shown.

9.6 Customer Feedback

The customer was in general satisfied with the team's pace of progression. There were some things in particular that he enjoyed more than other about the prototypes. As expected, the customer had very little comments on the first prototype, other than the fact that it was a bit unwieldy to use, as he did not have anything to compare it to. When the second prototype was introduced it became easier for the customer to give feedback. As expected, the main points he made about the second prototype was that it had too many levels, thus too many clicks. When the third prototype was shown, the customer was indeed pleasantly surprised. The tab system was mentioned as the best way of organizing the information within the application. He expressed the most interest for the third prototype and further prototyping will be based mainly on the design of this prototype.

9.7 Retrospective

The retrospective section summarizes the sprint and reflects on what went well, what the group learned they should start doing in the next sprints, what could have gone better, and what the group should stop doing in the following sprints. The section also contains a burn down chart

displaying the work left to do versus time. Both the ideal and the actual plot is displayed figure 9.4 to get an comparison between the predicted work time and the time the group actually used to complete the items in the sprint 1's backlog.

9.7.a What went well

Most of what was done within the first sprint went well. The group dynamics and administration of tasks started to shape up really well and everyone within the team was being productive. Also the prototyping went very well, even though it was a big unconventional. Both of the test subjects provided very useful feedback for further development of the application. Based on the feedback the team also got confirmation that the chosen framework would work well for development.

9.7.b What shall we start doing

Since the implementation was slightly halted this sprint the team should be able to put in more hours in the next sprint, since there will be more areas to divide work tasks into and thus more parallel work can be done.

9.7.c What could have gone better

The team could probably have spent more time agreeing on a single design for the prototype, but it did not have a crucial impact on the user testing after all.
If the work management continues to work as well as it has done this sprint it is likely that the team will provide more hours as mentioned above.

9.7.d What should we stop doing

The team should stop producing three prototypes and focus on making a single prototype for the second user test based on feedback from the first user test.

9.7.e Burndown chart

The first sprint started well with an even workflow in the weekdays and some minor work being done in the weekends. The group finished every task the team planned to complete, but the time it needed to finish the prototypes were underestimated. The reason for the underestimation was the program Axure, which was more complex then the team thought. When there were not any specific tasks to do people in the team were assigned to documentation, hence the underestimation of documentation. This burndown chart does only represent the different tasks in the backlog written earlier in this chapter. Meetings, learning new technology, and research is not taking into account in this diagram. Even though every task in the backlog was completed the team realizes that the implementation should have started this sprint as mentioned in 9.7.b. That said the team was happy with the outcome of this sprint.

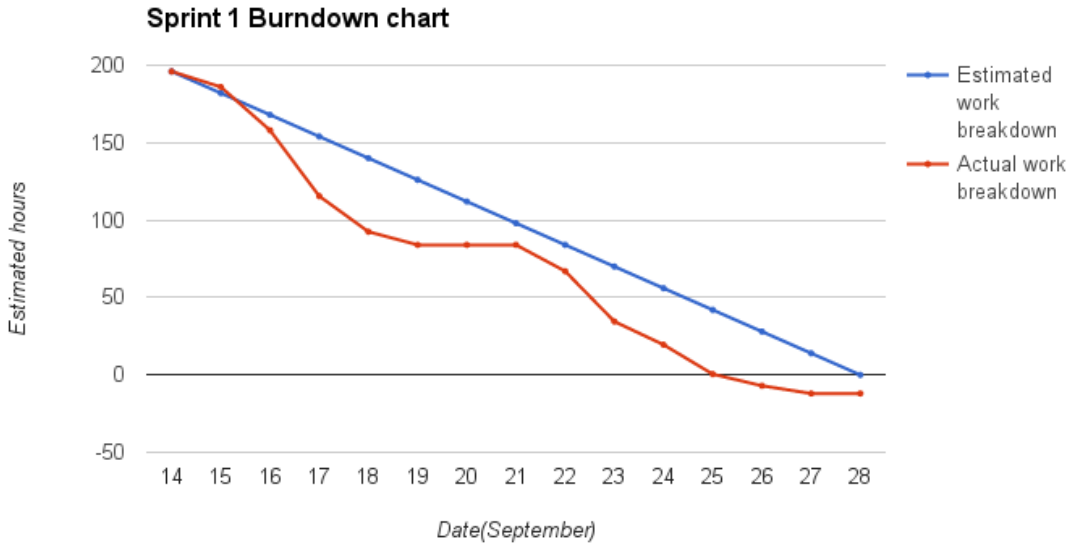


Figure 9.4: Sprint 1 burndown chart

10 | Sprint 2

Chapter 10 contains information about the how's and what's of the second sprint. Specifically it contains information about the sprint planning, duration, and backlog, as well as the most important tasks performed during this sprint. The second sprint was focused around initiating the implementation, building the second and third prototype, and creating as much content as possible for the midterm delivery of the report. All of this is introduced in separate sections and then summarized at the end of the chapter.

10.1 Sprint Planning

Most of the sprint planning was done at the end of sprint 1 and the group had set both goals and milestones for the sprint. The main focus of this sprint is on developing a second prototype for the customer, start implementing the front and back end of the application, and write as much content as possible in time for the midterm delivery. The deadline for the prototype implementation was on the 08.10.15 as the group was meeting the customer for the second prototype testing on the 09.10.15, which was one of the milestones of the sprint. The second milestone was to set up the skeleton code of the implementation by the end of September. The goals are described in more detail in section 10.3.

During sprint 1, the team discussed with the adviser about creating an administrator interface for the back end of the application. Although this had been discussed in the early stages of the project, the idea has not been presented for the customer before the start of the second sprint. The customer was very positive to the team implementing this, and therefore it was decided that the team should start implementing the administrator interface for the back end in sprint 2.

10.2 Sprint Duration

The second sprint lasted for three weeks, beginning on the 28th of September 2015 and ending on the 18th of October 2015.

10.3 Sprint Goal

The goals for this sprint was to develop an improved prototype for the second user testing as well as making progress with the implementation. The sprint contained a milestone for the implementation

where the goal was to have a functioning front end, similar to what was developed in the second prototype, up and running. In addition to the prototype and implementation goals there was also set goals for the documentation and the group wanted to complete the planning and requirements chapters that only needed some finishing touches. Other documentation goals was to write most of the technologies, development methodology, and sprint 1 chapters. Beyond that there were also a deadline for delivering the midterm version of the report on the 12.10.15. The midterm was meant to serve as a way the groups would receive feedback from the supervisor, and since there were no restrictions on how much of the report were delivered only a minimum limit for what should be delivered. This meant that the group would push to produce as much content as possible before the deadline in order to get as much feedback as possible.

10.4 Sprint Backlog

The sprint backlog section table 10.1 contains a list of the sprint backlog items that were completed during the second sprint. All the items are categorized by priority and are given an estimated time that it would take to finish the items, as well as the actual time used to complete the item. There is no order to the items. The visualization of the burndown of the tasks in the backlog is displayed in Figure 10.6.

No.	Description	Priority	Estimated Time	Used Time
01	Documentation	medium	150	150.5
02	Create prototype version 2	high	20	16.5
03	Create prototype version 3	high	10	8.5
04	Add skeleton code	high	5	8.5
05	Add example application for code demonstration and explanation	medium	4	2
06	Add basic data to use for demonstration	high	2	1
07	Add database collections for back-end	high	7	5
08	Change from implicit route name from path to explicitly named route	high	2	1
09	Add methods to connect database to front-end	high	5	3
10	Add bootstrap for initial database	high	3	2
11	Connect laws to views	high	3	2
12	Add methods to connect database to back-end	high	5	7
13	Add filter methods to get multiple outcomes	high	4	4
14	Add tab bar and navigation bars	high	8	4
15	Add icons to tab bar	high	2	1
16	Add logic for multiple choice buttons	high	30	21
17	Bug fixes	high	30	19
18	Testing axure prototypes	high	10	6
19	Integration testing	high	30	25.5
Total			**370**	**289.5**

Table 10.1: Backlog for sprint 2

10.5 Prototyping

In the second sprint there were two user tests with the customer. The first was performed the 09.10.15 and the second was performed the 13.10.15 After having decided to further develop the tab-design from the user testing in the first sprint, the group made improvements to the design and flow of the application based on the feedback. For the first user test the focus was on making sure that the flow of the application and the layout of each view were satisfactory. The group chose to only use "lorem ipsum", which is used as dummy text in web design, typography, and printing, in areas containing large amounts of text. Headers and buttons were filled with their usual text.

After having performed the first user test the group was asked to make a full implementation, with proper text, icons and buttons, as the customer had troubles picturing what the application would look like with the intended content. There was also some other feedback that will be further described in testing and results.

10.5.a Prototype 2

Prototype 2 was developed for the user testing that was performed 09.10.15 As mentioned above this prototype was designed to clarify the flow and layout of each view. Some of the new features for this prototype were, as displayed in the rightmost picture of Figure 10.1, the addition of the law view and the information view. The law view gives the user the opportunity to directly access all the relevant laws of the application and the information tab is meant to hold basic information about the application (creators, version number, and a user guide). The group chose distribute the views within tabs which were organized at the bottom of the screen. These did not have support for swipe functionality, since this was reserved for the tab system within the patient cases.

Based on feedback from the prototyping done in sprint 1 there was also added an option for the user to send relevant documents within each patient case to his or hers e-mail, as well as the opportunity to display only a short summary or the entire content in text boxes that in the previous prototype contained large amounts of text.

Figure 10.1: Prototype 2

10.5.b Prototype 2, complete version

After the user test on the 09.10.15 Dr. Moser requested that the group should return with a complete prototype a couple of days later. The prototype was to include all the intended text, formatting and flow that the customer had requested so far. This meant replacing all the dummy text with the proper text and formatting the views in the same way that the team imagined the final implementation. There were also some additional changes based of the feedback from the user testing on the 9th.

Most notably the customer wanted to restructure the path of the category 'Pasienten skader seg selv' ('Patient harms themselves'). This category should follow the same path as 'Pasienten nekter behandling' ('Patient refuses treatment') where the user is prompted questions about whether or not the patient is in need of emergency treatment, and if the patient is competent to give consent. Based on the answers on these questions the user should be navigated to the correct view, that being either laws concerning emergency treatment or a patients right to refuse treatment, or a view containing further classification of the patient.

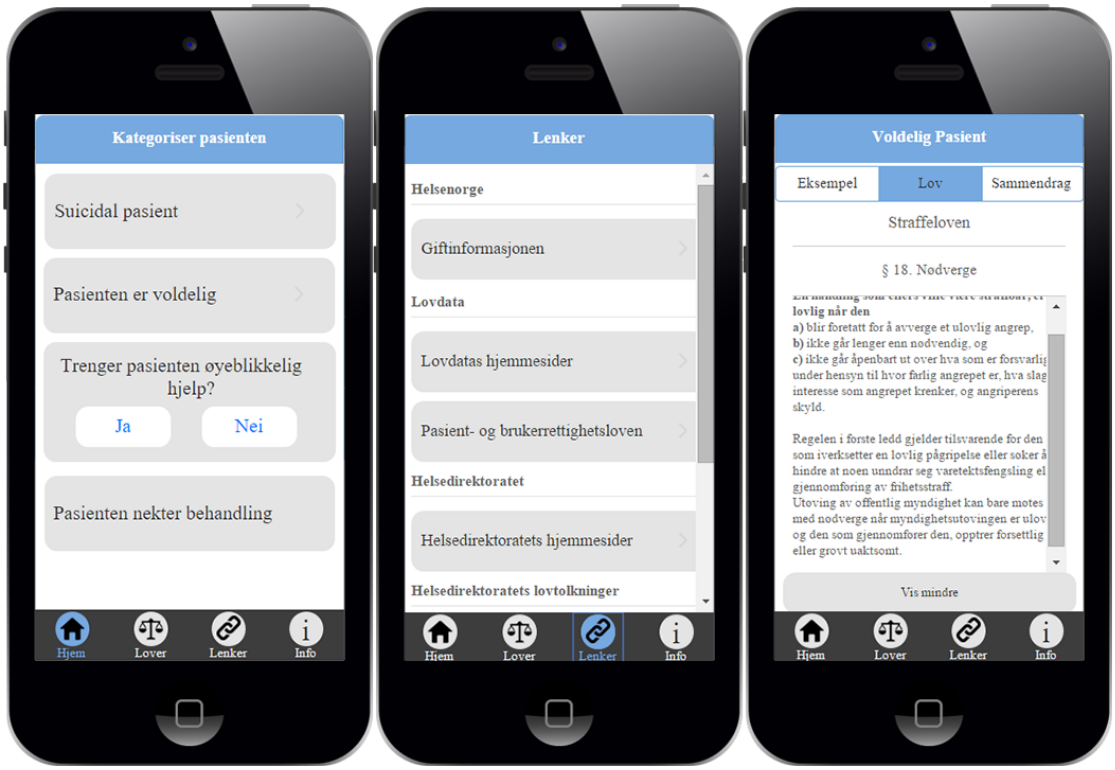


Figure 10.2: Prototype 2, version 2

As displayed in the middle screen of figure 10.2 the customer wanted a new view that would contain external links to sites that contain information that could be relevant for the emergency room. The view was added as a new tab in on the bottom bar. The layout of the grouping used for links from the same sites was also implemented in the law view, which originally had the layout displayed in the rightmost screen of figure 10.1.

10.6 Implementation

This section contains the different parts of the implementation, such as the structure and the various elements used. Some of the most important code snippets are also included, as well as an overview of the different fields.

10.6.a Structure

This sprint started with the initiation of the application as a skeleton of code. The structure is divided into folders, each containing files with separate concerns or responsibilities. The folders are *client*, *server*, *collection* and *lib*. When Meteor [14] compiles a project it packages all files together and loads each at a time. The load sequence is:

1. First, files in */lib* are loaded
2. Files are sorted by directory depth. Deeper files are loaded first
3. Files are sorted in alphabetical order
4. *main.** files are loaded last

This way, a folder can contain files concerning the view, controller or even the model related to a specific part of the application. The team chose to separate on a client-server pattern [3], the *client*-folder contains code related to the client, the *server*-folder contains code that will run on the server. In addition there is a *collections*-folder holding the models and a *lib*-folder with external libraries. Inside the *client*-folder the files are structured after the model-view-controller pattern [4] where the `app.html` functions as the view and `app.js` as the controller. The model would be the *collection*-folder, which defines the collections to be stored. On the server they are published, meaning they are made publicly available, and on the client they are subscribed on.

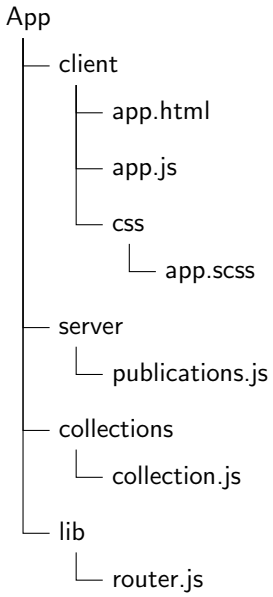


Figure 10.3: File structure of skeleton code

To reconstruct the logic of a flowchart the team needed a dynamic structure, so components could be added and subtracted from a path in the flowchart. The components introduced to cope with this is: **startcases**, **filters**, **laws** and **links**. This structure is illustrated in figure 10.4

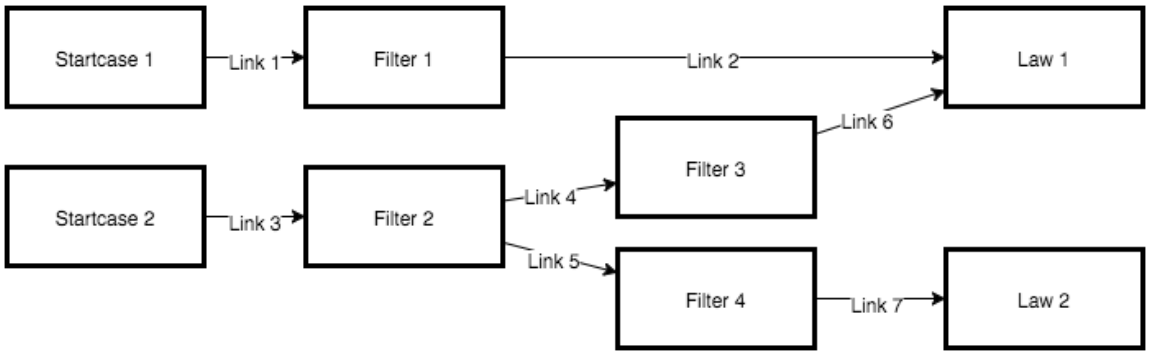


Figure 10.4: An illustration of the logic of paths

10.6.b Collections

When the skeleton was added the team started to define the different collections to support the path logic just introduced. 4 collections were added in this sprint, defined in figure 10.5. The top box denotes the collection name, the middle is the different fields, and the bottom box shows the collection methods.

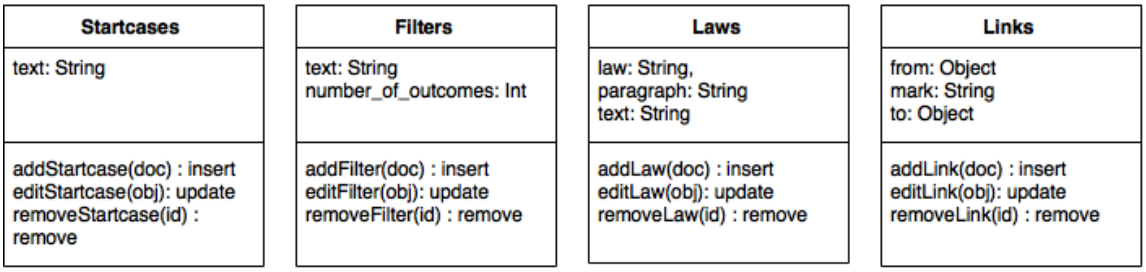


Figure 10.5: Collections introduced in sprint 2

The functions `add`, `edit` and `remove` are methods for altering the collections and their documents. Each field are described with the type of value it stores. The intended information to store is shown in table 10.2. After the collection structure were added the team made functionality to automatically load the collections with some initial information. This is found in the file `bootstrap.js`. It creates objects and fill up the database if 1) it is empty on startup, 2) the `UPDATE`-parameter is set to *true*. An excerpt is shown in listing 10.1

Collection	Field	Description
Startcases		
	Text	Textual content of the entry point
Filters		
	Text	Textual content of the law
	Number_of_outcomes	The number of outbound links from this filter
Laws		
	Law	Which legislation does this law belong to
	Paragraph	The paragraph number and name of the law
	Text	The content of the law
Links		
	From	The object in the start of the link (startcase or filter)
	to	The object in the end of the link (filter or law)

Table 10.2: The content of the collection fields for sprint 2

```
var UPDATE = true;

if (UPDATE == true) {
    Startcases.remove({});
}
Meteor.startup(function () {
if (Startcases.find().count() === 0 || UPDATE == true) {
    var startcases = [
        {text: "Pasienten er suicidal"},
        {text: "Pasienten er voldelig"}
    ];
    _.each(startcases, function(list) {
        Startcases.insert({text: list.text})
    });
}
```

Listing 10.1: A section of `botstrap.js` loading two *startcases*

10.6.c Template

After the collection was structured and bootstrapped back end and the server made them available by publishing them the team started working with the the front end. Inside the `app.html` there were constructed several templates, which works as different pages when connected in the `router.js`. Table 10.3 lists every template made in this sprint along with its functionality. Among the templates made were some without a back end at this point. It was decided that the end of a path should be divided into multiple tabs, with one for *Example* and one for *Summary*. In addition two of the main tabs was decided to be added, *Links* and *Info*. These templates does not yet have a accompanying back end, but this is the focus of the next sprint.

Template	Description
`defaultLayout`	The basic page structure, includes a top bar with title and the `Mtabs`
`Home`	The start page for the application, containing the `startcases`
`Laws`	Page listing all relevant laws
`Law`	Page displaying a given law
`Next`	A intermediate step for a path with a `filter`
`End`	The end view for a path
`Links`	Page with relevant external URLS
`Info`	Page with information about the application
`Example`	A view inside the end of a path, giving an example of the law
`Summary`	A view inside the end of a path, giving a summary of the path
`Admin`	The intended administrator interface
`Mtabs`	The main tabs, *Home*, *Laws*, *Links*, *Info*
`Ttabs`	Tabs inside the end of a path. *Example*, *Law*, *Summary*.

Table 10.3: Templates defined in sprint 2

10.6.d Controllers

When the views and models was implemented the functionality to tie them together was initiated, the *controller*. This functionality lives inside the `app.js` file inside the client folder. For every template created, this file contains **helpers** and **events**. The helper is a connection with the database and the events are functionality that reacts on interaction in the view. Listing 10.2 shows a helper for the `home` template. By calling `startcases` in the template you receive a list of all startcases from the collection. To call this function from the template and iterate over the list, you will write {{`#each case in startcases`}}.

```
Template.home.helpers({
        'startcases': function() {
        var cases = Startcases.find({})
        return cases && cases
    }
})
```

Listing 10.2: An example of a template helper

To cope with interactions you use events. Listing 10.3 shows the functionality that runs when a button is pressed on the `home`-template.

```
Template.home.events({
        'click .button': function(e) {
                if ($(e.currentTarget).attr("level") === "start") {
                        var f = buttons.pop()
                        Session.set(f, false)
                        $(e.currentTarget).fadeOut();
                        Session.set('currentFrom', this)
                        Session.set(e.currentTarget.id, true)
                        buttons.push(e.currentTarget.id)
                }
        }
})
```

Listing 10.3: An example of a template event

All helpers and events introduced in this sprint is displayed in the following tables, table 10.4 and table 10.5.

Template file	Helper name	Description
Home		
	`startcases`	Returns all startcases
	`links`	Returns all links
	`laws`	Returns all laws
	`filters`	Returns all filters
	`singleLink`	Return first link from a given object
	`yesLink`	Returns all links from a given object with *YES* as a mark
	`noLink`	Returns all links from a given object with *NO* as a mark
	`allLinks`	Returns all links from a given object
	`notClicked`	Returns true if button is not yet clicked
	`to`	Returns the object which a link points to
	`currentFrom`	Returns which item the link pointed from
	`oneEither`	True if startcase leads to one option
	`twoEither`	True if startcase leads to two options
	`oneYes`	True if *YES* mark leads to one option
	`twoYes`	True if *YES* mark answer leads to two options
	`oneNo`	True if *NO* mark answer leads to one option
	`twoNo`	True if *NO* mark answer leads to two options
Next		
	`current`	Returns which filter are currently active
	`currentLinks`	Returns all links pointing out from the active filter
	`to`	Returns the objects pointed to from the active filter
Laws		
	`laws`	Returns a list of all laws
Law		
	`thisLaw`	Returns the law for a given ID

Table 10.4: Template helpers defined in sprint 2

Template file	Event name	Description
Home		
	`click .button`	Sets variables that are used for the next level based on which level in the path is next

Table 10.5: Template events defined in sprint 2

10.7 Testing and Results

The first user test of sprint 2 was performed only with our customer contact Dr. Florentin Moser. The test was performed in an informal manner with the purpose of displaying the alterations and improvements made to the initial prototype. No assignments were made for Dr. Moser to perform, he was only asked to move around the application in the same fashion he pictured using it inside

the ER and think out loud so his thought process could be better understood.

The second user test was also performed with Dr. Moser and the focus for this one was the textual content of the application. Dr. Moser browsed through every view of the app and gave thorough feedback. He also provided links for the new tab that he thought were relevant and also said that other interest groups would provide their own links for the application.

After the two rounds with user testing our customer contact seemed satisfied with the solution that the group had developed and the pace of the project so far. Most of the feedback regarding the last prototype had to do with specifics in the text inside the application and also some clarifications around the external documents that were relevant inside each of the patient cases. After having dug into the specifics Dr. Moser found that only one or two of the cases had relevant documents that needed to be filled after or during the examination of the patient, and thus only a few of the summary - now called documentation - views needs the ability to send documents.

10.8 Retrospective

This section summarizes the second sprint and reflects on what went well, what the group should start doing in the upcoming sprints, what could have gone better in the sprint, and what the group should stop doing after the end of this sprint. Figure 10.6 displays the burn down chart from the beginning to the end of the sprint.

10.8.a What went well

Much like the first sprint most of what was done within sprint 2 went well. The group keeps working at an reasonable pace and most notably the implementation is shaping up nicely. The groups administration ability got tested with two of the group members working remotely, but through proper communication and great leadership it worked out well. Dr. Moser along with the group also felt that the user tests were a success and the group rounded off the prototyping with some useful feedback for the implementation.

10.8.b What shall we start doing

After receiving some feedback from our adviser and having discussed it within thoroughly within the group and the customer, the group has decided to develop a back end that administrators will be able to access in order to add, delete, or modify the content of the application. The motivation behind implementing a back end like this is to reduce the cost and time it takes to modify the application. By making the interface a user friendly interface that users without any particular technical experience will be able to operate, the need for hiring external consultants to perform this task is relieved.

In addition to continuing developing the front-end and back-end of the application, the customer wished that the team would add a sentence in each law page, where this sentence would function as a small summary of the law paragraph text. The team agreed that this was a good idea, and decided to start implementing it in sprint 3.

The team started the construction of the interface and logic for the administrator view during sprint 2 and will continue to further develop and implement the back end in the next sprint. The team has also agreed to improve the communication between the implementation and the documentation teams after experiencing a lack of communication between the two team towards the end of the sprint. The group shall also start documenting the sprint chapters as the sprint progresses.

10.8.c What could have gone better

As mentioned above, the communication between the implementation and documentation teams could have been improved as the group experienced a lack of communication between the teams. There has also been some discussion about how the implementation team could improve their use of the Trello board in order to keep an updated backlog for each of the sprint.

10.8.d What should we stop doing

The group should stop using time on prototypes and now focus their time and effort towards documentation and the final implementation.

10.8.e Burndown chart

In the second sprint, the team started implementing the application. The reason for the overestimation of the programming tasks is that the team did not know how long each task would take, and the fact that the team members that were set to program did not know how well they would work together. The programmers were not familiar with the framework that was chosen for this project, which also played a part in the overestimation. There were mainly three people working on the programming tasks, and the collaboration between them worked surprisingly well, which is one of the reasons that some of the programing tasks were finished more quickly than anticipated. The meetings, lectures and research tasks are not included in the backlog.

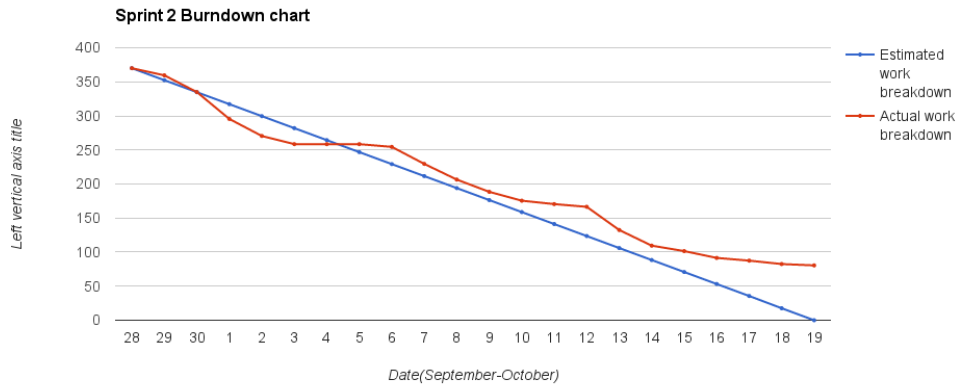


Figure 10.6: Sprint 2 burndown chart

11 | Sprint 3

This chapter contains information about the execution of sprint 3. It presents the different phases within a sprint; the planning phase, during phase and retrospective. The third sprint was focused on implementation and styling, as well as documenting what was done in the report. The goal for this sprint was to finish the implementation of the application, and to do as much styling as possible.

11.1 Sprint Planning

The planning for sprint 3 was started at the end of sprint 2. If there were certain tasks that were not completed by the end of the second sprint, the person in charge of this task was asked to finish it during the first week of sprint 3. All group members were given new tasks in the beginning of sprint 3 by using the Trello board. While there were certain tasks that the team members had to finish in this sprint, they were also able to choose among several tasks in the Trello board as they finished the previously assigned tasks.

By this point, the group had been split into two; three people working full time on the implementation, and three people handling the report, research and other tasks that needed to be done, for instance the styling of the user interface of the application. This was done so that the two different aspects of the project work could be done in parallel, and to increase the efficiency.

As well as finishing the implementation of the main view, the work on the administrator view was also continued. There were little focus on the styling of the administrator view, but rather the functionality.

As the deadline of the report delivery approached, there were a lot of time assigned to working on the report.

11.2 Sprint Duration

The third sprint lasted for three weeks, beginning on the 19th of October 2015 and ending on the 1st of November 2015.

11.3 Sprint Goal

In addition to having the goal of finalizing the styling, and fixing the bugs in the functionality for the main view, another major goal for this sprint was to finish the functionality of the administrator view. The team has scheduled a user testing in sprint 4, so the team aspired to make the application look as if it could have been released after the user testing session. In addition to these goals, the team had the ambition of finishing as much styling on the administrator view as possible, even though the team knew that it would probably be unrealistic to finish it completely.

11.4 Sprint Backlog

The sprint backlog section table 11.1 contains a list of the sprint backlog items that were completed during the third sprint. All the items are categorized by priority and are given an estimated time that it would take to finish the items, as well as the actual time used to complete the item. There is no order to the items. The visualization of the burndown of the tasks in the backlog is displayed in Figure 11.3.

No.	Description	Priority	Estimated Time	Used Time
01	Documentation	medium	100	34,5
02	Add URL links and display them	medium	6	8
03	Add phone numbers and display them	medium	6	8
04	Add data to database collection	high	15	12
05	Edit tabs view	medium	6	8
06	Update bootstrap with correct law texts	high	3	4
07	Add new fields for one-liners in each law page	high	5	6
08	Add iOS to platforms	high	3	4
09	Add Android to platforms	high	3	5
10	Add info page with tabs	medium	8	10
11	Add plug-in for in-app browser	medium	6	8
12	Create suggestion for styling	medium	20	23
13	Add styling to laws	high	10	19
14	Add styling to tabs	medium	8	7
15	Add styling to home buttons	medium	6	4
16	Add styling to link page	medium	6	5
17	Create suggestion for admin styling	medium	20	15
18	Bug fixes	high	50	36
19	Integration testing	high	20	46.5
Total			**324**	**263**

Table 11.1: Backlog for sprint 3

11.5 Implementation

This section contains the different parts of the implementation, such as the structure and the various elements used. Some of the most important code snippets are also included, as well as an overview of the different fields.

11.5.a Structure

During this sprint the skeleton structure did not scale with the amount of code and `app.html` and `app.js` grew too big. An effort was made to restructure these files into multiple, smaller files concerning a page or template at a time. This new structure can be seen in figure 11.1. This structures holds some initial and global information in the root of folder *client* and groups the views in `/templates` and controllers in `/js`. Within the templates folder, multiple files regarding the home page and paths were grouped in the `home` folder, and elements concerning the administrator interface was grouped in the folder `admin`.

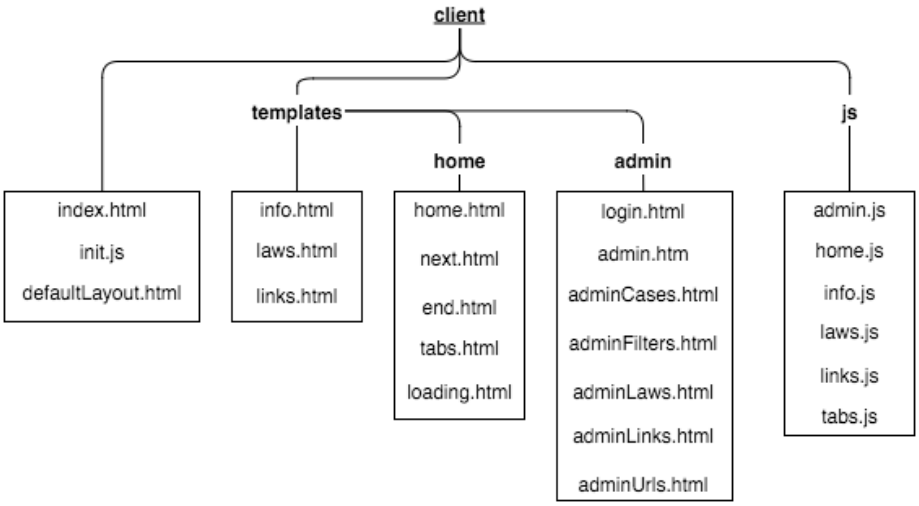


Figure 11.1: File structure for the `client` folder by the end of sprint 3

11.5.b Collections

This sprint started with the extension of the back end to match the front end from the last sprint. This means updating the database with three additional collections, `Info`, `Urls` and `Numbers`. There were also added three additional fields to the `Laws` collection. Figure 11.2 displays the altered and added collections, with changes marked with green.

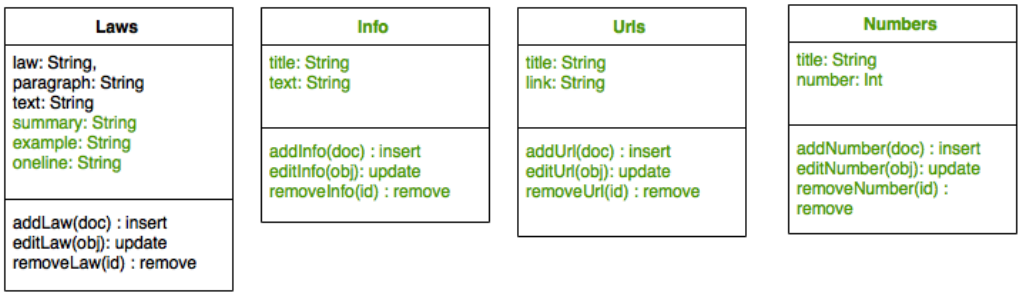


Figure 11.2: Collections added and altered in sprint 3

The table 11.2 explains the intended information to be stored in each field.

Collection	Field	Description
Laws		
	Summary	A sentence that summarizes the law text
	Example	An example case where the law applies
	Oneline	A custom sentence describing the content of the law
Info		
	Title	A title for this link
	Text	The external URL (http://www...)
Urls	*Title*	A title for this information element
	Link	Textual description of this information element
Numbers		
	Title	The target for this number
	Number	The telephone number

Table 11.2: The content of the added collection fields for sprint 3

The new collections were published by the server and subscribed on the client, and the `bootstrap.js` were updated to load more initial information to the collections. The listings 11.1 and 11.2 shows and excerpt on how the collections are made available, both by the server and on the client.

```
Meteor.publish('startcases', function() {
    return Startcases.find();
});

Meteor.publish('laws', function() {
    return Laws.find();
});
```

Listing 11.1: The way collections are published in `server > publications.js`

```
Meteor.startup(function () {
        if (Meteor.isCordova) {
        Meteor.subscribe("startcases");
        Meteor.subscribe("laws");
        // ...
        }
});
```

Listing 11.2: The way collections are subscribed in `client > init.js`

The new fields to `Laws` will be used with the internal page tabs in the end of a path, while the new collections, *Info* and `Urls` will receive independent pages and be added to the main tab system.

When new pages need to be added, the `router.js` controls this. It assigns URLs to templates. An example of this is seen in listing 11.3.

```
Router.route('/laws', function (){
    this.layout('defaultLayout');
    this.render('laws');
}, {
    name: 'laws'
});
```

Listing 11.3: The way pages are routed in `lib > router.js`

11.5.c Templates

This sprint, the templates grew by 8 additions. A `loading` template, with a spinner before the template is ready, a `login` template, requesting user name and password, and one `admin` template for each element, *cases*, *filters*, *links*, *laws*, *info*, and *urls*. There were also made tabs in the top of the `links` page to switch between *urls* and *numbers*.

Template	Description
`loading`	Displays a spinner if template is not ready
`login`	Prompting for user name and password
`adminCase`	Administrator interface to manage startcases
`adminFilters`	Administrator interface to manage filters
`adminLinks`	Administrator interface to manage links
`adminLaws`	Administrator interface to manage laws
`adminInfo`	Administrator interface to manage information
`adminUrls`	Administrator interface to manage URLs
`urls`	Sub page of `links`, listing relevant URLs
`numbers`	Sub page of `links`, listing relevant phone numbers

Table 11.3: Templates added in sprint 3

11.5.d Controllers

Several helpers were also introduced to accompany these new templates. The functionality for the admin templates typically is the same as for the reflected template without admin. For example, `admincases` was equipped with a helper `'cases'` which returned all start cases.

`Category`-helpers were added to **laws** to group the elements in the list together by legislation.

Both the **info** page and the **links** page were divided into tabs. For the info page the tabs were `about`, `howto` and `conditions`, and for the links page the tabs were `urls` and `numbers`.

Template file	Helper name	Description
Admin		
	`cases`	Returns all start cases
	`laws`	Returns all laws
	`links`	Returns all links
	`filters`	Returns all filters
Home		
	`username`	Returns the user name of the logged in user
	`link`	Returns a single link
	`filter`	Returns a single filter
	`law`	Returns a single law
Laws		
	`category`	Returns a list of all distinct legislation in the Laws collection
	`categoryLaw`	Returns all laws for a given legislation
Links	`one/two`	Determines which tab the user is in
	`urls`	Returns all URLs
	`numbers`	Returns all telephone numbers
Info		
	`one/two/three`	Determines which tab the user is in
	`about`	Returns the info-element regarding *About the app*
	`howto`	Returns the info-element regarding *How to use the app*
	`conditions`	Returns the info-element regarding *About the customer and developers*

Table 11.4: Template helpers added in sprint 3

When introducing a tab system the team needed events to load the correct sub page when the user clicks the tab. When loading the `law` view for a specific law the transitions looked like you changed a main page, so the team implemented an event to load this view in a different fashion. All events implemented in this sprint can be seen in table 11.5.

Template file	Event name	Description
Links		
	`click #one`	Sets current tab to 1
	`click #two`	Sets current tab to 2
Info		
	`click #one`	Sets current tab to 1
	`click #two`	Sets current tab to 2
	`click #three`	Sets current tab to 3
Laws	`click #law`	Turns off transition animations

Table 11.5: Template events added in sprint 3

11.5.e Styling

Some effort went into styling in this sprint. A few elements was styled directly in the templates (HTML), especially when there was a need to customize some of the default Ionic elements. For example, to make a thin line dividing elements on the screen, an *item-divider* was manipulated like this: `<div class="item item-divider text-center" style="border-bottom: 3px; border-bottom-style: solid; border-bottom-color: blue;">{{this}}</div>`

A need arose to style multiple elements with the same styling, in particular the start case buttons, so these were wrapped in `button-container`'s and styling was added in the `client > css > app.scss`. Ionics default `<a>` elements styling did not fit with the listing of the external URLs and phone numbers, so a `link` class was added and styled in the same document. Laws was grouped by legislation and an item divider was added to separate them.

11.5.f Deployment

In the start of this sprint some effort went into running the application on devices. Up until now, the application had been running locally, on the developers computer, and on the web by deploying to one of Meteors servers:

```
meteor deploy tiaco.meteor.com
```

This works as the central server which all the mobile devices will communicate with. This way, if the customer updates the database, all clients will also be updated.

Meteor also provides command line tools for deploying the application to devices. As of this point, Meteor only supports deploying on devices from a UNIX based operating system [13].

```
meteor run ios-device --mobile-server tiaco.meteor.com
```

This will package the application as an **Xcode**-project [17] and the user will need to run the application from within Xcode.

```
meteor run android-device --mobile-server tiaco.meteor.com
```

This will packages the application for an Android device, and starts the application on a connected device. A series of development tools will need to be installed and the device properly

configured before this can be accomplished. The procedures to do this can be found in the online guides here [31] and here [46].

11.6 Testing and Results

In this sprint there were no testing with external people, but there were many tests internally, such as system and integration tests.

11.7 Retrospective

The retrospective section summarizes the sprint and reflects on what went well, what the group learned they should start doing in the next sprints, what could have gone better, and what the group should stop doing in the following sprints. The section also contains a burn down chart displaying the work left to do versus time. Both the ideal and the actual plot is displayed figure 11.3 to get an comparison between the predicted work time and the time the group actually used to complete the items in the sprint 3's backlog.

11.7.a What went well

The functionality of the application was finished in the third sprint, and the team was satisfied with the product. The team spirit continued to be high in sprint 3, and the team members were motivated and focused on the tasks that were to be completed.

11.7.b What shall we start doing

The team has scheduled a user testing in sprint 4, and so the team members need to prepare for this type of testing. The application must be tested on different platforms and any bugs or errors must be fixed within the date of the user testing.

11.7.c What could have gone better

In the first week of sprint 3 the team suffered with some illness and absence of a few team members, which affected the amount of work that was done and the workload of the rest of the team members. It is inevitable to avoid illness, and so the team could not have done anything to prevent this, but the team members that travelled during this sprint should have worked more on the project while they were away.

11.7.d What should we stop doing

Instead of styling each element, the styling needs to be centralized into one single file.

11.7.e Burndown chart

Some of the implementation tasks from the second week were continued in the third sprint, by expanding the files with additional code and data. Compared to the actual work that was done, during

this sprint the team has also overestimated, like in sprint 2. The reason for this overestimation is mentioned in section 11.7.c.

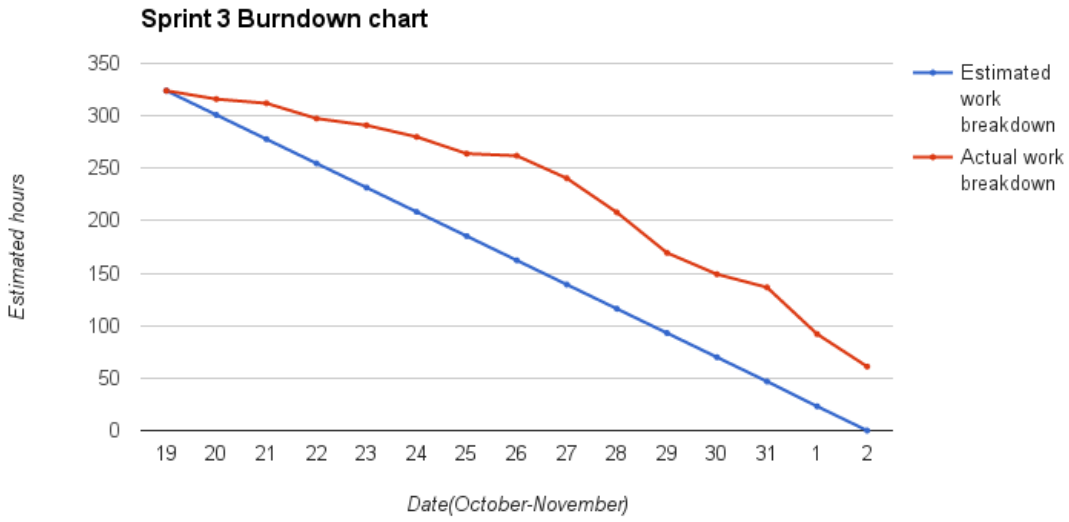


Figure 11.3: Sprint 3 burndown chart

12 | Sprint 4

This chapter contains information about what has been done, and how this was executed, during the fourth sprint. It presents the different phases of the sprints and therefore includes planning, retrospective, and the phase in between, which focuses on the implementation and styling. An overview of the testing as well as some screenshots from the final product is included. Because this is the last sprint of the project, the focus will be centered around finalizing the report, fixing functionality involving the administrator view, and straightening out minor bugs in the application, as well as preparing as much as possible of the presentation. A major usability testing was scheduled at St. Olavs hospital, and a final meeting with the customer was planned so that the textual content in the application would be legally correct.

12.1 Sprint Planning

The planning of sprint 4 was started at the end of sprint 3 and continued in the beginning of sprint 4. If there were certain tasks that were not completed by the end of the third sprint, the person in charge of this task was asked to finish it during the first half of sprint 4. All group members were given new tasks in the beginning of sprint 4 by using the Trello board. While there were certain tasks that the team members had to finish in this sprint, they were also able to choose among several tasks in the Trello board as they finished the previously assigned tasks.

12.2 Sprint Duration

The fourth, and final, sprint lasted for three weeks, beginning on the 2nd of November 2015 and ending on the 12th of November 2015.

12.3 Sprint Goal

The goal of this final sprint is to finish the implementation of the application, both back-end and front-end, to finalize the report and to start working on the presentation.

12.4 Sprint Backlog

The sprint backlog section table 12.1 contains a list of the sprint backlog items that were completed during the fourth, and final sprint. All the items are categorized by priority and are given an estimated time that it would take to finish the items, as well as the actual time used to complete the item. There is no order to the items. The visualization of the burndown of the tasks in the backlog is displayed in Figure 12.13.

No.	Description	Priority	Estimated Time	Used Time
01	Documentation	high	250	201
02	Add final styling to application	medium	30	22,5
03	Move one-liner to top of page	medium	2	1
04	Add dynamic path storing for display at documentation view	high	3	2
05	Divide numbers into internal and external	medium	2	1
06	Remove duplicate laws from list of laws	medium	2	2
07	Add global subscription to database	medium	1	1
08	Add "edit" and "add" buttons for all editable pages	medium	5	9
09	"Add new category" for admin implemented	medium	8	5,5
10	"Delete path" for admin implemented	medium	4	3,5
11	Create tree structure for all possible paths	medium	4	4
12	Add initial admin user and security for it	medium	4	2
13	Add structure for showing related documents	medium	5	4,5
14	Add functionality to preview documents	medium	3	2
15	Add swiping functionality	medium	2	4
16	Bug fixing	medium	30	35
17	Integration testing	high	12	22,5
18	Final usability testing	high	18	12
Total			**385**	**334,5**

Table 12.1: Backlog for sprint 4

12.5 Implementation

This section includes the advances made in code base in sprint 4. As this is the last sprint the material presented here are the final state of the system. The resulting structure is presented, and changes made to the collection, templates and controllers. Finishing touches regarding styling is presented in section 12.5.e, and an overview of the resulting system can be found in detail in appendix I.

12.5.a Structure

The restructuring done in sprint 3 proved to scale well and was not changed further. The structure is divided into client-side and server-side code, along with one folder for libraries and one for the collections. There is also a folder called *public*, which will hold assets and other static files, as the relevant documents to be served. The complete file structure can be seen in figure 12.1

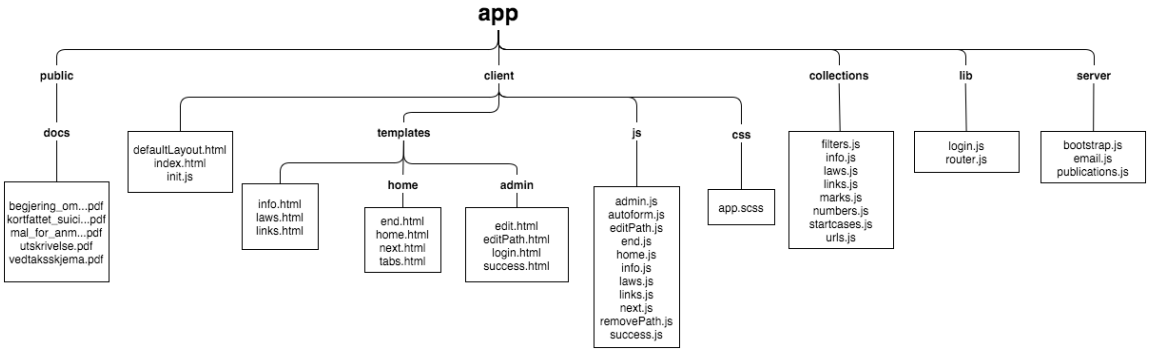


Figure 12.1: File structure for the application by the end of sprint 4

12.5.b Collections

The only alteration to the database in this sprint was an additional field to the `Numbers` collection. The changes are displayed in figure 12.2 and the table 12.2 describes the type of information to be stored in this field.

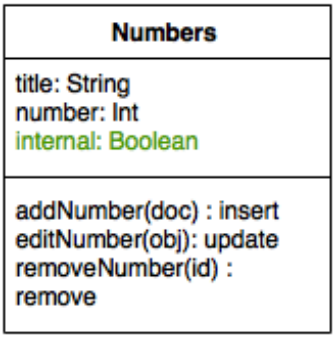


Figure 12.2: Collections added and altered in sprint 4

Collection	Field	Description
Numbers		
	Internal	Is this an external or internal number (true/false)

Table 12.2: The content of the added collection fields for sprint 4

For a complete list of all collections defined in the application, see appendix I.2.

12.5.c Templates

In this sprint the team decided to restructure how the administrator interface should work. The intention was to create separate views for a authenticated user with the administrator options. To make the editing more intuitive they chose to view the application in the exact same way, but with administrator options showing as buttons on top of the regular view. This made the `admin` templates introduced in the previous sprint obsolete. This was replaced by two groups of templates: `edit` and `editPath`.

As for the *edit* group, there exists individual templates for each element of the site that can be updated by a user. This means that there are `editLayout`, `editCase`, `editFilter`, `editLinks`, `editInfo`, `editLaw`, `editUrl` and `editNumber`. All of these does essentially the same thing for each component: They return the form of the component to be altered. The listing 12.1 shows how these templates are structured. There were also made similar templates for adding components.

```
<template name="editcase">
    {{> quickForm    schema="StartcaseSchema" collection="Startcases"
                     id="editStartcaseForm" type="normal" doc=selectedDoc
    }}
</template>
```

Listing 12.1: The content of the `editCase` template

When trying to add or change a path we discovered that much more functionality was needed. This now lives inside the `editPath` file. Here are templates for all the different functionality, such as adding buttons, changing questions and determining outcomes.

The `info` page was equipped with tabs, `about`, `howto` and `conditions`. The entire tab system received a renovation in this sprint. The global tabs on the bottom of every view, linking to *home*, *laws*, *links* and *info* now exists in its own file called `tabs`. The top tabs that are found in the *end* of a path, *links* and *info* pages now lives in the respected page's file. It is now possible to swipe between these tabs, and there have been made separate templates for the individual sub pages of the tabs. The `summary` tab from the end of a path has now been renamed to `documentation` and displays a list of relevant documents which the user is able to send to him- or herself through email. The templates added in this sprint can be seen in table 12.3.

Template	Description
`edit`	A group of templates providing forms for editing and adding components
`editPath`	A group of templates to assist in editing and adding paths
`documentation`	Renaming of `summary` from last sprint
`about`	The page displaying information about the customer and developers
`howto`	The page displaying instructions on how to use the app
`conditions`	The page displaying information about the content of the app
`success`	The page displaying that the path addition was successful

Table 12.3: Templates added in sprint 4

The templates for individual pages was re-factored into multiple smaller templates, combined together in a `layout` template for each pages. This layout defines how the page should look and includes a `header`, possibly the `topTabs` and the main bottom tabs, `mtabs`.

For a complete list of all templates defined in the application, see appendix I.3.

12.5.d Controllers

Helpers

Since the administrator interface was restructured all the previous helpers for this was removed. All components of the application was equipped with an admin helper for their edit template called `selectedDoc`. Multiple helpers were added for the *editPath* template as well, as seen in table 12.4

Template file	Helper name	Description
Admin		
	`selectedDoc`	Returns the selected item to be edited
editPath		
	`next`	Returns next mark to be connected to a filter or law
	`hasnNext`	Returns true if there are marks to be connected
	`selectedCase`	Returns the startcase for the current route
	`showQ`	Shows add yes/no question button if applicable
	`addingLaw`	Returns true if a law is being added
	`laws`	Returns laws sorted by law-category
	`category`	Returns a list of law-categories
	`categoryLaws`	Returns all laws for a given category, unique laws only
	`default`	Sets number of outcomes to 2 for the object
	`default`	Sets number of outcomes to 2 for the object
	`scase`	Returns true if any startcases added through current addpath
	`fil`	Returns true if any filters are added through current addpath
	`li`	Returns true if any links are added through current addpath
	`la`	Returns true if any laws are added through current addpath
	`addedScase`	Returns startcases added through current addpath
	`addedFilters`	Returns filters added through current addpath
	`addedLinks`	Returns links added through current addpath
	`lawOrFilter`	Returns the paragraph of a law or the text of a filter
	`startLink`	Returns "Kobling til startkategori" if the input is empty string

Table 12.4: Template helpers added to the group of `admin` and `editPath` templates in sprint 4

The other templates received helpers for swiping between tabs and find relevant documents to laws, among others. A list can be found in table 12.5

Template file	Helper name	Description
End		
	`swipeleft .endSwipe`	Sets the tab to the next to the right
	`swiperight .endSwipe`	Sets the tab to the next to the left
	`One/Two/Three`	Sets the tab to the first/second/third
	`thisLaw`	Returns the current law for the tabs
	`case`	Returns the current start category
	`path`	Returns the current path taken
	`anyRelated`	Returns true if there are any related documents
Info		
	`swipeleft .infoSwipe`	Sets the tab to the next to the right
	`swiperight .infoSwipe`	Sets the tab to the next to the left
Links		
	`swipeleft .linkSwipe`	Sets the tab to the next to the right
	`swiperight .linkSwipe`	Sets the tab to the next to the left
	`numbersEkst`	Returns external numbers
	`numbersInt`	Returns internal numbers
Next		
	`toLaw`	Returns true if the link links to a law
	`case`	Returns the current start category

Table 12.5: Template helpers added to the group of `end`, `info`, `links` and `next` templates in sprint 4

For a complete list of all template helpers defined in the application, see appendix I.4.

Events

While the events in sprint 3 was focused around how to determine which tab the user is looking at, this sprint implements multiple events to support the administrator interface. An overview of events from this sprint can be found in table 12.6. For a complete list of all template events defined in the application, see appendix I.5.

Template file	Event name	Description
Home		
	`click .removeButton`	Displays confirmation popup, removes path if confirmed
EditPath		
	`click .lawChooser`	Creates link between current question/answer to the law clicked
	`click .addLawButton`	Sets addLaw to true, indicating the user wants to add a law
	`click .connectN`	Renders the form for connecting to an N answers question
	`click .connectQ`	Renders the form for connecting to a Yes/No question
	`click .connectL`	Renders the display for connecting to a law
End		
	`click #one`	Returns true if current tab is set to 1
	`click #two`	Returns true if current tab is set to 2
	`click #three`	Returns true if current tab is set to 3
	`click .openDoc`	Sends app to url for the document clicked
	`change .checkbox input`	Adds or removes item from array it was checked or unchecked
	`click #send`	Displays prompt for user email, sends email if it is provided
Next		
	`click .firstLevelButton`	Adds question to path of questions answered, sets next has been visited
Laws		
	`click .removeLawButton`	Removes the law from the database
Links		
	`click .removeUrl`	Removes the url from the database
	`click .removeNumber`	Removes the number from the database
Success		
	`click .homeButton`	Sends the app to the 'home' view

Table 12.6: Template events added in sprint 4

12.5.e Styling

A lot of effort went into finishing touches in this sprint. As the team had added "edit" and "remove" buttons to all views they needed to be placed correctly on the page. The individual buttons was classed with a name and styled in `app.scss`. The text sizes was adjusted in almost every view, and the *oneliner* tab was moved to top of the end view after feedback from the user testing 13.5.c. The opacity of the title indicating which tab you are on was increased to improve readability.

12.6 Testing and Results

In this sprint there were performed one user test, as well as integration tests.

12.6.a Usability testing

At the start of sprint 4, the team conducted a usability testing session at the St. Olavs Hospital with six medical interns. The interns got an introduction to the purpose of the application from the team members before starting the test. The tasks that the interns were asked to perform, and more information about this user testing can be found in the chapter 13. To summarize the test results, the users were generally very happy about how easy and intuitive the application was, as well as saying that the application could become highly useful in the everyday use. The System Usability Scale (SUS) gave the application a score of 90 out of 100, which gives the team an indication that our focus on making the application as easy to use as possible have paid off.

12.7 The final product - Akuttjus

Because sprint 4 is the last sprint of this project, the team decided to include screenshots of the final product - the application called "Akuttjus". Figure 12.3 through Figure 12.8 are a part of the main view, while Figure 12.9 through Figure 12.12 are a part of the administrator view. Explanations of these screenshots, and screenshots of all functionality in the application, are further described in Appendix J.

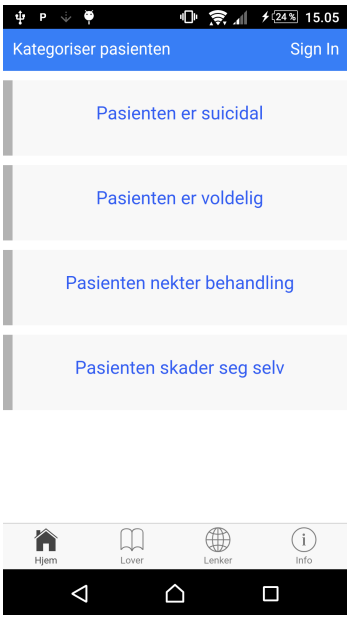


Figure 12.3: 'Front' page

Figure 12.4: 'Law' page

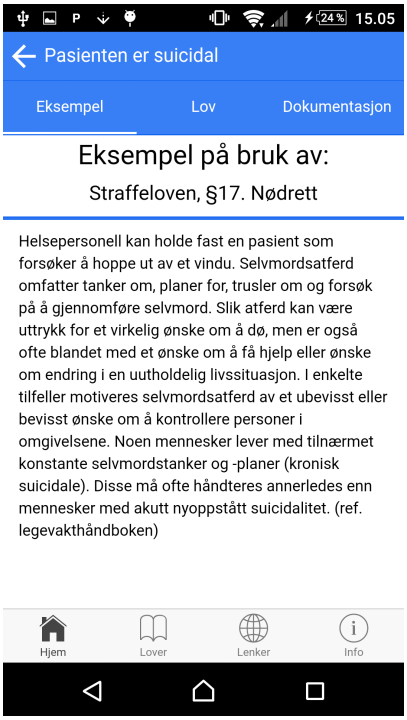


Figure 12.5: 'Example' page

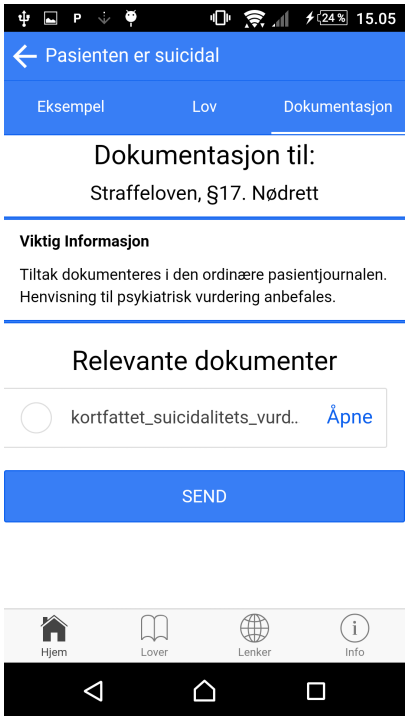


Figure 12.6: 'Documentation' page

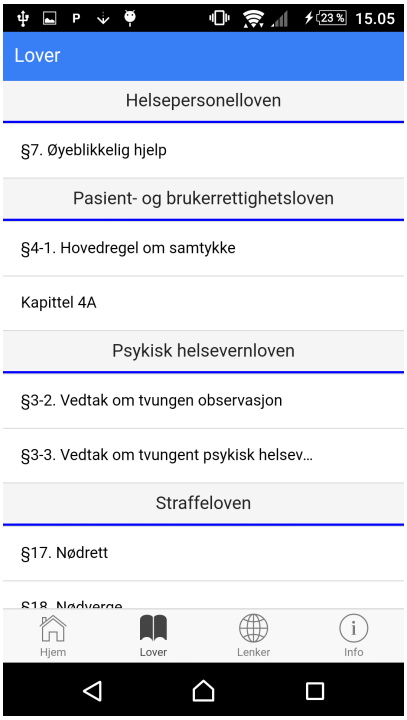


Figure 12.7: 'List of laws' page

Figure 12.8: 'Law' page

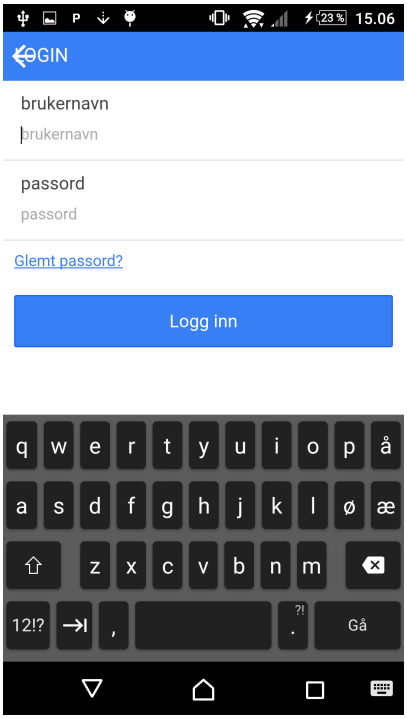


Figure 12.9: Admin: Log in

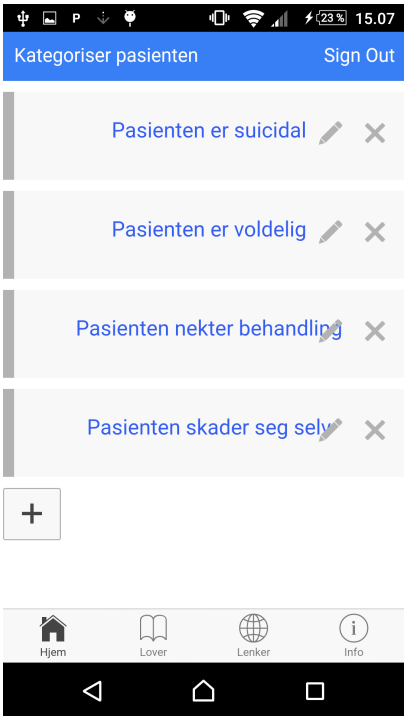


Figure 12.10: Admin: Edit main page

Figure 12.11: Admin: Delete category

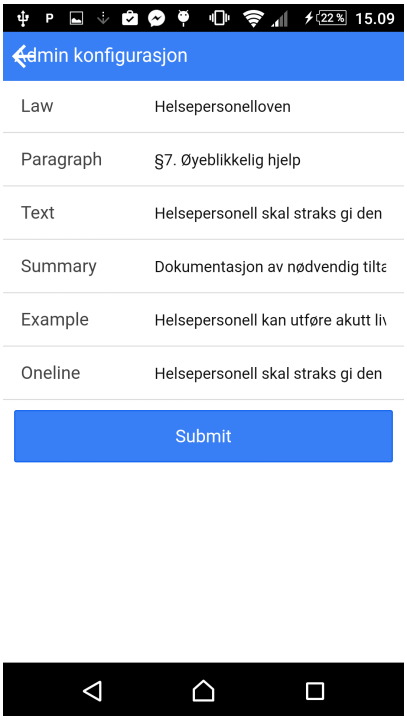


Figure 12.12: Admin: Edit 'law', 'summary', and 'example'

12.8 Retrospective

The retrospective section summarizes the sprint and reflects on what went well, what the group learned they should start doing in the next sprints, what could have gone better, and what the group should stop doing in the following sprints. The section also contains a burn down chart displaying the work left to do versus time. Both the ideal and the actual plot is displayed figure 12.13 to get an comparison between the predicted work time and the time the group actually used to complete the items in the sprint 4's backlog.

What went well

The feedback from the different people the team tested the application on, was very good. Some of the comments the users mentioned made the group feel very satisfied with the application. In addition the team were able to finish the report within the deadline with as much content in the report as the team had planned.

What shall we start doing

The team should concentrate on the presentation of the project which is on 18.11.15 as the delivery for the report is the last day of the sprint.

What could have gone better

The deadline that was set for the implementation was on the 09.11.15. This limit could have been set earlier, so that the team would have started implementing earlier. As predicted, there were a lot of work to do in sprint 4. The work could have been distributed better, by doing more work in the previous sprints.

What should we stop doing

Writing the report and implementing. The deadline for the implementation was set at 09.11.15, in order to be able to document the last bits of code that were added. If errors are found after the deadline, the team should document them in the further work section.

12.8.a Burndown chart

Some of the implementation tasks from the third week were continued in the fourth sprint, by expanding the files with additional code and data. This sprint's actual work compared to the estimated work is similar to what happened in sprint 2 and 3. The reason for the overestimation is because the team knew there were quite a lot of work related to finishing code and documentation remaining. The team tried to take this into consideration and that is why the hours estimated were much higher then other sprints, when comparing the amount of days in the different sprints. Some team members also suffered from short termed illness, which is on of the reasons why the team fell behind the intentional plan for this sprint. That said the team worked really hard the last days and therefore made up for what time was lost earlier this sprint.

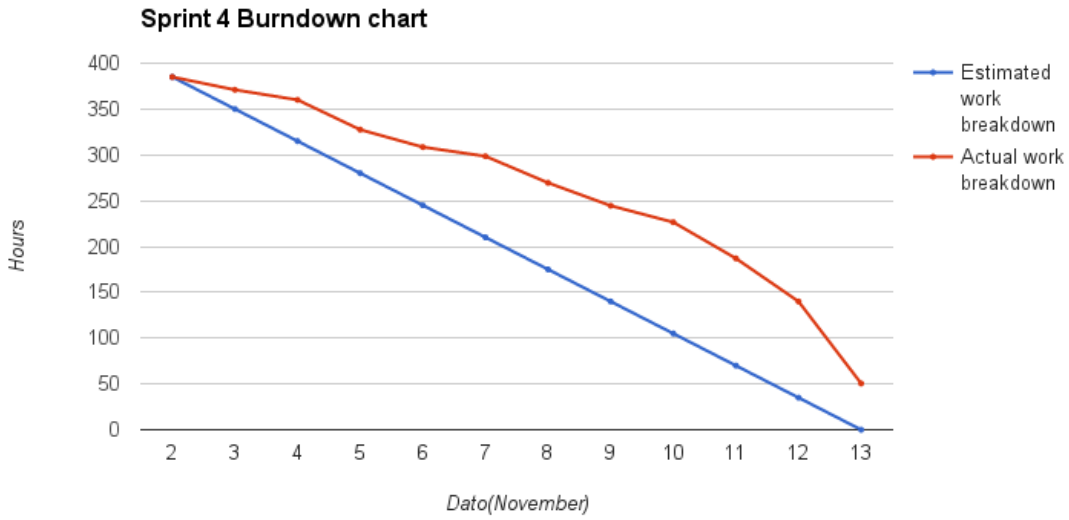


Figure 12.13: Sprint 4 burndown chart

13 | Testing

This chapter contains information on how the team planed and perform different tests of the application throughout the project. A rationale of testing is followed by a description of the different types of tests applied. Usability testing will receive the most attention in this chapter and will include the results of a prototype test and a user test.

13.1 Methods and goals

The goal of performing tests are to assure that the code produces the expected functionality, and that the functionality fulfills the customers requirements. The core test plan is to make sure code modules behave correctly, both on valid and invalid input, and that modules will be integrated to the system correctly. Further the team will run system test to make sure the functional requirements are covered, and usability tests to get feedback on the front-end design and refine it. At the end of the project, the resulting product will be presented for the customer for a final acceptance test.

13.2 Unit testing

Unit testing is the process of testing individual components in isolation [22]. This is a time consuming process and not applicable within the time frame of this project. Instead of unit testing all components of the system the plan is to integration test when new components are added to check whether the system still is stable. This gives a quick indicator on which components are faulty and needs further inspection.

13.3 Integration testing

Integration test is an extension of unit testing where units are aggregated into components and the interface between them are tested [28]. As the team develops a unit of the system the developer starts a local server running the application and investigates how it works and continue developing until the unit behaves as desired.

This as well is a time consuming process, but the code base will progress as the unit is tested and further developed and the process contributes to refining the system to a desired state.

13.4 System testing

System testing during development involves integrating components to create a version of the system and then testing the integrated system [22]. The focus is testing the interaction between components and the behavior of the system. This phase assures that the requirements are met, by manually testing the application.

13.5 Usability testing

User or customer testing is a stage in the testing process in which users or customers provide input and advice on system testing [22]. This is the most valuable testing phase of the project, and it includes the customer and a selection of potential end users. During the usability tests the team will observe and make notes in an observation form. This form is included in the Appendix E. The test subject will be given a series of tasks to conduct and will be asked to fill out a System Usability Scale-form afterwards.

13.5.a System Usability Scale

The System Usability Scale (SUS) is a form of measuring the usability of a system; how well a system works and how well it is received by the users. This form is usually filled out after a user test by the users. It contains ten questions, with five option answers, a scale from 1-5, where 1 is "Strongly disagree" and 5 is "Strongly agree". An example of this type of form is shown in Appendix D.

13.5.b Prototype testing: 22. September 2015

The team conducted a prototype testing session at the St. Olavs Hospital with the customer contact and one of his colleagues that also works at the ER.

The team prepared 3 scenarios, three different tasks, that the test subjects will be asked to perform. The scenarios are listed in table 13.1.

No.	Task
01	A suicidal person is arriving at the emergency room. Locate the corresponding law.
02	A violent person is arriving at the emergency room. Locate the corresponding actions or measures.
03	A person arrives at the emergency room which refuses treatment, does not need immediate medical attention, is not cooperative and has a suspected serious mental disorder. Navigate to the summary for this patient.

Table 13.1: Table of tasks for prototype testing

These are tasks that will cover the main functionality of the application, so that results can be gathered for all of them and analyzed afterwards. The subjects will be presented with three

different prototypes, and will preform the tasks on all three. Before the test was conducted the team emphasized the following for the test subjects:

1. The text in the prototype is not correct
2. The main focus will be functionality and flow through the app
3. The design of the app is not complete

During the execution of the tasks the team did not guide or instruct the test subjects. Their task was to make notes where the subject struggled. The entire session was filmed to be analyzed afterwards.

Feedback from the prototype testing

The session found place on 22. September 2015 in a meeting room at St. Olavs Hospital with Florentin Moser and Jostein Dale. Florentin performed all tasks for the three prototypes, while the rest observed. When he completed the tasks both Florentin and Jostein compared the three prototypes and pointed out what they liked from each. The feedback received is summarized in the list below.

- A list of all laws, relevant in this application
- Display the law first, when a case is selected
- Remove the section about actions or measures
- Provide more feedback in the case when you are asked multiple questions

Figure 13.1: Prototype testing

In sprint 2, two other user tests with the customer was performed. These tests are further explained in section 10.5.

13.5.c User testing: 04.11.15

The team conducted a usability testing session at the St. Olavs Hospital with six medical interns. The interns got an introduction to the purpose of application from the team members before starting the test. The tasks that the users were asked to complete is listed in Table 13.2. The tasks were originally written in Norwegian, but is translated into English.

The level of technical knowledge varied from user to user, but all users owned a smartphone, so they were all familiar with apps. The average time spent on each task is listed in Table 13.3.

No.	Task
01	Find the law to use when handling a suicidal patient.
02	Find an example of a violent patient.
03	Find "Straffeloven §17".
04	Find out which law applies to a patient that: - refuses treatment, and - does not need immediate help, and - that is not competent to give consent, and - is suspected of having a serios mental illness.
05	Find the phone number to "Østmarka ekspedisjon".
06	Find the link to "Lovdata".

Table 13.2: Table of tasks for user testing

Task No.	Avg. time spent	Min. clicks needed from home page
01	5 seconds	1
02	9 seconds	2
03	13 seconds	2
04	23 seconds	4
05	16 seconds	2
06	6 seconds	1

Table 13.3: Table of average time spent on each task

User No.	Avg. time spent on user test (1)	SUS score
01	74 seconds	90
02	82 seconds	85
03	79 seconds	85
04	66 seconds	90
05	65 seconds	95
06	65 seconds	95
Total	71,83 seconds	90

Table 13.4: Table of SUS score from each user

(1) This time is measured from when the test leader initiated the task, by saying "begin" until the user completed the task, by saying "finished". Some of the users had thoughts or questions after a task was completed. The time spent on this purpose was not included in the calculation.

Feedback from the user testing

Some of the feedback from the user testing have been implemented or changed in the application. The font-colors have been changed and the one-liner was moved from the middle of the page to the top of the page. Although there were some that thought it was hard to find the page where the relevant phone numbers were, this has not been changed, because the users added that if they had used the application once, then they would be able to remember where to find this page. In general, all the users thought the application was easy to use and that they would use it if they ended up working in the ER. The concrete feedback is listed below.

- Some of the texts are hard to see because of the font-color.
- Did not expect that there would be phone numbers within the "Links" tab.
- It was hard to see the text that was written when the brightness was lowered.
- The purpose of the application should be in the info-page.
- The one-liner text should not be in the middle of the page, because then it looks like it is a part of the law paragraph.
- Some of the categories need more than one law in order to make a decision
- The one-liner looks like a heading.
- Some of the laws could have a direct link to relevant phone numbers. (For example when the doctor needs to ask for a psychiatric evaluation)
- The design is suitable for hospital use.
- The application is surprisingly easy to use.

13.5.d Acceptance testing

The final phase of the test plan involves the customer testing the end product, and deciding whether or not it is ready to be accepted and deployed.

The team planned to have an acceptance test in the last week of the project. Unfortunately, this was not possible to arrange, because the customer was not available for a meeting the last week of the project. The planning of this acceptance test should have been presented for the customer earlier, so that the test would have been performed. Even though the team did not perform a dedicated acceptance test, the team has been getting feedback from the customer at every meeting that was held, and at the last meeting with the customer on 06.11.15, he stated that he was 'so happy with the application that he got chills'. The team takes this as an extremely good compliment, and concludes that the customer has accepted the application that the team has implemented.

13.6 Summary

This chapter shows that the goal of performing tests are to assure that the code produces the expected functionality, and that the functionality fulfills the customers requirements. It describes how unit testing is more time consuming than integration testing, and that the latter is preferred by the team. Some missing functionality were listed after the usability tests, but in general, all the users thought the application was easy to use and that they would use it if they ended up working in the ER.

14 | Evaluation of the product

This chapter involves the evaluation of the finished product. An overview over how the implementation was made and it's problems will be reflected upon. After this section every requirement, functional and non-functional, will be displayed in a table, where the status of it is represented as completed or not. The customer's response to the product, is also presented in this chapter.

14.1 Implementation

As is typical for programming projects, the true scope and difficulty of the application was discovered over time, through further talks with the customer, NTNU, and through the implementation process.

14.1.a Scope

Especially the requirement of solid information structuring and storage required extensive work, which was unfortunate as it was a late suggestion by NTNU guidance and not something we could pick up or predict easily from customer interaction, as they deemed it largely unnecessary. The reason we include it is because it would be necessary for most expansions for the application, which both the customers and the team see many opportunities for. It also enables good knowledge management, which though currently a low priority for the customer, is generally good practice for applications and necessary in many future expansion options.
Thus, the full scope of the requirements for our frameworks was creating an environment that facilitates developing a hybrid application, which has offline functionality, knowledge management solutions, that is very easy to use, that can display, store, add and modify a tree structure of questions and information, where the administrative function of modifying and adding information in the app should be usable for an administrator without any software development experience. The resources for this project's implementation are very limited, so the framework environment needs to facilitate fast development as well. Because we have no outside support for learning implementation environments and because the time frame for self-learning is limited, previous familiarity with the frameworks is heavily favoured.

14.1.b Cordova

Without enough resources for a true native product for the most common phones and browsers to be feasible, the team needed a hybrid solution. We solved this requirement with Apache Cordova,

which to a large degree worked well. The app looks and feels close to as expected on all tested devices and browsers, and platform-related issues were minimal. How the app looks on certain screen sizes and environments could be optimized further, but this is a time-intensive task which was relatively low on our priority list. One of the main drawbacks of Cordova (and hybrid development in general) is lower performance than native, but because our app uses few resources on front-end tasks like animations and rendering this was no real issue. An added benefit is that Cordova development is almost exclusively done in HTML, CSS and JavaScript, which let some members of the team benefit from previous experiences. Assuming the project does not require functionality that cannot be delivered through Cordova, the team would definitely use it in the future.

14.1.c Meteor

For the main portion of the application we used Meteor with various framework additions. Meteor with Meteoric let us create UI elements and modify them in a short amount of time. The time saved through using Meteor and Cordova were a large part of the reason we were able to deliver on most requirements within the time frame of the project. However, Meteor and the necessary additions we included also have some drawbacks. Meteor and especially a lot of the frameworks we added are still young. We encountered a large amount of outdated information and examples, which lead to misinformation that necessitated debugging of faulty code that had been broken by framework changes since the information was published. Some portions of the frameworks even had breaking changes during our development process.
Because of poor documentation and resources for newer features of some frameworks, for instance Meteoric, extensive research of source code was necessary to implement some functionality well. This is of course very time consuming and challenging for a developer completely unfamiliar with the framework at project start. The team had little to no experience with Meteor and the framework additions we used. This reduced the quality of code and led to longer development time, but is largely unavoidable for all feasible options because we are general programmers and not experienced web developers. A lot of the logic is implemented in JavaScript (with additions), however, which is similar to programming experiences the team has.

14.1.d Back-end

As advanced knowledge management is not the focus of Meteor's back-end development frameworks we had to implement a lot of custom functionality for the administration function and the structure of information in the app in general. Because of these issues, low development time and limited possibilities for learning, the back-end of the application was a time sink and relies on some poor or mediocre code and methods.
Because Meteor's main focus is fast dynamic and reactive display, it has less extensive features relating to persistent information, like offline functionality and advanced information structuring.
Offline functionality is as of the writing of this chapter not functional, because this is not intended usage of Meteor's main database solution MongoDB. We relied on a young addition called GroundDB with limited support which the team have had trouble getting to work in our application.
Security for the database was largely handled by Meteor's own frameworks for user accounts. We used server side methods for database interaction only accessible by users in the system, which is the standard Meteor way. We used the Meteor package useraccounts for login functionality. This

was simple, fast and functional. This was a very positive experience with Meteor development for the team.
Optimally we would implement the back-end in a more advanced environment for information structuring and create a more robust handler for the tree structure of information in the application, but this would be a difficult task within the time frame even with perfect knowledge of all requirements at the very start of the development process.

14.1.e Conclusions

Frameworks for fast hybrid development enabled us to create a product that would be completely infeasible within the time frame with custom native solutions. However, the scope of technology the product requires needs to be as extensively explored as possible before frameworks are chosen to make sure the they solidly support as much of the requirements as possible. Otherwise, development time will be extended or the quality of implemented solutions lowered.
The challenges of working with frameworks that are young and under heavy development need to be taken into account and planned for. Dealing with breaking changes, missing functionality, limited support and poor learning resources and documentation takes extra development time. We could have chosen a database system with more advanced features to handle the information structure, and explored frameworks similar to Meteor to find something with better handling of the offline case and few database changes in general.

14.2 Fulfillment of requirements

In chapter 4, the functional and non-functional requirements are presented in table 4.1 and 4.2, respectively. This section presents the requirements that are fulfilled and the ones that are not completed.

Req. ID	Priority	Completed	Comments
F1	High	Completed	Start menu with 4 categories
F2	High	Completed	Shows the choices the user has made
F3	Low	Completed	Changes the color of the side bar
F4	High	Completed	
F5	High	Completed	Common color scheme and fonts
F6	Medium	Completed	Tab bar at the bottom
F7	Medium	Completed	Pop-up is implemented
F8	Low	Not completed	Did not have time to implement it
F9	Medium	Completed	Front page goes straight to content
F10	Medium	Completed	
F11	Medium	Completed	
F12	Medium	Not completed	Implemented, but undiscovered bugs denies the app to fully load

Table 14.1: Fulfillment of functional requirements

As stated in table 14.1 and 14.2, some of the requirements are not fulfilled. Requirement F8 was down prioritized because the team realized that there was not enough time to finish all requirements.

Req. ID	Priority	Completed	Comments
NF1	High	Completed	Boots within 5 seconds when online
NF2	High	N/A	Needs further testing in the future
NF3	High	Completed	At most 4 clicks to get to the most advanced law
NF4	Medium	N/A	Needs further testing in the future
NF5	Medium	Completed	Administrator view
NF6	Medium	Completed	Administrator view
NF7	Medium	Nearly completed	Completed to a certain degree, law texts are hard-coded
NF8	High	Completed	Proof read by lawyer
NF9	High	Completed	
NF10	High	Completed	Tab bars are used
NF11	Medium	Nearly completed	Administrator must be logged in, Admin cannot delete or upload documents
NF12	High	Completed	Added one-liner which sums up each law
NF13	High	Completed	Works on Android, iOs and Web
NF14	Medium	Completed	User tests showed that the app was easy to use

Table 14.2: Fulfillment of non-functional requirements

The requirement of F12, that the application should be able to be used offline, is not completed, even though the team wanted, and tried, to implement it. Requirement NF2 and NF4 are hard to test without having tested the system over a longer period of time, and therefore the team does not know if the requirements are fulfilled or not. Requirement NF7 is nearly completed, except the fact that the law texts are not dynamic. The reason for this is that the team did not have the time to research how to extract the laws used in this application from the "lovdata" database. The administrator is however able to change these law texts if they are logged in. The requirement NF11 is completed to a certain degree. The only aspect of this requirement that is not fulfilled is the possibility of the administrator to add and delete documents.

14.3 Customer response

Throughout the whole duration of the project, the customer has been satisfied with the work that the team has produced. There have been some changes to the design, and to the textual content, but the application itself, and its functionality have been received well by the customer. The customer has stated from the start of the project that he wishes the application to be launched in the future. His dream is that the application "Akuttjus" can be used in all the emergency rooms at every large hospital in Norway, by updating the application with local regulations for the different hospitals. The customer hopes that with this application, the usefulness of technology in hospitals will become obvious to the Norwegian Directorate of Health, and that the application might be supported by them.

14.4 Further work

The "Akuttjus" application was not quite finished when the project ended. From the start to the end the project has experienced several changes to requirements and new key features have been along the way. But due to the time restrictions of the project the group opted to leave out some of the features that were discovered in the later stages of the project. There were also some of the implemented features that could use some polishing, and as with almost every application "Akuttjus" has some minor bugs that could use a little smashing. All of the points introduced here and some others will be discussed further in the subsections below.

14.4.a Local procedures

One of the features that was discussed early in the project was the possibility to implement local procedures into the application. Even though every hospital in Norway is subject to the same national laws there are different interpretations of the laws and the local procedures may vary from region to region. Since the customer is eager to make "Akuttjus" a nationwide application it would be a good thing to support the option for the application to present the local procedures for each hospital and region within the country.

A way to implement this would be to make use of the integrated Global Positioning System (GPS)-system that most smart phones possess and based on the users current position the application would display the local procedures governing the users region. Some of the drawbacks of implementing it this way is the fact that not everyone wants to make use of the GPS-system on their phones, and the GPS might be unstable in regions where the phone signal is weak or non-existing. In order to mitigate this problem one could make use of Internet Protocol (IP)-addresses, and giving the user an option to manually select their region and hospital within the application.

Suppose the team had time to implement this feature it would also be necessary to implement functionality to add, edit, and delete information about the local procedures in the same way as for the rest of the application. But since this is a functionality that is dependent on the application getting distributed to other hospitals it did not make it into the teams backlog and is regarded as something that would be a key functionality to implement if the application were to spread to different hospitals in Norway.

14.4.b Security

The security of an application like "Akuttjus", where key information that could potentially affect patients in life threatening situations, is important, especially since the application now contains the option for administrators to add, edit, or delete any of the content inside the application. When the project was finished group had not had time to implement a form of security that would not be easily breached if malicious users where to try and alter the content of the application. Since the administrator functionality was not considered until almost half way into the project the focus was set on implementing the functionality for the administrator and not the security level of the authentication.

Within the final weeks of the project there also emerged some requirements for users to be able to

fill some of the forms for special cases from within the application. In order provide some kind of security regarding who should be able to fill out these forms the application would also need authentication of this group of users, and not only the administrators. Much like the authentication for administrators the authentication of other users was not implemented due to the time restrictions of the project, but in further development of the "Akuttjus" project this would be a key feature, since the application should, and probably would, not be distributed without this functionality due to the threat malicious users would pose to the application.

14.4.c Distribution of the application

As mentioned in the security section, the application was not published to any application store by the time when the project ended. The main reason for this being that several key features are still missing and as mentioned in the introduction to this section, the application is closer to being a prototype of the final product than being a final product. But in the case where the project was finished and the core parts of the application that is currently missing had been implemented the application would need to be distributed to the users. The easiest way to do this is to publish "Akuttjus" to application stores like Google Play Store and AppStore. Another option would also to provide download links on the web sites of the hospitals using the application or more central governing sites like the Norwegian Directorate of Health.

14.4.d Communication with the police department

In some cases patients tend to be violent towards health care personnel and the need for police assistance might be needed. An idea brought forward by the hospitals lawyer, Merete Blokkum, was to implement a version of the system that focuses on bringing forth the information that doctors and other employees in the ER should forward to the police in cases where the police is involved. By keeping the same format as the main view of "Akuttjus" each of the categories would direct the user to information which is relevant to the police when they document their work. In addition to this Blokkum also wanted to include information about how to communicate with the police and the form used to file police reports, as well as contact information for the local police department.

In order to implement this the team envisions a version of the application that becomes a lot bigger than today's application and the application that is now called "Akuttjus" will only serve as a one of two major parts of the future application. This would certainly be an interesting feature to implement and it would also serve as helpful tool for the employees in emergency rooms within the country.

14.4.e Tutorial

The tutorial feature is a feature that the team would very much like to implement into the application, but due to the limited time of the project this feature was not prioritized in the final product. Tutorials are always useful for applications that explores new domains and offers a user interface that might be unfamiliar to new users. Additionally since the functionality of the administrator is quite complex a tutorial of how to perform the different tasks inside the administrator view would be needed. Even though the application contains a 'how to use' section a graphical tutorial would serve as helpful tool.

14.4.f Offline mode

One of the issues with developing a hybid application is that it is heavily web based, and often relies on having an internet connection in order to function. The meteor framework the team chose to work with supports the use of ground:db which provides the ability to copy the content of the server database to the clients phone, and by using this the phone should be able to run without a connection to the server. This was one of the requirements, F12 in table 4.1, that established in the prestudy phase due to the hospitals somewhat unstable wifi-connection. By the end of the project the team was yet to figure out why the application would not run in offline mode, and it is one of the features that needs to be fixed in the future in order to satisfy the customers requirements.

14.4.g Documents

In the requirement NF11 from table 4.2 states that the administrator of the application will be able to add and edit content. This requirement also includes the ability to add and edit the external documents that are attached to some of the categories. Due to the time limitations of the project this feature was not implemented, but it is an important feature that needs to be implemented in a later version in order to give the administrators the ability to fully customize the categories that are added.

14.4.h General improvements

In addition to the major functionality described in the subsections above there are some general improvements that could be made to "Akuttjus". There are were some minor feedback from the user tests performed in sprint 3 which were not fixed before the end of the project. One of those were to add references to EQS by adding numbers to the procedures based on the laws, allowing the user to quickly perform a search in EQS in order to find more information about the procedure. The implementation team also experienced difficulties when trying to extend the application with the possibility for users to upload documents, as stated in requirement NF11, in table 4.2. This issue was not resolved by the end of the project.

There are also some other minor bug fixes that are currently known, and since the application has not undergone extensive testing there are most likely a number of bugs that has not been discovered yet.

15 | Conclusion

15.1 Lessons learned

Throughout this project the team went through all the phases of developing an application for a customer. This including a prestudy phase, an iterative prototype phase, an implementation phase, and a testing phase. For all of the groups members being involved in a project of this size was a new experience and the team learned a lot of lessons were learned. Most notably was the underlining of the phrase 'requirements always change'. Throughout the entire project the requirements for the application has changed as new features were constantly added. What started as a pretty small project ended up becoming to large for the group to finalize before the project ended. From this the group learnt that one should always be prepared for change, and thus plan an implement in the best way possible to mitigate the setbacks caused by changes in the requirements.

One of the earliest decisions the group made was on how the group should work, whether or not the group should have set hours where everyone would get together and work or everyone who was available to work should try to be at the office when they could. Since none of the members had the same time schedule the group decided every member should try to be at the office whenever they did not have other lectures or worked on other courses. This was not a particularly strict policy, and quickly lead to some members working from home from time to time. Whether or not this lead to less efficient work or not is hard to say, but in some stages of the project it is possible that the group would have benefited from having everyone work from the office since it makes communication between members a bit easier than having a large part of the communication go through social media chats.

Since this was the first project for most of the group members where they worked with a task given by a customer the importance of establishing and maintaining a good customer relationship was discovered. For this specific project the customer has been participating a lot, both with providing requirements, feedback, and a helpful hand with specifics regarding the health domain. From the start of the project the team was able to establish a good relationship with the customer and it was maintained throughout the project, and this has proven to be a valuable asset for the group.

The framework the group chose to work with, Meteor with Meteoric, was technology that none of the team members had any experience with and even though multiple members of the group is used to writing HTML, Javascript, and CSS which the framework is build on, getting to know the framework became a time consuming task. After the prestudy phase where the team discovered the

framework and did some initial tests with it, did not have much information about the framework and was just counting on the framework to provide the properties needed. Luckily it did, and the only trouble it caused the group was when the framework was updated and some of the existing syntax was altered. The lesson learnt based on this was that new frameworks may be cumbersome to learn, and making the entire project reliant on a framework that is in the development stage could prove lethal for the entire project.

Working with the scrum methodology the team got used to performing daily stand-up meetings where each member presents what they have done since last meeting, which task they are currently working on, and which tasks they will work on until the next meeting. With non of the members having much experience from the scrum methodology from earlier projects this proved to be a very useful tool with regards to project planning. By performing stand-up meetings with the members that were present each day the other group members were always up to date with who was working on what, and it made sure that everyone had something to do at all times.

One thing that followed from the group assuming that the project to be developed was not very large, was the fact that the start of the implementation was postponed and a lot of time was spent on developing prototypes. This, together with the ever changing requirements from the customer, led to the application not being finished by the end of the project. A lesson learned based of this is to start the implementation earlier, and maybe even try to follow a lean development methodology

As the end of the project came along the team sat down to produce the WBS the group quickly realized that the documentation of work hours throughout the project had been done poorly. The actual hours worked were documented throughout the entire project, but members had either forgotten to fill in their working hours into the WBS or in some cases put their hours in the wrong categories. This then lead to the process of having to work through every members hours throughout the duration of the project and try to get the hours allocated correctly. This proved to be a tedious task as it was quickly discovered that the group had only created larger, more general tasks on the Trello board, which lead to the difficult task of remembering what was done, and more importantly how long it took to perform the task. This was a valuable lesson for the team, following the scrum guidelines for the breakdown of tasks serves as a very useful tool when it comes to both estimating and documenting hours spent at the end of each sprint.

The final lesson that the team learnt is that absence over a prolonged period, whether it is due to illness or travelling, impacts the project. The group experienced prolonged illness which lead to a heavier workload on the remaining members of the group. There were also a period where several of the group members were traveling and working remotely, and in this period the group experienced a slight lack of motivation which lead to reduced productivity. Even though illness and traveling is something which is always regarded as a risk in a project, it was something that affected the group and its productivity, and in future projects the members might be better prepared to handle such situations and also they might be more resilient to the reduced motivation which often occurs as a result of uneven workloads.

15.2 Concluding remarks

As this project is the biggest and most extensive project any one on the team has been a part of, everyone in the team agreed that it was the most stressful, challenging, and time consuming project the team has been a part of. With that said, it was extremely educational and worth the time spent on it. The magnitude of what has been learned involving project management, team-work, programming, customer relationship, and development in general is a lot bigger than members of the group would have learned in the same amount of other courses.

It is a common thought in the team that the hours spent on this project has been worth while, because of the amount of experience everyone has obtained. With the combined skill level of the team, it was concluded that the final product was a success, even though the possibility of improvement is always present. According to the customer the application could have a massive impact on the percentage in which medical personnel documents the correct law, and that the correct actions is taken when a complicated situation occurs in the emergency room. This comment from the customer made the team feel like the application actually could help with problems in important situations. If the project should have been done all over again the team absolutely agreed on that starting the implementation sooner with have improved the final result of the application.

16 | Reflection

This chapter involves the reflection of the project's process. Group dynamic, the customer, and the advisors will be reflected upon in the project section and a reflection of the development methodology, technical problems, and the risks through the project will be expressed in process section. The group dynamic will be divided into paragraphs related to what phase they were in.

16.1 The project

The project section contains information about group dynamic during the lifetime of the project, the evaluation of the customer, and the advisors the team has been in contact with.

16.1.a Group dynamic

In the beginning of the project the course responsible arranged a kick-off session so that the team members could get to know each other. In this kick-off the group discussed the assigned project, along with the objectives for the project and the vision of the team. After this discussion, the team did some exercises to get to know each other better. The group members presented their software experience, hobbies and qualities. The team also formed rules for the group.

Preliminary study evaluation

In the preliminary study phase, the team worked well together, but not as often as they should, as shown in table 3.3. At this stage, the team were not aware of the amount of work that needed to be done. Even though the group knew that this course is demanding in terms of the amount of work that needed to be done in such a short amount of time, the group did not feel the need to work as much as they should have. The group dynamic was good, and roles of the different team members were formed.

Sprint 1 evaluation

When the first sprint began, the group realized that they needed to work more often in order to complete all the tasks that needed to be finished. The group started meeting more often, and the roles that each team member were assigned came into play. Two of the team members worked on making a prototype for the application, two started researching for which framework that would be used in the report, and the rest worked on the report and other miscellaneous tasks. By dividing

the group into three, the team achieve a higher degree of efficiency, which also influenced the morale of the team in a positive way. The team were motivated to work to reach a common goal.

Sprint 2 evaluation

In the second sprint, there were some restructuring of the smaller collaboration groups of two. The team quickly realized that they should have started the actual implementation earlier, and so the group split into two instead of three. One half of the group worked in the implementation, while the other half focused on the report, the second prototype, and other tasks that the implementation team did not have time for. At the end of sprint 2, the application had all the basic functionality needed, and the work of the functionality of the administrator view begun. This boosted the team's motivation even more, and the team members had at this point come to be friends, and the working environment was enjoyable, but still professional.

Sprint 3 evaluation

At the end of sprint 2, and the start of sprint 3, the group suffered from the absence of some of the team members, which lowered the motivation of the team members that were present. By knowing that this sprint was the second to last sprint, the team continued to work as if there had been no absence, and some of the team members even worked more hours than scheduled. Most of the group members had specific tasks assigned to them, but some of the team members helped both with writing the report and the implementation. The fact that the team members were flexible with what they were able to work at, was a big help to the group and the project work in general. At the end of sprint 3, the team were excited to perform the usability test in sprint 4, and so the motivation increased.

Sprint 4 evaluation

In the first week of sprint 4, there were exciting times for the group. There were several meetings this week, including meetings with the customer, the hospital lawyer, and a user testing session. Even though the team were tired from working more than usual in sprint 3, the team did not focus on that before the second week of sprint 4, because of the desire to perform as well as possible at the meetings and the user testing. When the second week of sprint 4 began, all team members, except for one, worked on writing the report. The team members found it easier to write at this point, because of the time pressure. The team spirit was still high, and the fact that the project came to an end helped to boost the motivation.

Summary of group dynamic evaluation

Overall the group worked well together during this project, and the motivation was in general high, except for at the beginning of sprint 3. Even though some team members have preferred to work individually, the collaboration within the group divisions have worked well. The team is generally satisfied with the work effort, and very satisfied with the application the team has made.

16.1.b The customer

The customer for this project, Dr. Florentin Moser, has been very eager in the whole process of the application development. He has been available for questions, and has generally been very helpful

towards the team. In addition to meeting the team when the team has requested it, he has also invited the team to come to the St. Olavs hospital for a round tour of the facilities. In some of the meetings the team had with the customer, Dr. Moser also invited other parties that were involved or interested in the project. The team has been on two separate meetings where the customer has invited the hospital lawyer, Merete Blokkum, and his colleague, Dr. Jostein Dale. In addition to arranging these meetings, Dr. Moser has also helped the group to set up a user testing session, where the customer has invited medical interns to test the group's application.

16.1.c The advisor

The advisor for the team, Soudabeh Khodambashi, has been of great service, and has been available for questions any time of the day, even though she travelled to the United States of America for a relative large amount of the project time slot. There has been scheduled an advisor meeting almost every week, where a status report was discussed, ideas for further progress and suggestions for improvement were considered. In addition to Soudabeh Khodambashi, the team have also had Pekka Abrahamson as advisor for a couple of weeks. As well as giving the team some valuable pointers in these weeks, he has also lead the team leader meetings which have been held six times over the project's lifetime, where advices and guidance have been given for general problems in the report. There has also been given the opportunity to talk and discuss about problems in the individual groups in these team leader meetings. Overall the advisors of this course have been helpful and given valuable information used to finish the final application and report.

16.2 The process

In this section an overview of the process is presented, where the development methodology and implementation will be discussed and evaluated. Furthermore there will be a description of the risks (6.1 and 6.2) the team encountered during the project.

16.2.a Development methodology

Since every group in the course was encouraged to use Scrum in the project, and all of the team members were very positive and motivated to try this methodology, it was decided that the team would use Scrum. Another reason why Scrum was chosen may be the fact that several of the companies visiting NTNU made compelling points about why Scrum is so successful in a working environment. Because companies who work with information technology often use the Scrum methodology, the team thought that getting knowledge in this methodology would be very helpful for future projects.

The team quickly understood some downsides with working with Scrum. One of the major problems was finding time to have daily stand-up meetings where everyone attended. The reason for this is that every specialization is represented in the group, and therefore there are a great deal of different courses that must be taken into account. Since the schedule were so different the team planned to meet a couple of days a week, and communicate daily updates through communication technologies and management tools like Facebook and Trello. Later on in the project the team was divided into two groups to simplify the communication process to some extent.

The schedule problem made it difficult to use a physical scrum board since the team were not working together that often. To solve this problem the team created a Trello board where every task was placed and delegated to the selected persons. This virtual scrum board may not have motivated us as much as a physical, but it was the best solution to the problem faced by the team. Google sheet was also used to track the time spent on the different tasks.

16.2.b Technical problems

One of the biggest difficulty with regards to the technical aspects of the project, has been to use unfamiliar framework. In addition to learning new technology, both Meteor and Meteoric has released new versions of their software during the period of the project, which lead to deprecated syntax, among other challenges.

The fact that Meteor cannot deploy the application from a Windows computer, has been inconvenient. Even though half of the team uses Apple's macbook, the other half has a computer with the Windows operating system, which meant that in theory only half of the group could deploy the application on their own computer.

Two other minor problems, was that one of the team member's laptop crashed, which lead to this member needed to work from home at his stationary computer, and the fact that the Internet connection was at times slow.

16.2.c Time estimation and consumption

As mentioned in the start of this report, the group had little experience with working on such an extensive project as this one. Even though the team are used to estimate time in the daily life, the team found it hard to estimate the amount of hours that would be used on tasks that they have not done before, such as working with the unknown framework, Meteor.

Ideally, as stated in the organizational demands, each team member would work approximately 25 hours per week for 11,4 weeks. This yields to a total of 1710 working hours. The team started dividing the 11 weeks into sprints, and would then calculate how much time should have been spent on the project each week, and thus each sprint. The team decided to divide the project into four sprints, with an additional phase of pre-study in the first few weeks, which was not included in the sprints.

Realistically, the team thought that 25 hours per week was unobtainable. In order to make a realistic estimation compared to the actual hours spent, the group set a goal for each team member to work approximately 23,5 hours per week, which would then reduce the total amount of hours for the whole project to 1610 hours.

After calculating how many hours needed to be spent each week, the group started to allocate hours to the different aspects of the project, such as the planning, research, implementation, documentation, and testing. This is shown in table 16.1.

Task	Hours estimated per week
Research	10
Implementation and testing	41
Documentation	50
Meetings and planning	29
Administrative work	11
Total	**141**

Table 16.1: Estimation of work per week

The estimation of the different tasks turned out to be not that far away from the actual hours spent per week. In sprint 1, the team underestimated the total hours worked by 12 hours, which is a reasonable margin for the estimation. In sprint 2, the team overestimated by 80 hours. One of the reasons for this, was that the team experienced the absence of two of the team members the last week of the sprint. In the first week of sprint 3, there were also some absence, which made the team overestimate once again, by 61 hours. In the last sprint, sprint 4, the team estimated even more hours as the three weeks long sprint 2. The reason for this is that when sprint 3 ended, the team realized that there was still a lot of work that remained, both on the report and on the implementation. Even though the estimated hours for this sprint was quite high, the team overestimated by only 13 %, which is not that much considering the amount of hours estimated per day versus the actual hours worked per day. If the team includes the unexpected absence of the team members, the team did a decent job of estimating the amount of hours worked, compared to the actual hours spent.

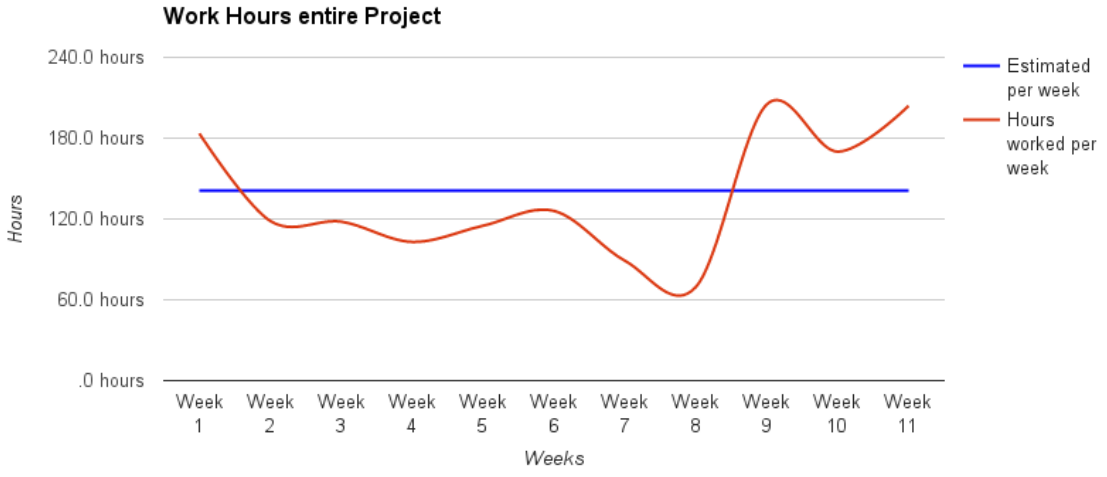


Figure 16.1: Breakdown chart for all sprints

There was a lot to learn about time estimation throughout the whole duration of the project. The team made the estimations based on the limited experience the group had, and by researching the time spent on other projects that were similar to this one. As it turned out, it was impossible to estimate the time spent on specific sub tasks that the team had not done before. Even though some of the estimation of the smaller tasks were wrong, the total amount of time estimated on all the tasks were not that far from the reality.

Another difficult aspect of the estimation of time was the estimation of implementation and testing. Since the group tested the implementation while actually implementing it, it was hard to track how much time was spent on writing code, and how much time was spent testing it, which was one of the reasons why the team decided to put the implementation and testing together as one item.

The unexpectedness of illness and absence also played a part in the team's estimation versus the team's actual hours spent. Naturally, the estimation was based on none of the team members being ill or absent. This is of course unrealistic, even in smaller and bigger projects than this one. The impact of the hours worked were not that large, since it meant that the team members that were available to work needed to take on some of the tasks of the team members that were absent. This is not the optimal way of allocating the work, but nevertheless needed to be done in order for the group to finish the project.

Initially, the team planned to have three sprints with a duration of three weeks each. Just before the beginning of sprint 1, the team felt that it was unnecessary to have such a long duration on the sprint, so the team decided to change the duration of the first sprint to two weeks. The team ended up with four sprints in total. The first, and the third sprint lasting for two weeks, the second lasted for three weeks, and the fourth lasted for 1,5 weeks.

16.2.d Risk evaluation

In the preliminary study phase the team looked at several internal and external risks with the project. While the team did not encounter many internal problems, there were several external risks which described problems encountered during the project. Changes in the specification were made and software updates created some problems for the team.

Since the team's project was fairly easy in the beginning there needed to be an extension to the workload, which made hours worked in the project a considerable amount higher. Even though this presented itself as increase in working hours, it worked out well in context with the estimated total hours. In addition to the change in the specifications of the project, there were updates in the syntax in meteoric which deprecated some of the functions the team used. Luckily the time to replace the written code with the new syntax did not take an large amount of time. Therefore, none of the external risks had a major consequence when looking at the nearly project itself.

The biggest influence on the project from the internal problems were unexpected prolonged illness, and members of the team traveling relatively much. Having said that, the other members in the team were flexible and was able to work extra, and therefore, to some extent, replacing the members which were sick or away. Lack of knowledge about the different technologies we used were also encountered, but since the team expected this particular problem, by estimating hours for it,

it was solve easily. One could say a bad choice in framework was done, because the syntax were changed, and since the framework is in the early stages of development. Nevertheless an decent amount of time were spent to research and test this framework, which made the team conclude with using it.

16.2.e Suggestions for course improvements

Overall, this course has been very interesting, but the team still wants to share some thoughts about what could be improved for the next years to come.

- The compendium should be updated with the correct dates and information.
- The lecture hours should have been at a fixed time each week, as it would help the team to plan better.
- The notification of the lecture dates should be sent out earlier.
- The lectures should not be mandatory. Some of the lectures that were held have been very similar to lectures in previous courses.
- The lectures should have breaks every 45 minutes, and could also be shorter in length.
- The advisor should at a minimum be present at NTNU for the weekly meetings, and not just via skype.

16.2.f What the future customer-driven course students should know about the course

- The course has very educational, so make sure you participate in every aspect of the project.
- The course is very time consuming. Plan to do as much work as you can in the beginning of the project, so you will not need to work all day, every day in the last weeks.
- It is important to have a good relationship with the customer. Make sure you update the customer of the status of the project as you go along.
- Team work is important. Make sure to set aside some time to get to know the team, so you will get a good working environment.
- Be sure to use the advisor's help as much as possible, and do not be afraid to ask for help.

Appendices

A Acronyms

ACE Aid to Capacity Evaluation.

CDP Customer Driven Project.

EQS Electronic Quality System.

ER Emergency Room.

NTNU Norwegian University of Science and Technology.

QA Quality Assurance.

UX User Experience.

WBS Work breakdown Structure.

A | Glossary

Administrator
A user with extended privileges compared to a normal user.

Android
An operating system for mobile devices developed by Google.

API
Functions, procedures, methods and classes used by applications to request services from an operating system.

Application
A computer program designed to do a set of creating tasks.

Compile
Produce by assembling information collected from other sources.

Database
An organized collection of elements.

Flowchart
A type of diagram that represents an algorithm, workflow or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows.

Framework
Often a layered structure indicating what kind of programs can or should be built and how they would interrelate.

Hardware
The physical parts or components of a computer, like the hard disk, monitor, or processor.

iOS
An operating system for mobile devices developed by Apple.

Platform
A defined set of software or hardware that programs can run on.

Rigid methodology
A development methodology which is not flexible.

Software
A set of instructions that directs a computer to perform specific operations.

Stakeholder
A person, group or organization that has interest or concern in an application, or other matters.

Web browser
A software application for retrieving, presenting, and traversing information resources on the World Wide Web, like Google Chrome, Safari, Opera, Firefox, and Internet Explorer.

Wifi
Technology that allows electronic devices to connect to the network.

B | Templates

B.1 Group Meeting

Date: *ex. 15.09.2015*

Name	Present
Sarah Serussi	
Morten Kartevoll	
Eirik Jensen	
Are Haartveit	
Petter Johansen	
Thor Martin Abrahamsen	

Agenda:

- Theme 1
  - Comments
  - Comments
- Theme 2
  - Comments
  - …
- …

TODOs

- Morten
  - Notes
- Are
  - Notes
- Eirik
  - Notes
- Petter
  - Notes
- Sarah
  - Notes
- Thor Martin
  - Notes

Other Notes:

Figure B.1: Group Meeting Template

B.2 Advisor Meeting

Date: *ex. 15.09.2015*

Name	Present
Sarah Serussi	
Morten Kartevoll	
Eirik Jensen	
Are Haartveit	
Petter Johansen	
Thor Martin Abrahamsen	

Questions to customer:

- Questions 1
  - Answer
- Questions 2
  - Answer
- …
  - ...

Other Notes:

Figure B.2: Advisor Meeting Template

B.3 Customer Meeting

Date: *ex. 15.09.2015*

Name	Present
Sarah Serussi	
Morten Kartevoll	
Eirik Jensen	
Are Haartveit	
Petter Johansen	
Thor Martin Abrahamsen	

Questions to customer:

- Questions 1
  - Answer
- Questions 2
  - Answer
- …
  - ...

Other Notes:

Figure B.3: Customer Meeting Template

C | Installation Guide

This installation guide describes how one can deploy the application from a Mac to an Android or iOS phone. The application is also available online at 'http://tiaco.meteor.com'. Unfortunately, the application cannot be deployed from a Windows computer. This is due to the fact that Meteor does not support it yet.

Step 1
Clone the repository from 'https://github.com/tiaco-it/ER-assist' .

Step 2
Download software to be able to run Meteor.
Follow the directions from these pages if you are deploying to an Android device:
'https://github.com/meteor/meteor/wiki/Mobile-Development-Install:-Android-on-Mac'
'https://github.com/meteor/meteor/wiki/How-to-run-your-app-on-an-Android-device'
Follow the directions from this page if you are deploying to an Apple device:
'https://github.com/meteor/meteor/wiki/How-to-run-your-app-on-an-iOS-device'

Step 3
If you are deploying to an Android device, make sure the developer mode is enabled.
Follow the directions from this page:
'http://www.greenbot.com/article/2457986/how-to-enable-developer-options-on-your-android-phone-or-tablet.html'

Step 4
Open the terminal/command prompt, locate the /app folder in the /ER-assist repository and type in the following to deploy to an iPhone:

```
meteor deploy tiaco.meteor.com
meteor run ios-device --mobile-server tiaco.meteor.com
```

This will package the application as an **Xcode**-project and the user will need to run the application from within Xcode.

Type in the following to deploy to an Android phone:

```
meteor run android-device --mobile-server tiaco.meteor.com
```

D | System Usability Scale form

Participant: _____ Product: _____ Date: _____

System Usability Scale

Instructions: For each of the following statements, mark one box that best describes your reactions to the application today. ☑

		Strongly Disagree				Strongly Agree
1.	I think that I would like to use this app frequently.	☐	☐	☐	☐	☐
2.	I found this application unnecessarily complex.	☐	☐	☐	☐	☐
3.	I thought this application was easy to use.	☐	☐	☐	☐	☐
4.	I think that I would need assistance to be able to use this application.	☐	☐	☐	☐	☐
5.	I found the various functions in this application were well integrated.	☐	☐	☐	☐	☐
6.	I thought there was too much inconsistency in this application.	☐	☐	☐	☐	☐
7.	I would imagine that most people would learn to use this application very quickly.	☐	☐	☐	☐	☐
8.	I found this application very cumbersome/awkward to use.	☐	☐	☐	☐	☐
9.	I felt very confident using this application.	☐	☐	☐	☐	☐
10.	I needed to learn a lot of things before I could get going with this application.	☐	☐	☐	☐	☐

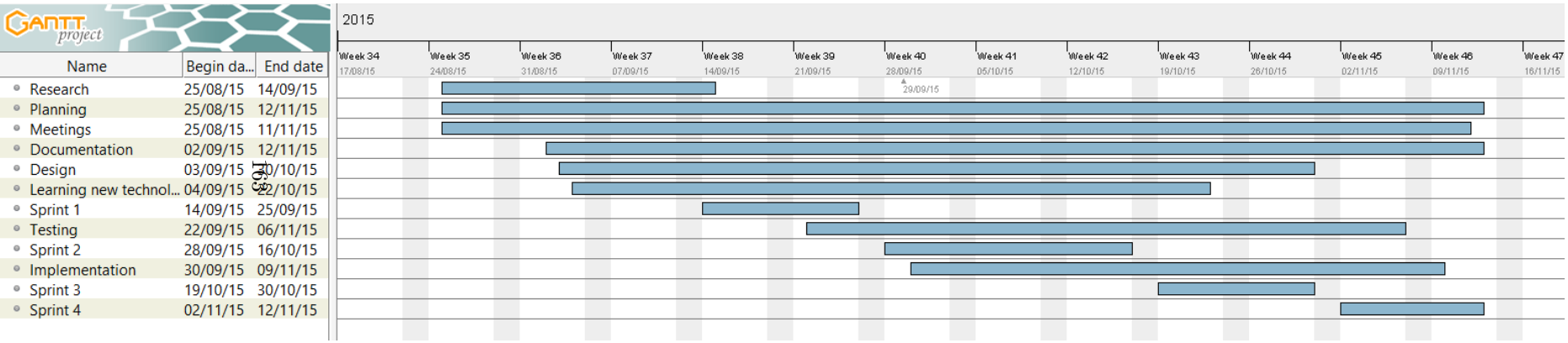
E | Observation form

Observation Form - usability test

Observer:	_____	Date/Time:	_____
Test product:	_____	Test leader:	_____
Test subject:	_____	Age/Gender/Other:	_____

Time	Problem	Cause	Suggestion for solution
?	?	?	?
?	?	?	?

F | Gantt diagram


GANTT project
2015
Name
Begin da...
End date
Research 25/08/15 14/09/15
Planning 25/08/15 12/11/15
Meetings 25/08/15 11/11/15
Documentation 02/09/15 12/11/15
Design 03/09/15 30/10/15
Learning new technol... 04/09/15 22/10/15
Sprint 1 14/09/15 25/09/15
Testing 22/09/15 06/11/15
Sprint 2 28/09/15 16/10/15
Implementation 30/09/15 09/11/15
Sprint 3 19/10/15 30/10/15
Sprint 4 02/11/15 12/11/15
Week 34 17/08/15
Week 35 24/08/15
Week 36 31/08/15
Week 37 07/09/15
Week 38 14/09/15
Week 39 21/09/15
Week 40 28/09/15
29/09/15
Week 41 05/10/15
Week 42 12/10/15
Week 43 19/10/15
Week 44 26/10/15
Week 45 02/11/15
Week 46 09/11/15
Week 47 16/11/15

G | Rules at Tiaco

Rules are important for a common understanding of the work ethic and the expected amount of performance effort within the team.

G.1 Group rules

1. If there are any disagreements, there will be a democratic vote.
2. Take breaks regularly.
3. Every team member should be punctual. If one member is not able to show, notice will be given.
4. Respect each others' opinion.
5. Do not hesitate to ask for help.
6. Encourage each other and discuss challenges.
7. Be available via e-mail, cell phone, and Facebook.
8. Document the progress and the work that has been done every week.
9. Have weekly meetings with encouraging food.
10. Use the determined code and design conventions and report work hours.

H | Interview with customer

This chapter contains notes from the first meeting and interview with the customer. This meeting was originally held in Norwegian, but is translated to English.

H.1 Date: 25.08.2015

Interviewers: Sarah Serussi, Morten Kartevoll, Eirik Jensen, Are Gotteberg, Thor Martin Abrahamsen, and Petter Bakkan Johansen.
Interviewee: Dr. Florentin Moser.

Please explain which product you want us to make.

The product will be an application that will handle challenging patients and legal issues when dealing with such patients in the ER at the hospital. It is meant as a tool to help medical personnel by making it easier to make decisions regarding difficult patients. Whether they should call the police, admit the patient to the psych ward, or not do anything. It is important that the application will give guidance according to the Norwegian law.

On which platforms do you wish to deploy this application?

The application should work on smart phones, tablets and computers.

Are there any existing systems that are used for this purpose today?

There exists a system that is called EQS. It is a huge collection of procedures, but the search function is hopeless, and it takes a lot of time to find the relevant laws and documents needed in specific situations.

How do you use this system today (EQS)?

You need to log in to a stationary computer with your access card, which can take up to five minutes. After that you need to open the software, type in username and password before you can access the information. This takes too much time, and after you have logged in it can take up to 15 minutes to find the relevant documents or procedures that you are looking for.

How fast do you want the application to be?

The user of the application should find the relevant law/paragraph within one minute.

Is the Internet connection stable at the ER?

No, the connection varies from day to day. Sometimes it can be hard to connect to the Internet, so the app should work offline.

Who is this application meant for?

The application is meant for doctors that works in the ER, especially the ones that are under 50 years old.

Have you been involved in app development before?

Yes. It's called VTE. This is an app that is used for assessment, diagnostics, and treatment of the disease Venous Thromboembolism, which is used by medical personnel only.

I | Code

I.1 Code Structure

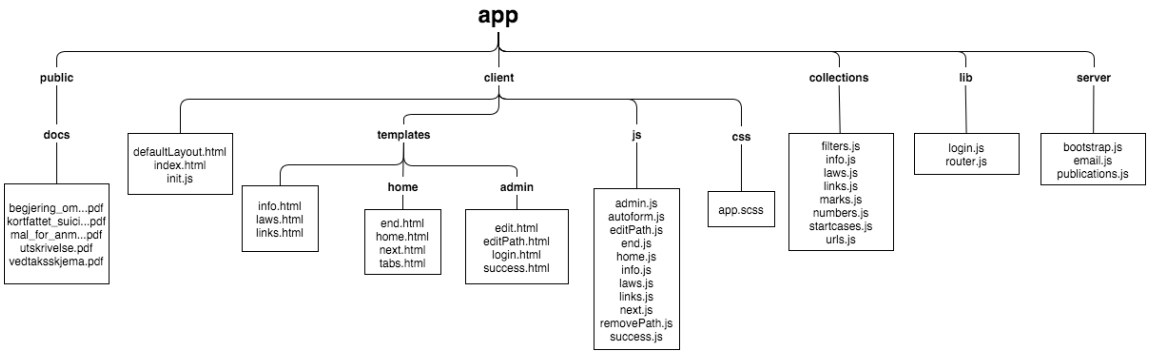


Figure I.1: File structure in the application

I.2 Code Collections

The following figure presents the collections defined in this application. The top box denotes the collection name, the middle is the different fields, and the bottom box shows the collection methods.

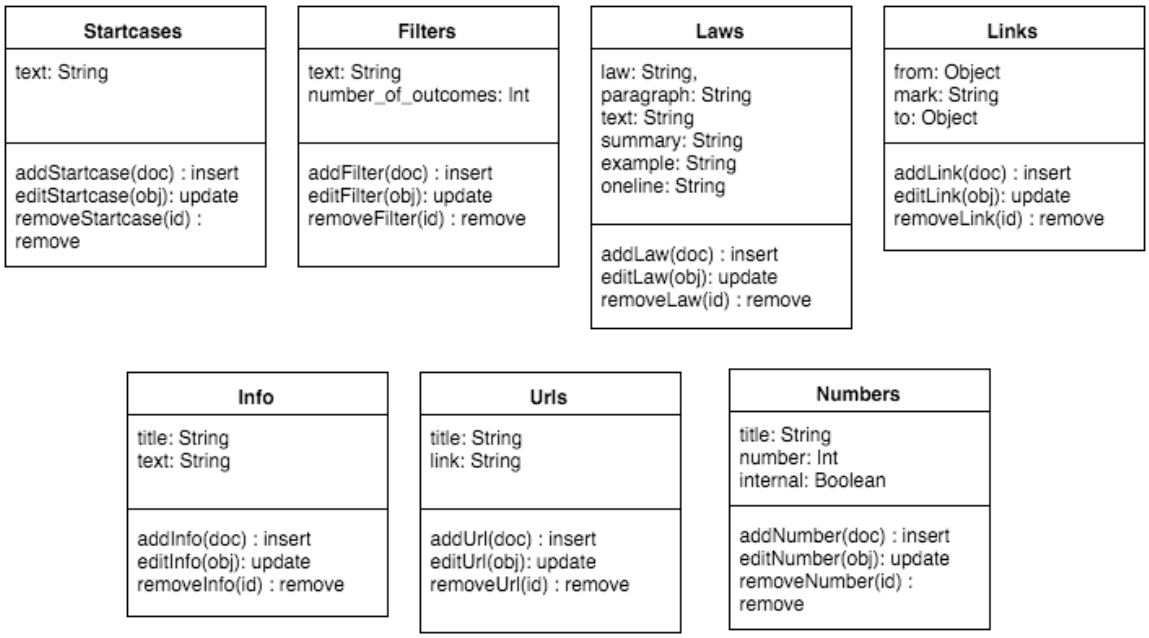


Figure I.2: Collections defined in the application

I.3 Code Templates

The following figure presents the templates defined under the `client` folder. The outer container refers to a folder, the inner container refers to the file name and the content is the template names.

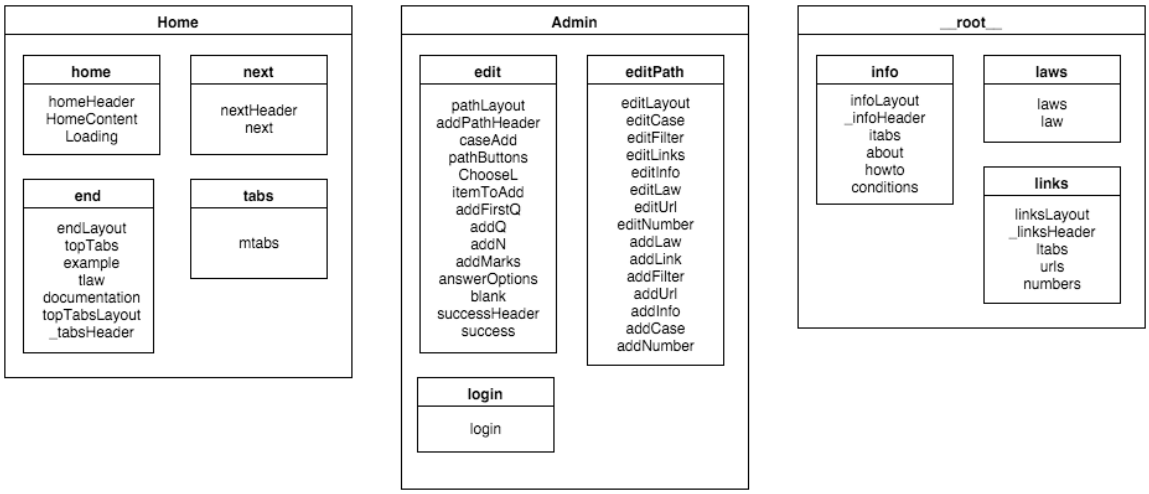


Figure I.3: Templates defined in the application

I.4 Code Helpers

The following figure presents the template helpers defined in this application. The outer container refers to the file name, the inner container refers to the template name and the content is the helper names.

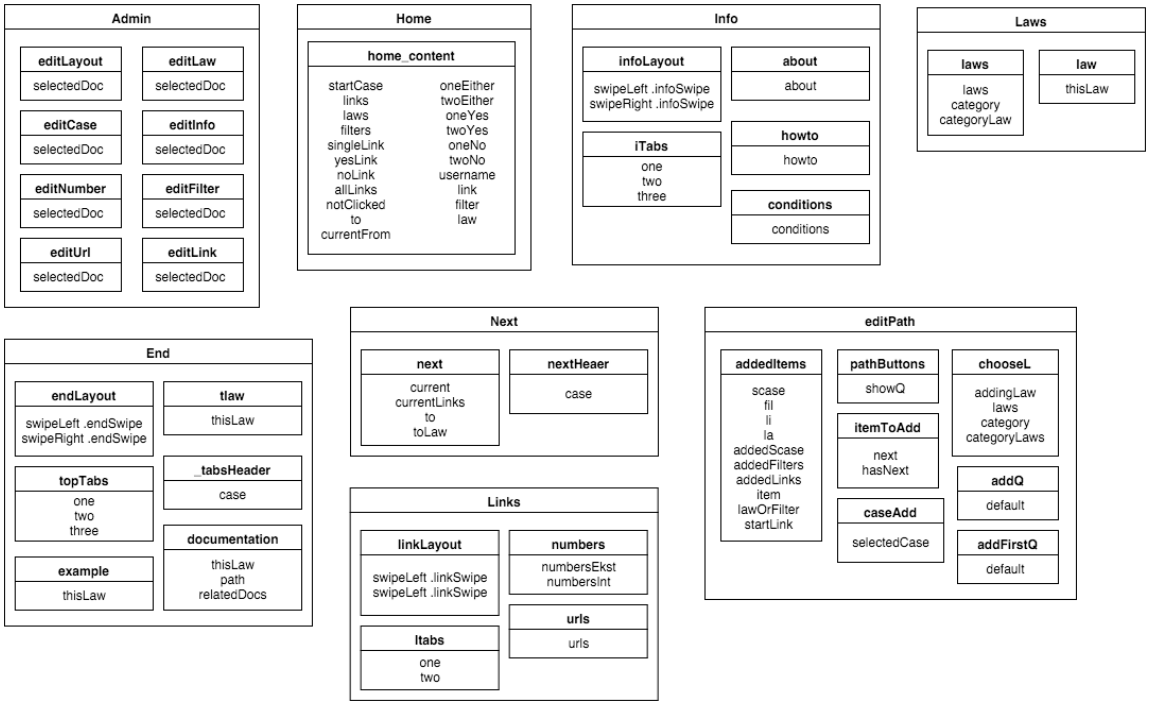


Figure I.4: Helpers defined in the application

I.5 Code Events

The following figure presents the template events defined in this application. The outer container refers to the file name, the inner container refers to the template name and the content is the event names.

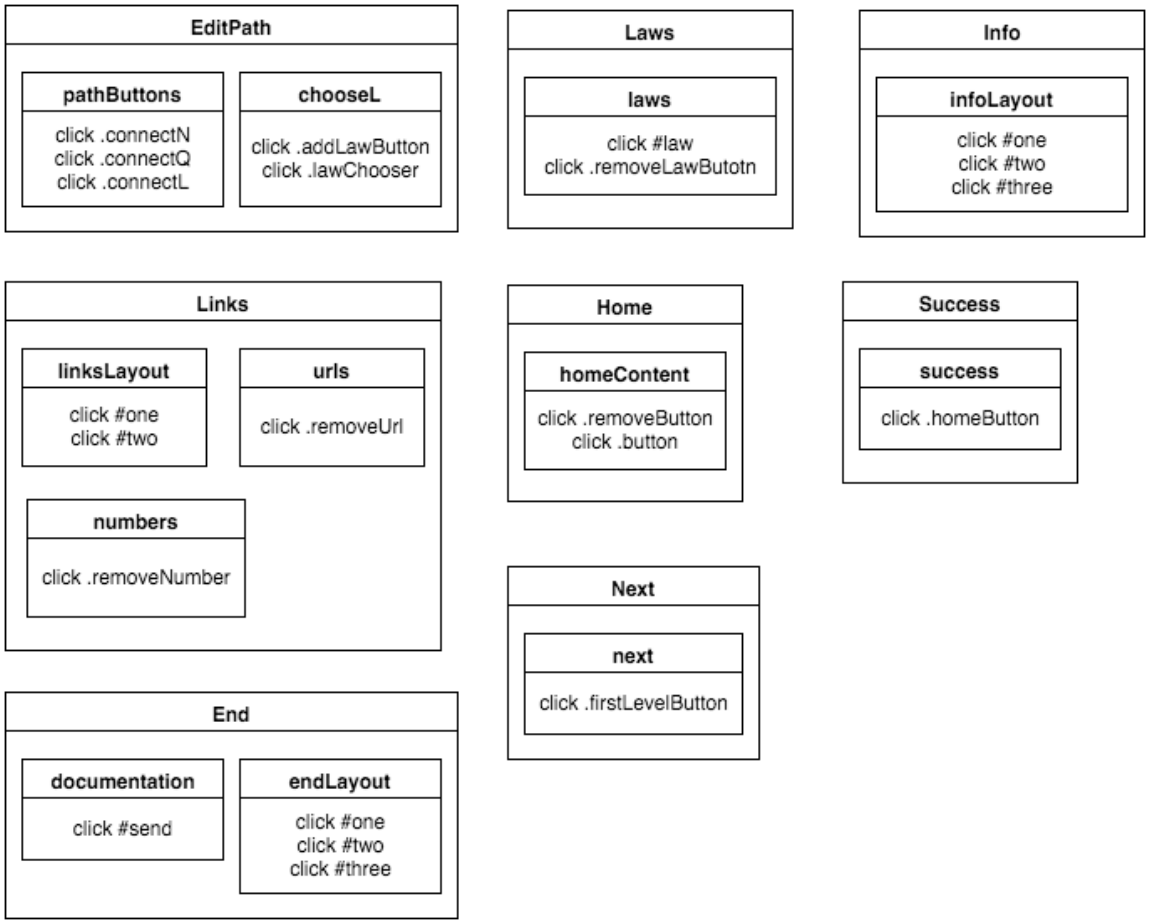


Figure I.5: Events defined in the application

J | User Guide

This appendix is gives a detailed guide of how to use "Akuttjus". In the first section the user view is explained, and in the second the administrator view is described. One thing that is common for both parts is that every view that is not reachable from the bottom navigation system has a return ('Tilbake') button that returns the user to the previous view.

J.1 User

Users without administrator rights only get to see part of the application, as all the administrator rights is reserved for the administrators. This part of the guide takes the reader step by step through the application and explains what each button on the screen does.

J.1.a Home view

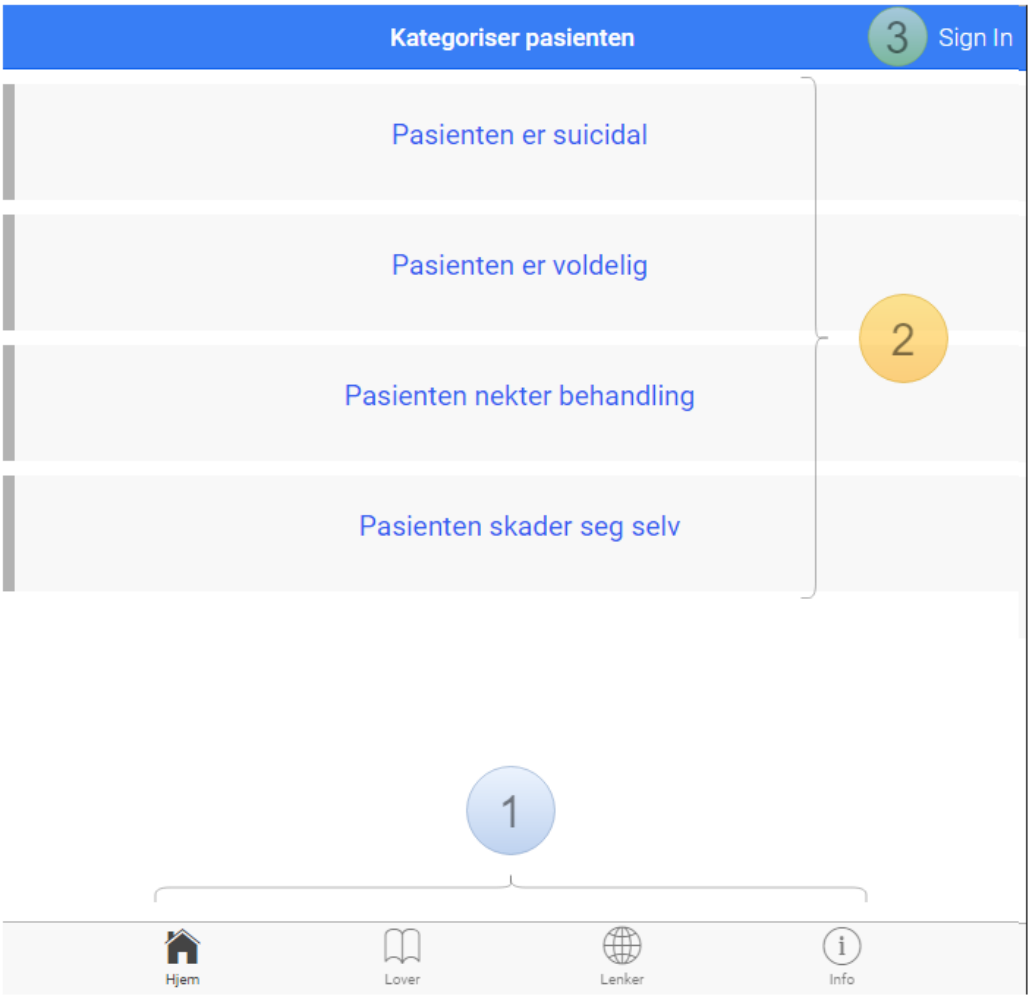


Figure J.1: Front page

The first view that the user accesses is the front page, or home view. The front page gives the user three different options. The buttons at the bottom, marked by the number 1, is used to navigate between the four main views. The functionality of each is described in more detail in later sections. These navigation tabs are presented in all views and allows the user access any main view from anywhere inside the application. As displayed in figure J.1 the home (hjem) icon is selected, as indicated by the black fill of the icon.

The buttons marked by the number two 2 are called categories. These are the main starting point on the path towards finding the correct law and information regarding patients which fall

inside this category. The application assumes that the user, i.e. the doctor, has already classified the patient and the application provides no information about how to classify a patient except for a few examples of patient groups that always fit to a category. The category buttons work in three different ways, the direct type leads directly to the tabular view, which is described in section J.1.b, which contains examples, laws, and documentations that is related to the category.

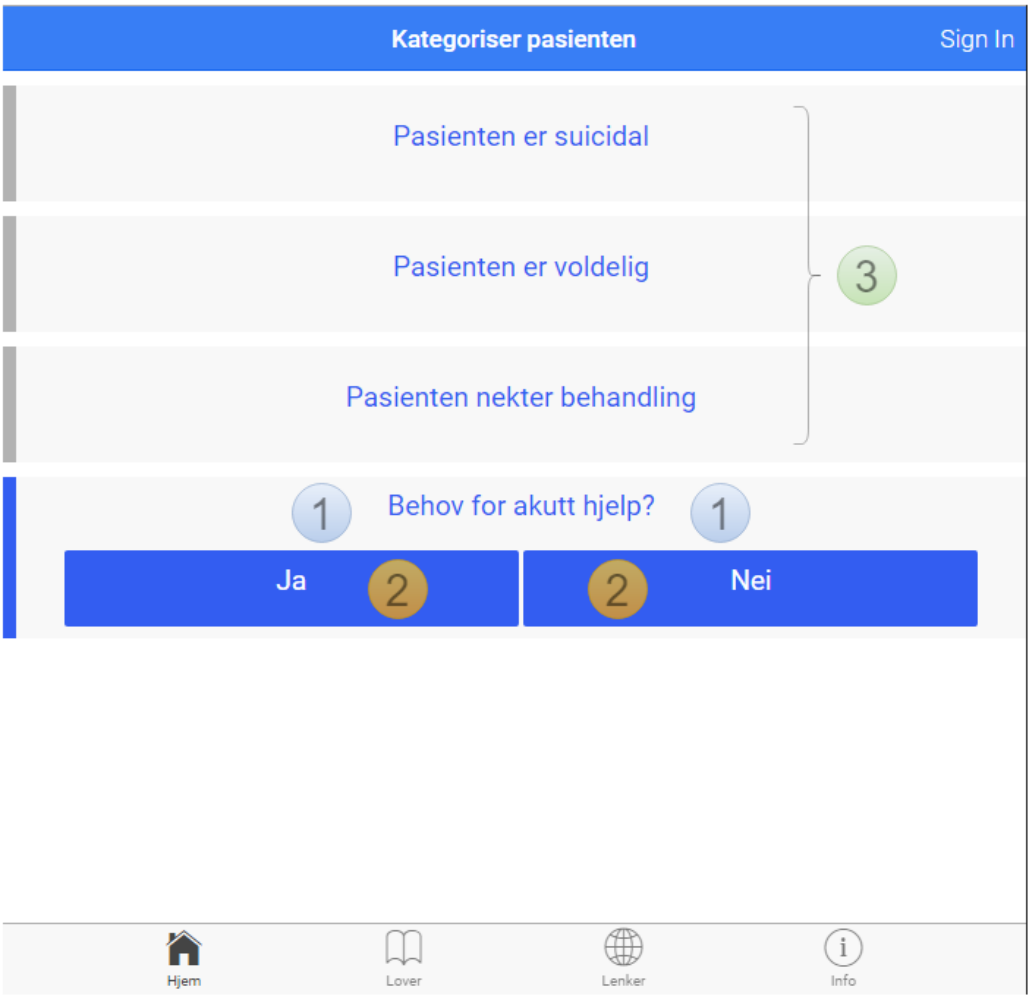


Figure J.2: The yes/no category type of categories

The binary type, also called yes/no type, prompts the user with a question that can be answered with either yes or no. By pressing the yes or no button, illustrated in figure J.2, one of the following three things will happen: the user is taken directly to a tabular view, similarly to what happens when the user presses a direct category; the user is prompted another yes or no question; or the user is navigated to a view that is similar to the front page, presented in figure J.3 where the user further categorizes the patient based on category chosen from the home view.

The third and final type of buttons are buttons that takes the user to a new view where the user is asked to answer a question that has more than two alternate answers. Each of these answers will take the user to the tabular view associated with the chosen answer.

The sign in button, marked with the number 3 in figure J.1, prompts the user with the option to sign in. Users that has authentication information, which at this point in time is only the administrators, will have the option to submit their username and password, and from this be able to access the administrator part of the application. How to use the administrator part is described in section J.2.

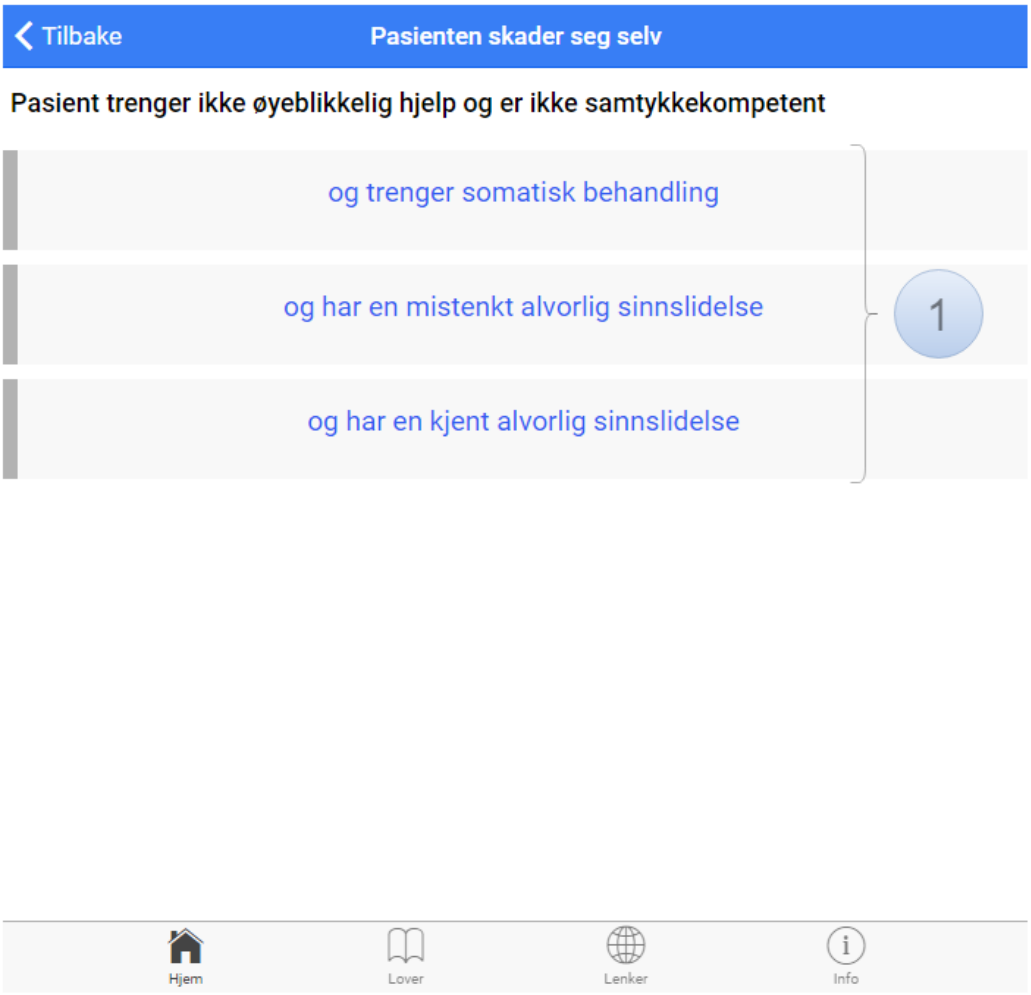


Figure J.3: Example of a view where the user is asked to futher categorize the patient

J.1.b Tabular view

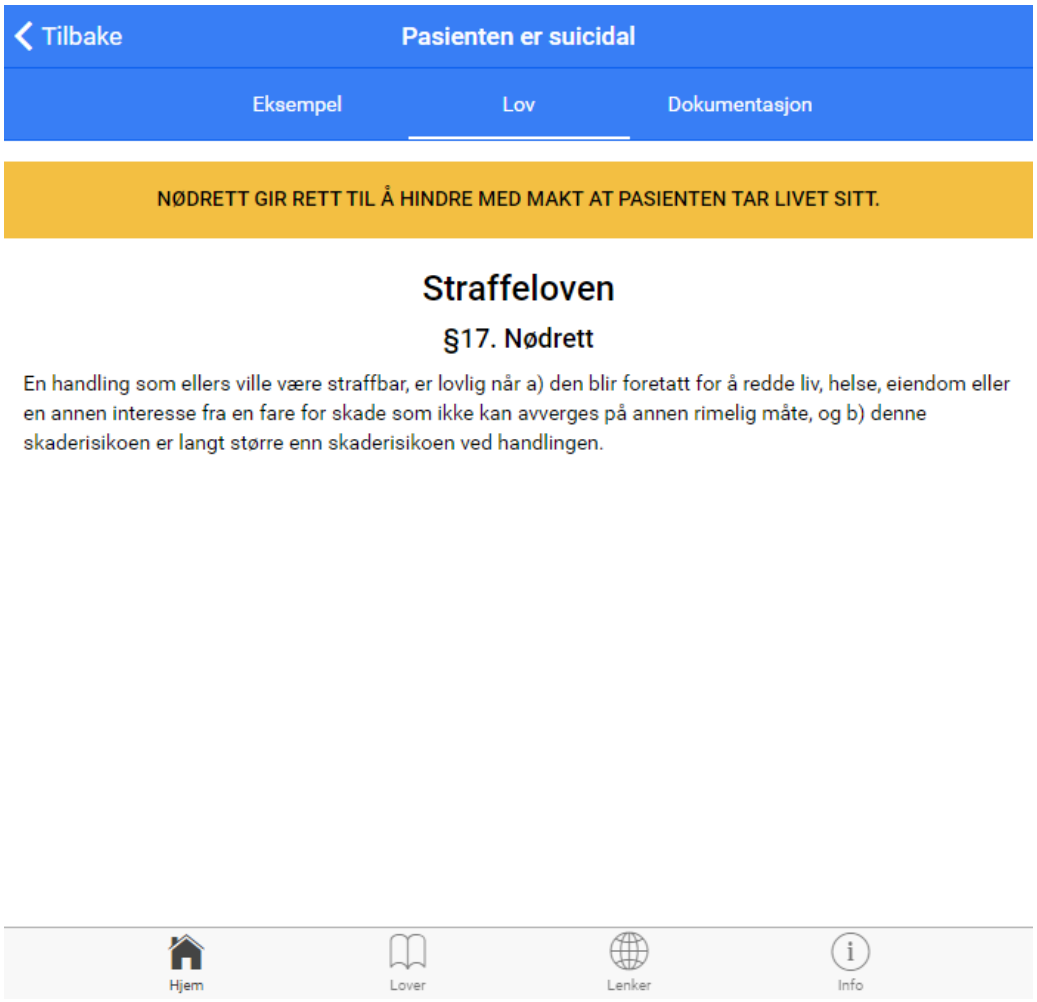


Figure J.4: The tabular view

Once the user has finished categorizing the patient the tabular view with related information is displayed. This view has a tabular design that supports swipe motions in addition to clicks in order to toggle between the three tabs. The example and law views only displays raw text and serve no other purpose than providing information to the user. The documentation view on the other hand contains relevant documents that the user can either open in their web-browser, by pressing the open document (Åpen) button, or to chose the documents he or she wants and send them to their e-mail. The documents which the user wants to send is selected by pressing the name or the circle on the left hand side of the name of the document and then pressing the send button.

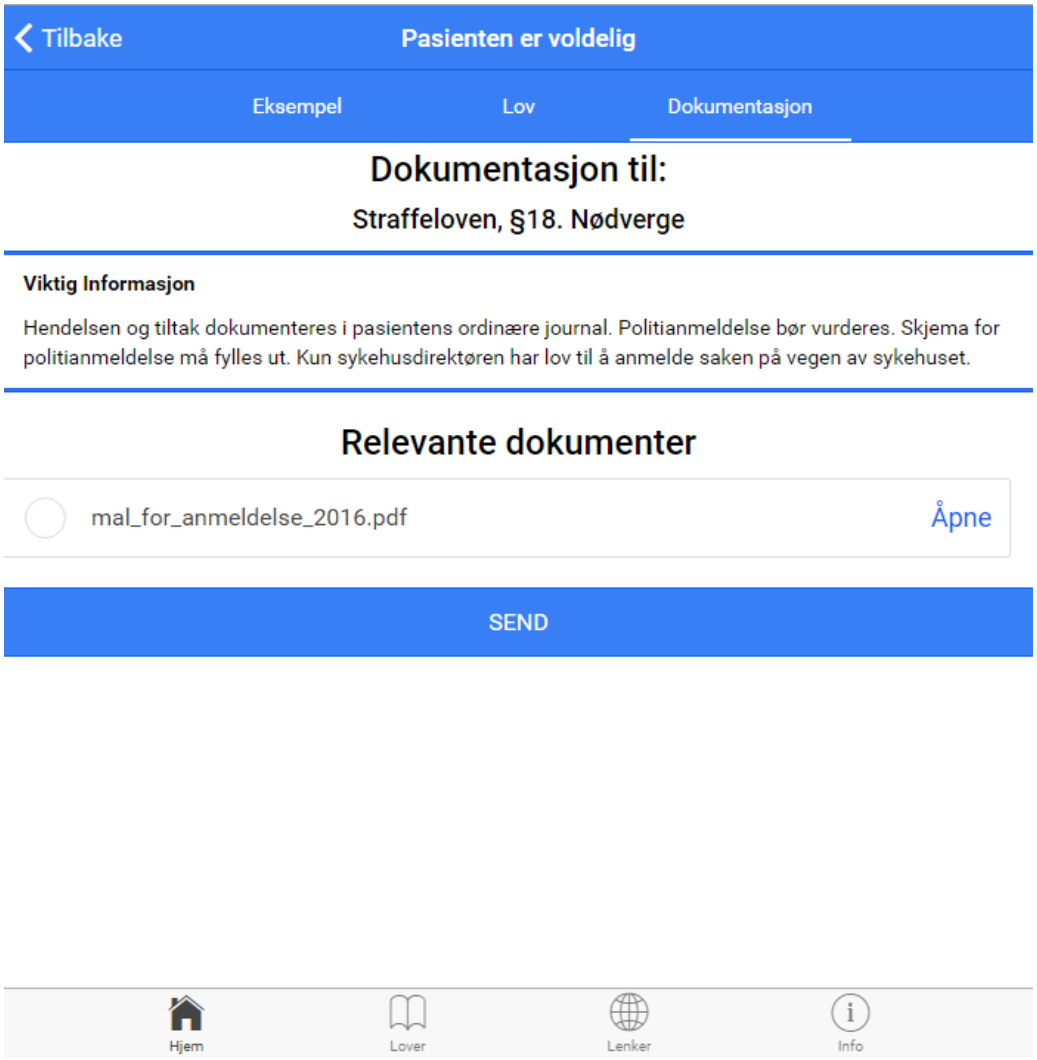


Figure J.5: The document tab in the tabular view

J.1.c Links and numbers view

In figure J.6 the links and numbers view is displayed, as indicated by the black fill of the icon on the bottow navigation menu. The view is organized in the same tabular fashion as the tab view. By either using swiping motions or pressing the buttons next to the number 1 the user can toggle between the links and number tabs. Both the views contain clickable buttons, as indicated by the number 2. In the links view these buttons opens the website linked to in the users web-browser. In the numbers view these buttons initiates a call to the number displayed on the button. This of course is only possible on mobile phones.

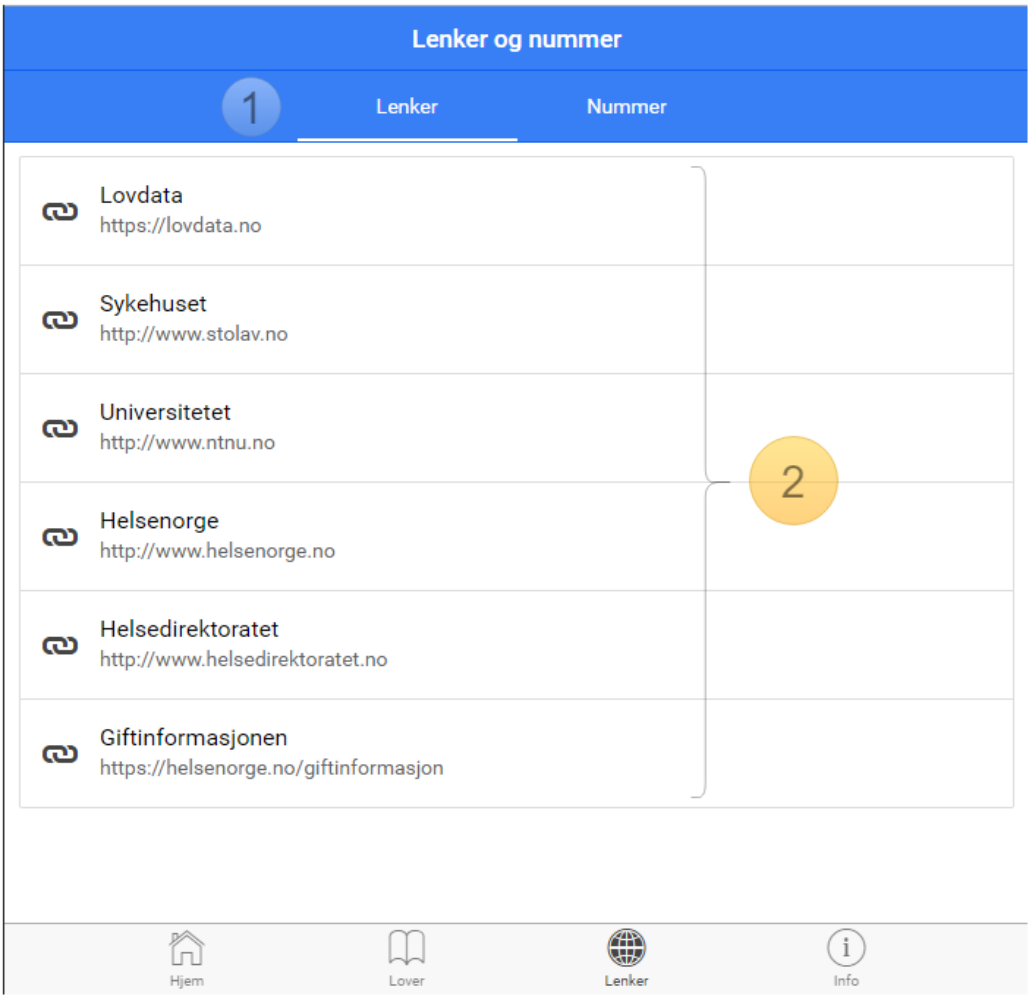


Figure J.6: The links and numbers view

J.1.d Law view

Figure J.7 is presented. The law view contains a list of the laws that used within the application. Each of the entries, marked by the number 1, are clickable and will open the law display view, which is illustrated in figure J.8.

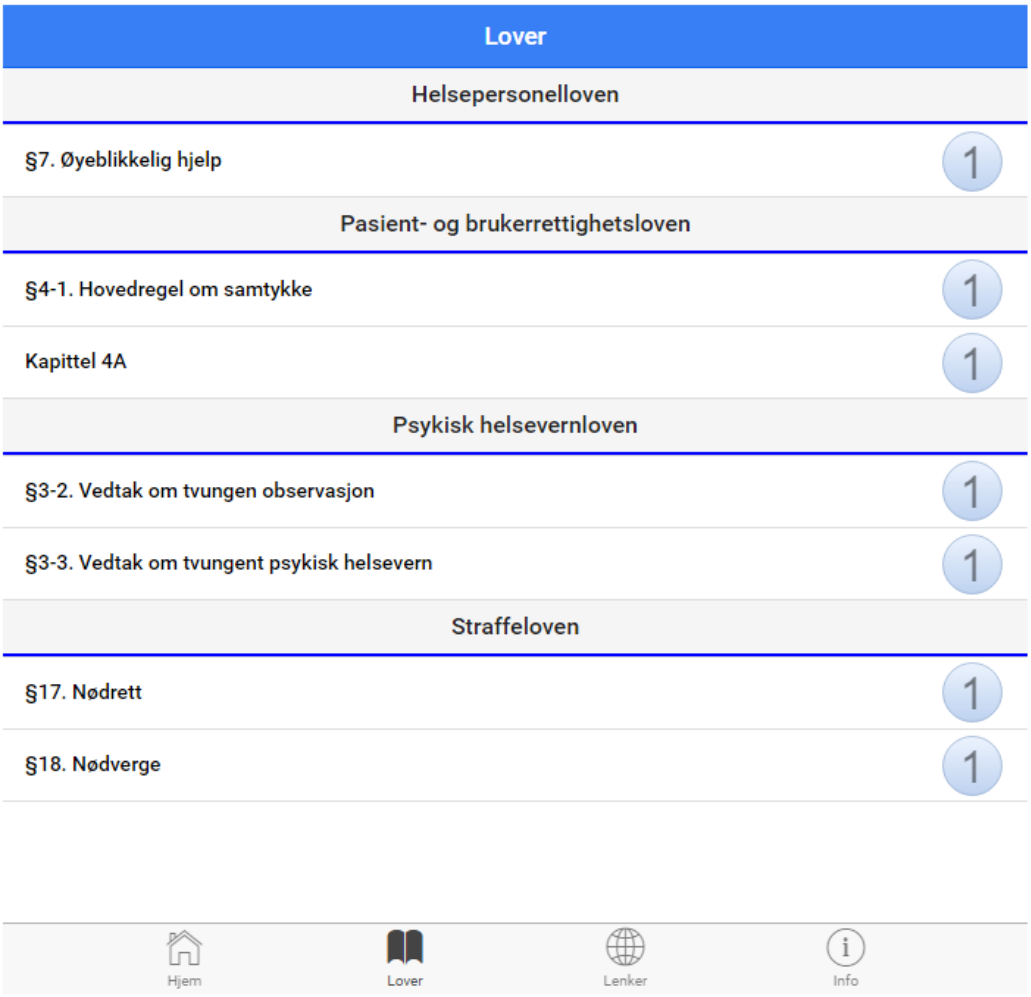


Figure J.7: The law view

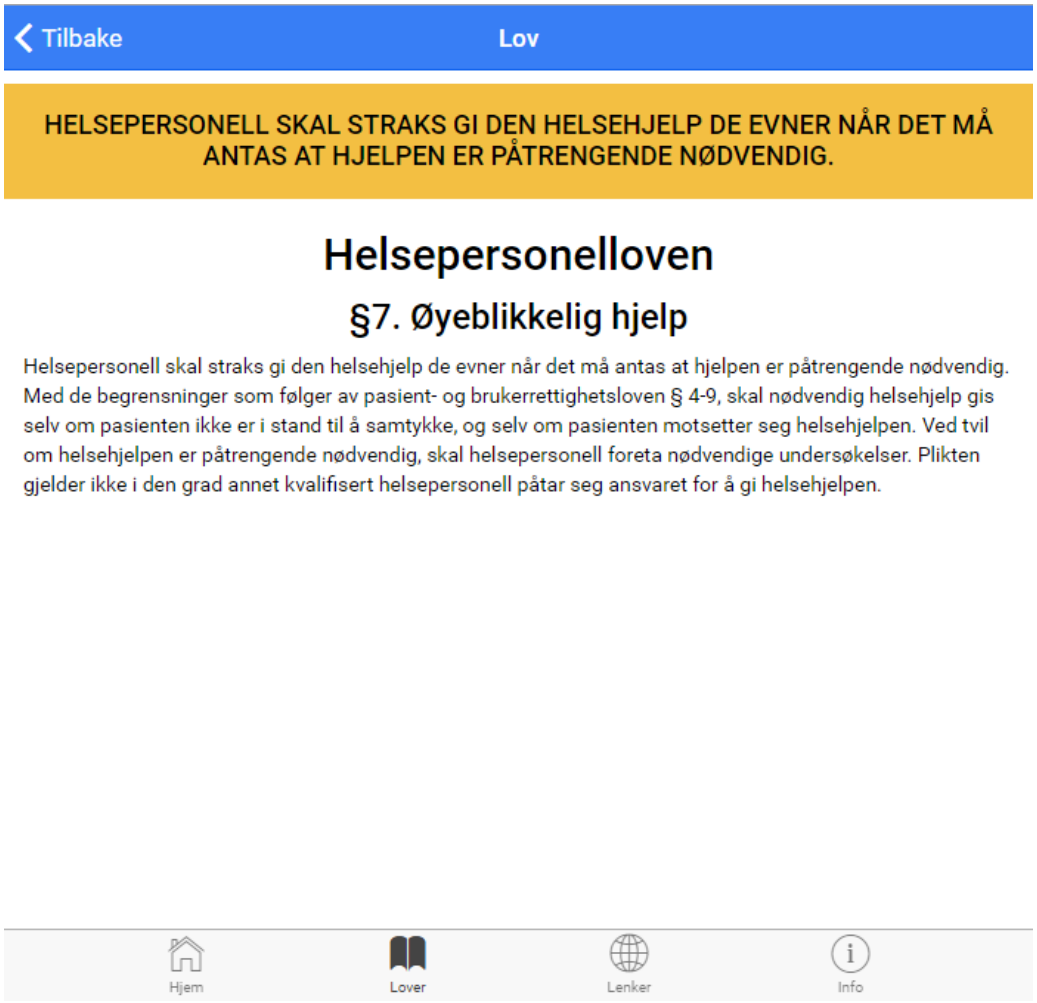


Figure J.8: The view displaying the law information

J.2 Administrator

Administrators are user with extend privileges that allows them add, edit or delete content in the entire application. A lot of the functionality in the administrator part of the application is quite complicated and this user guide will try to make the customization of the application a bit more comprehensible. It is worth noting that the application behaves in the exact same way as described in the user section, which allows the administrator to utilize the application in the same way as a regular user when signed in.

J.3 Home view

The first view that is displayed to the administrator after signing in is the home view. This view has two notable changes and the functionality of each of these changes will be described.

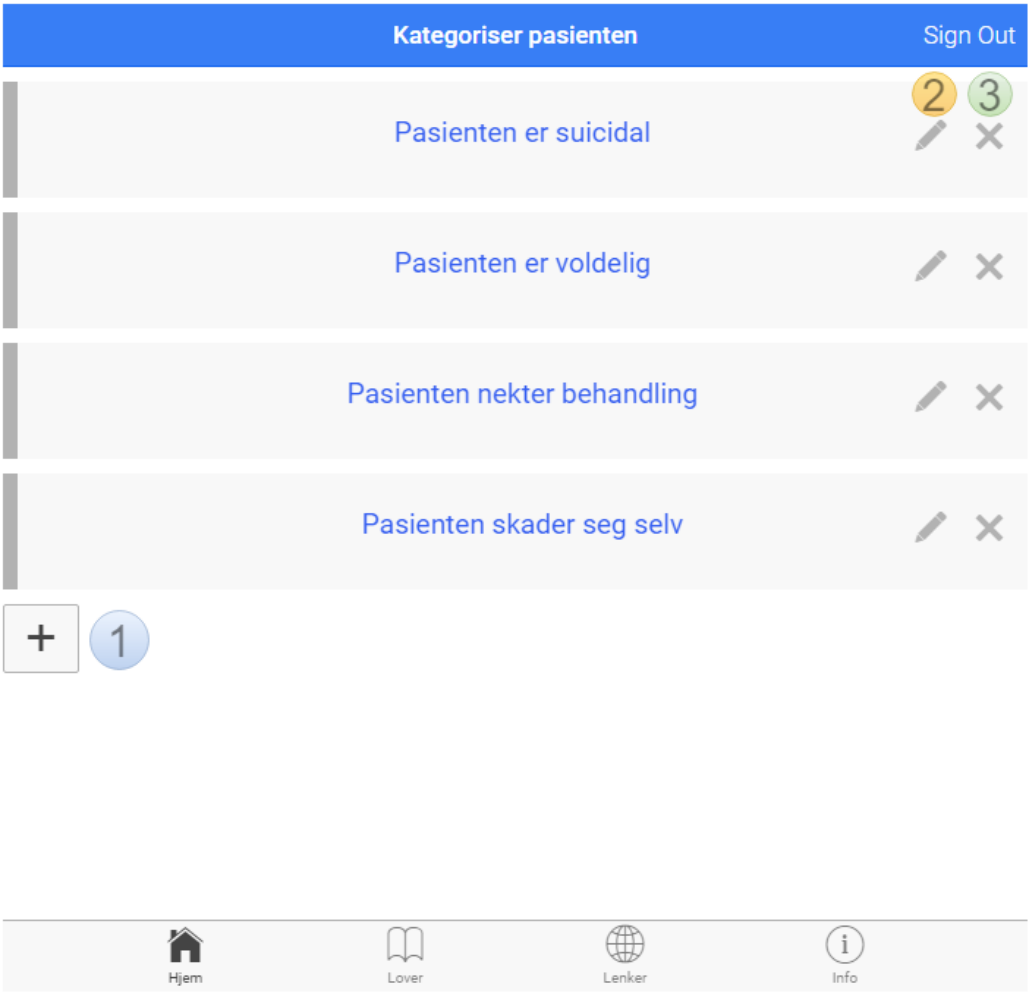


Figure J.9: The front page

Firstly there is a large plus icon added on the left hand side of the screen, marked by the number 1. By pressing this button the administrator initiates the functionality for adding an entirely new category to the application, which presents the user the view displayed in figure J.15 in section J.3.d. This section also describes the process of adding a new category.

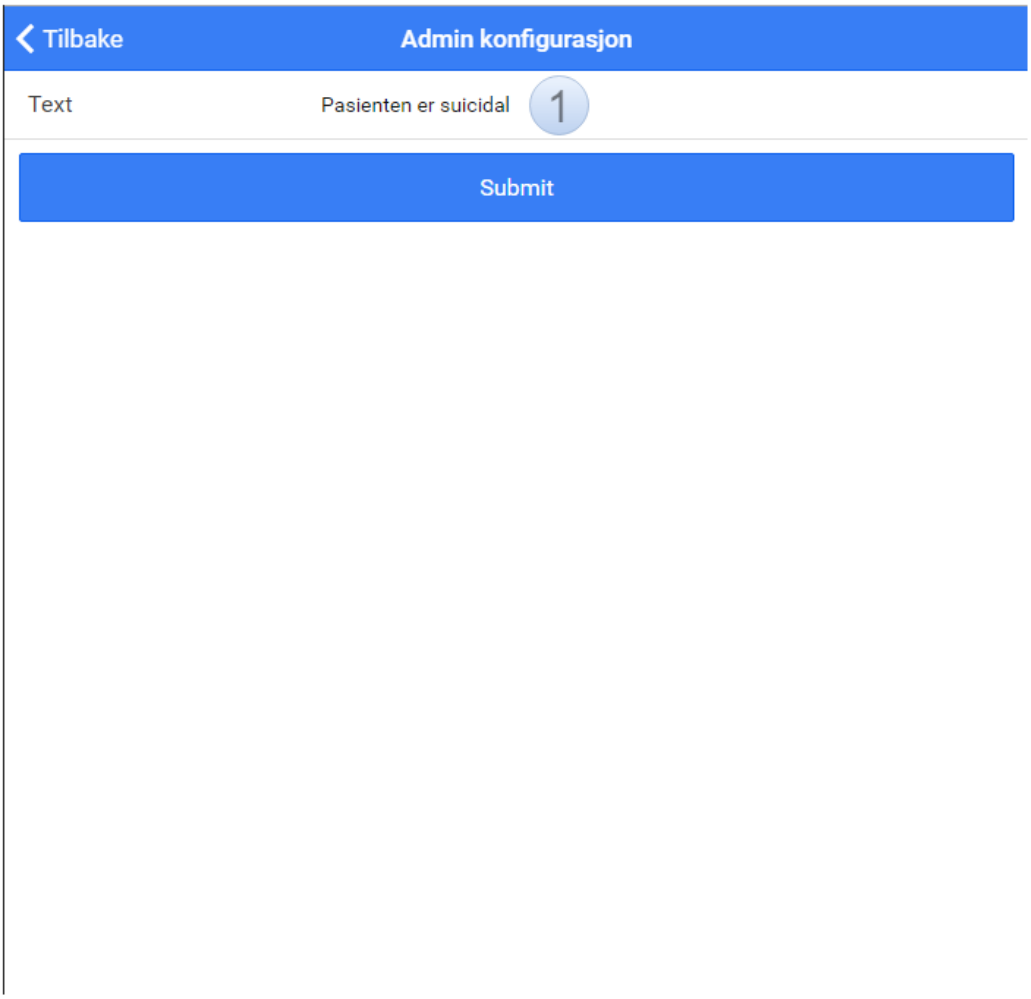


Figure J.10: The user is able to change the name of the category

Secondly there is the edit and delete icons on each of the categories, displayed by the number 2 and 3 in figure J.9. By pressing the edit icon on a category the user is displayed the view displayed in figure where the text of the category can be edited. If the delete button on the front page is pressed the user is prompted with a small box, as presented in figure J.11, which asks the user to confirm the deletion of the category.

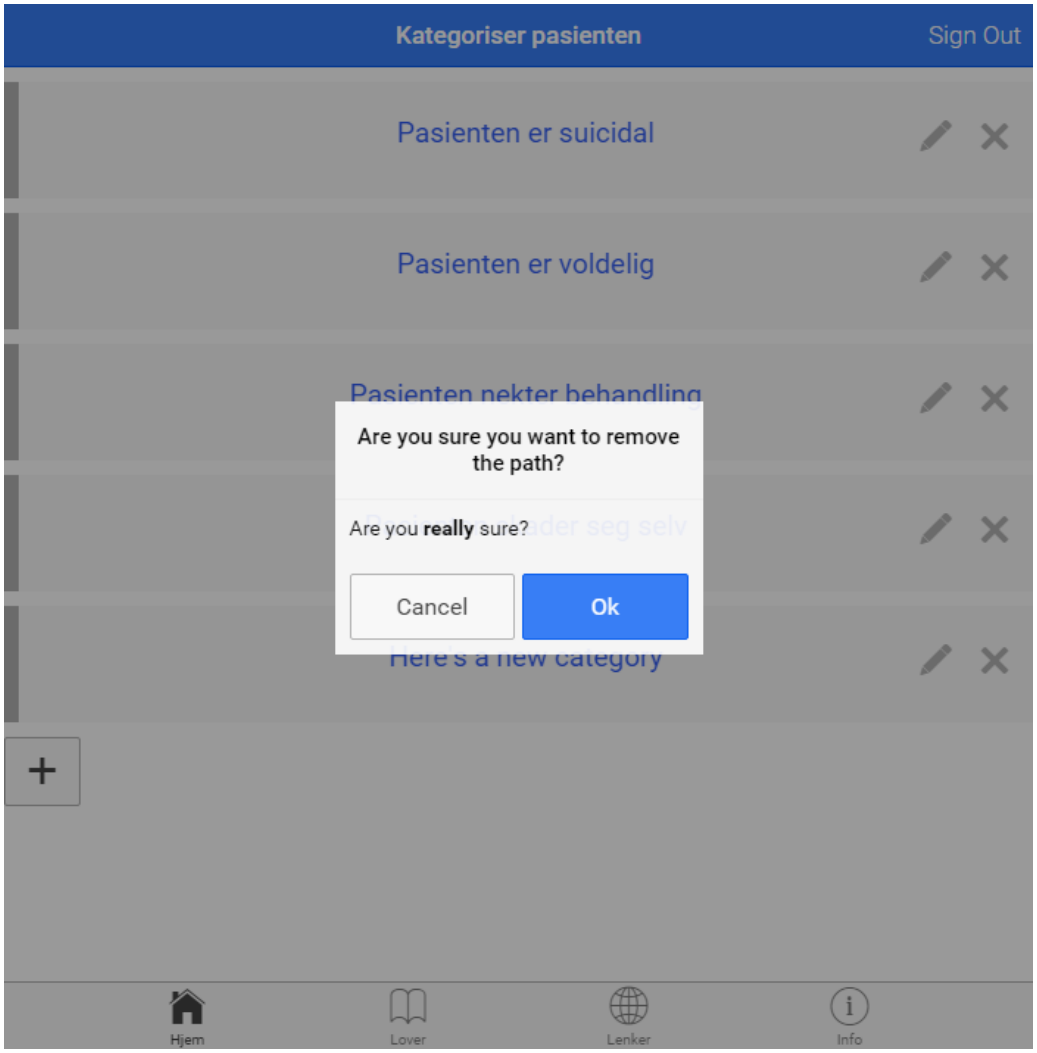


Figure J.11: Confirm delete prompt

J.3.a Links and numbers view

The links and numbers view is nearly the same in the administrator part of the system as it is in the user part of the system displayed in figure J.12. The differences is the add button, pluss, on the bottom of the page, displayed by number 1, and the edit and delete buttons in their respective links, and numbers. When pressing the add button, the administrator will be navigated to a new page, where what title the link should have, and what the actual link is. The same goes for numbers.

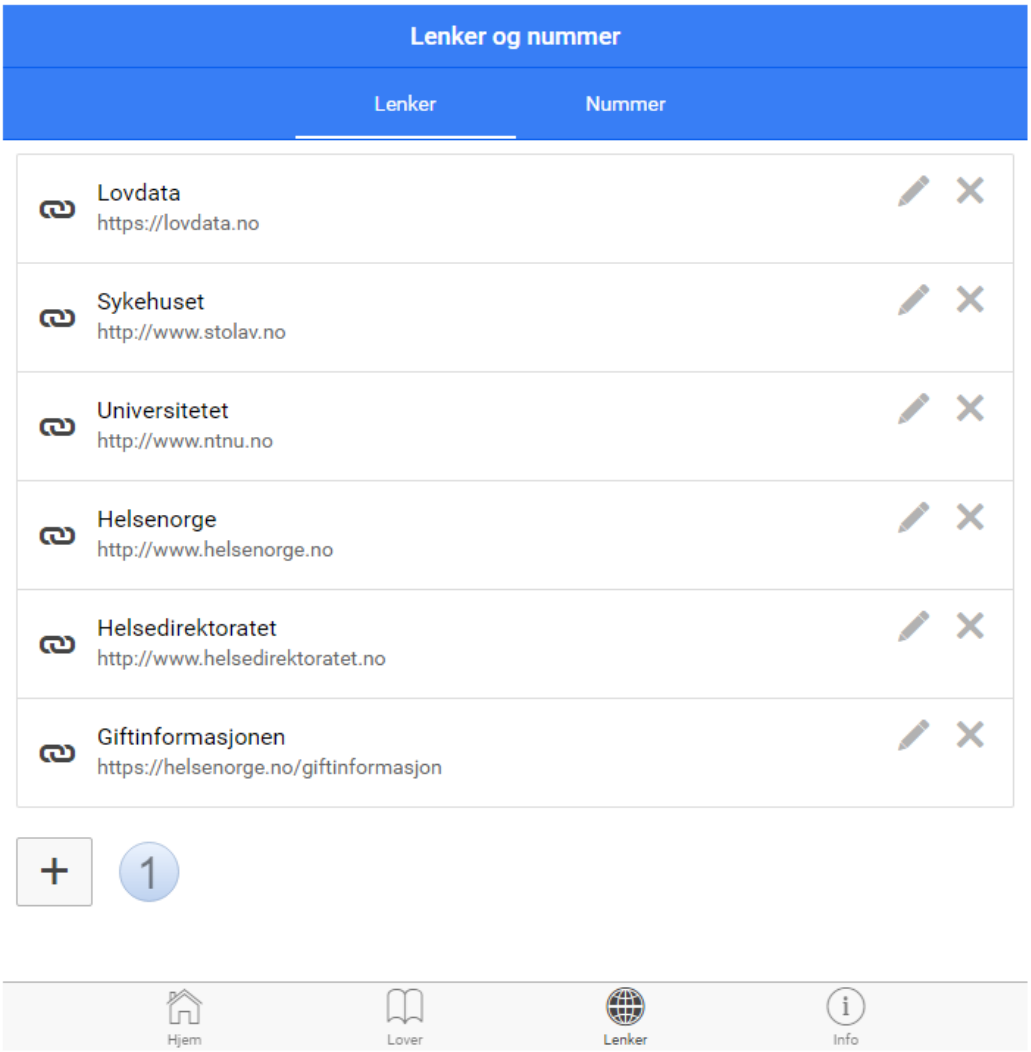


Figure J.12: The links and numbers view in the administrator part of the system

J.3.b Law view

The edit law view follows the same consept as the links and numbers view, presented in figure J.12, in the administrator part of the system. There is a add button on the bottom of the page, displayed as number 1 in figure J.13. Edit and delete is, respectively, represented as 2 and 3 in the same figure. The edit button will allow the administrator to change the content in the law, example, and documentation, and the delete button allows the administrator to delete a law.

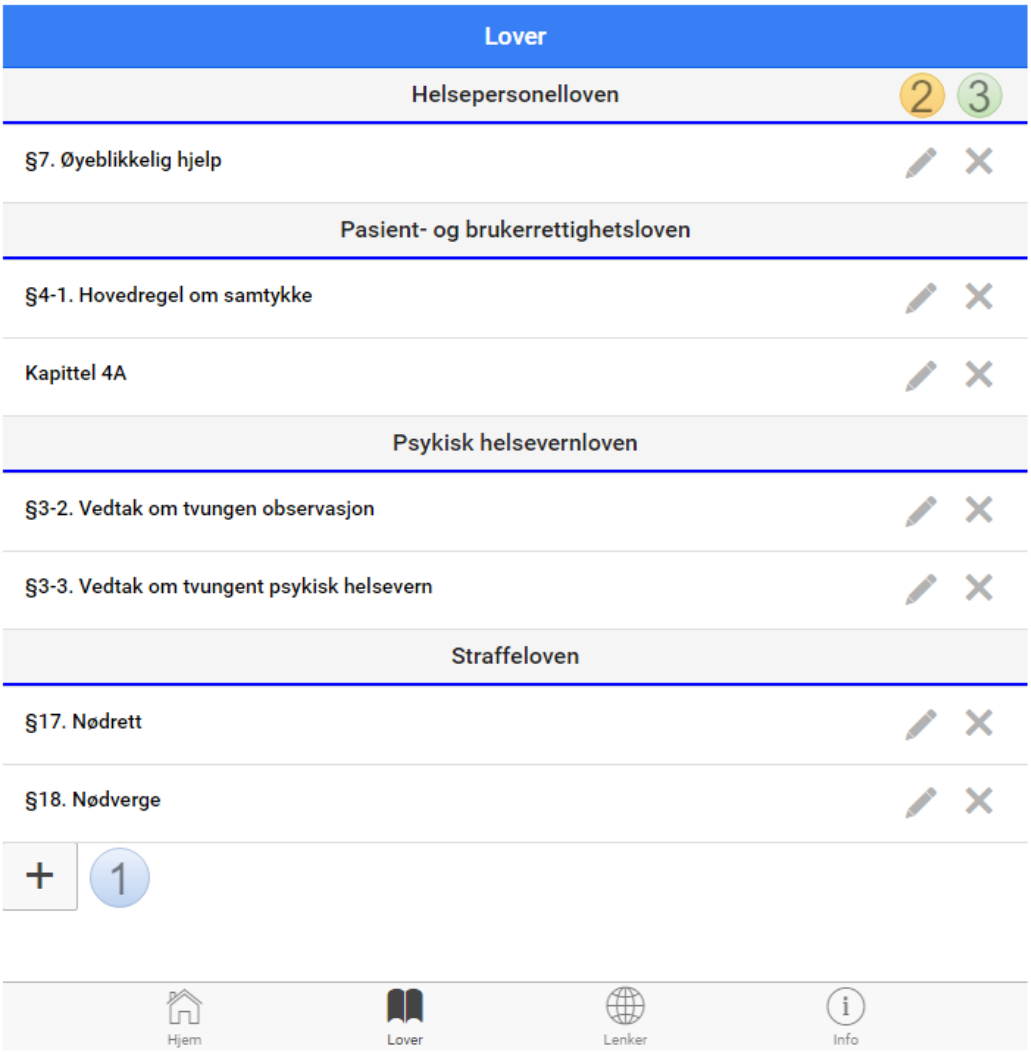


Figure J.13: The law view in the administrator part of the system

J.3.c Information view

In the information view, displayed in figure J.14 the administrator is only allowed to change the content in the different tabs, and not allowed to delete tabs. If he or she wants to delete tabs there most be a configuration in the code.

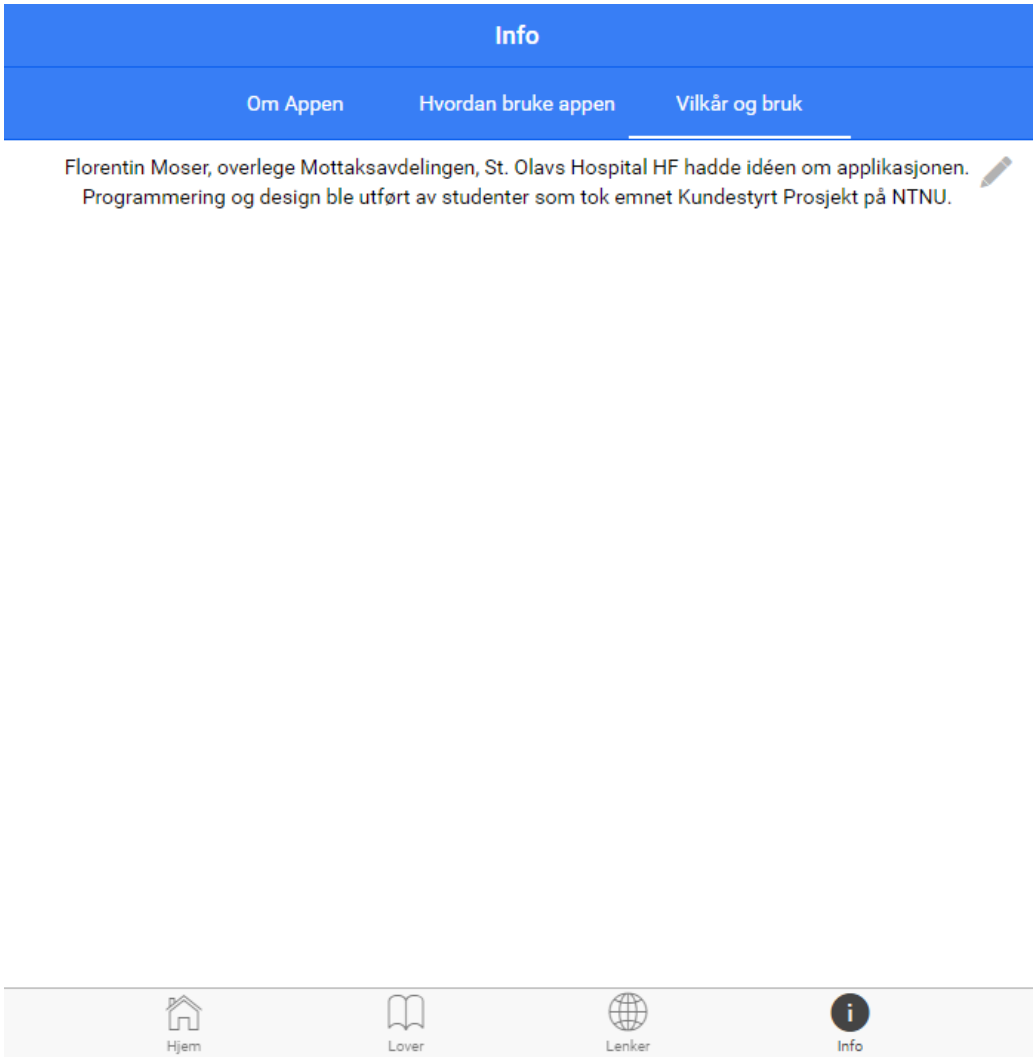


Figure J.14: The information view in administrator part of the system

J.3.d Creating a new category

After having pressed the plus icon in the main view the process of creating a new category is initiated. This subsection will describe the entire process from start to end, and will try be as detailed as possible in order to help the administrator understand the logic behind. This is usually a process that is done directly in the code by software engineers who have got experience with customizing application. Making administrators without any programming experience able to do this is not a easy task, and as mentioned earlier, the logic of process gets a bit complicated, but through this step by step guide the goal is to make the task as manageable as possible. It is worth noting that at any part of the process the user able to cancel the process. By doing this none of

the added content so far will be added to the application.

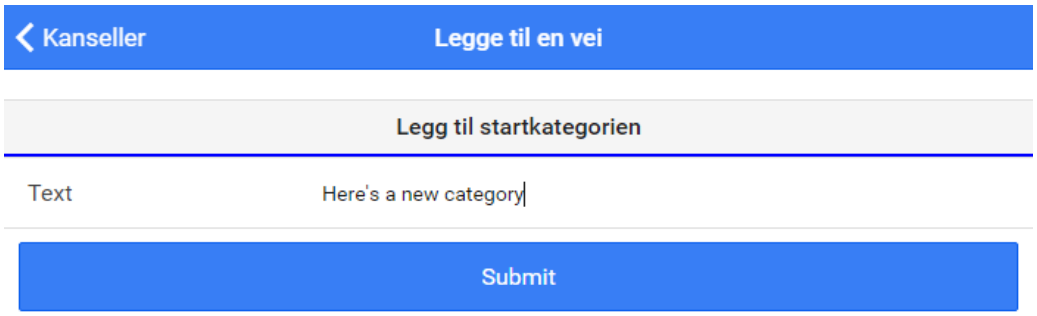


Figure J.15: Step 1: Add text to be displayed on the category

Step 1:
After having pressed the plus icon on the user is navigated to the view illustrated in figure J.15. In this view the administrator is asked to add the text that will be displayed on the category. After the category text is added the administrator submits the text by hitting the submit button.

Step 2:
In the next step the administrator is able to chose which type the category should represent. The three types are presented in section J.1.a, but they will be described in more detail underneath. In figure J.16, J.17, and J.18 the text added in step 1 is displayed next to the number 1 and the three different category types are displayed by the numbers 2 through 4.

After having followed the instructions under the chosen category and made sure that all the content is properly added the administrator clicks the submit button and the confirmation prompt, presented in figure J.19, where the administrator is asked to conform the addition of a new law

- **Direct category:** the direct category is displayed in figure J.16. As shown in the figure this category is represented by the number 4 ('Link til lov'). For this category the administrator chooses one of the laws that are already implemented in the application. Keep this in mind when creating a new category, the law needs to be added first if does not already exist in the application. The law that the administrator wishes to use is selected by pressing the choose button next to the wanted law.

- **Yes / No category:** the yes-no category is shown in figure J.17. For this category type the administrator submits the question to be displayed on the category, marked by the number 5 in the figure, after it is pressed and submits it by pressing the submit button. The number of outcomes, indicated by the number 6 in the figure, is always set to two when the administrator presses submit, even if he or she chooses to change the value before pressing submit. After submitting the question on for the category the user is asked to determine the behavior of the yes and no buttons. The behaviour of these buttons are described in section J.1.a. The behaviour of these buttons are implemented following the same pattern as the behaviour of the categories and the administrator is asked to read the guide based on the categories.
- **Multiple answer category:** the multiple answer category is displayed in figure J.18. This type acts similarly to the yes-no category, except that the administrator is given the option to have three or more possible answers. Each of the possible answers and their behaviour needs to be added by the administrator.

Step 3:

After submitting the new category the category should be displayed on the home page in the same way as in figure J.20 and the process is completed.

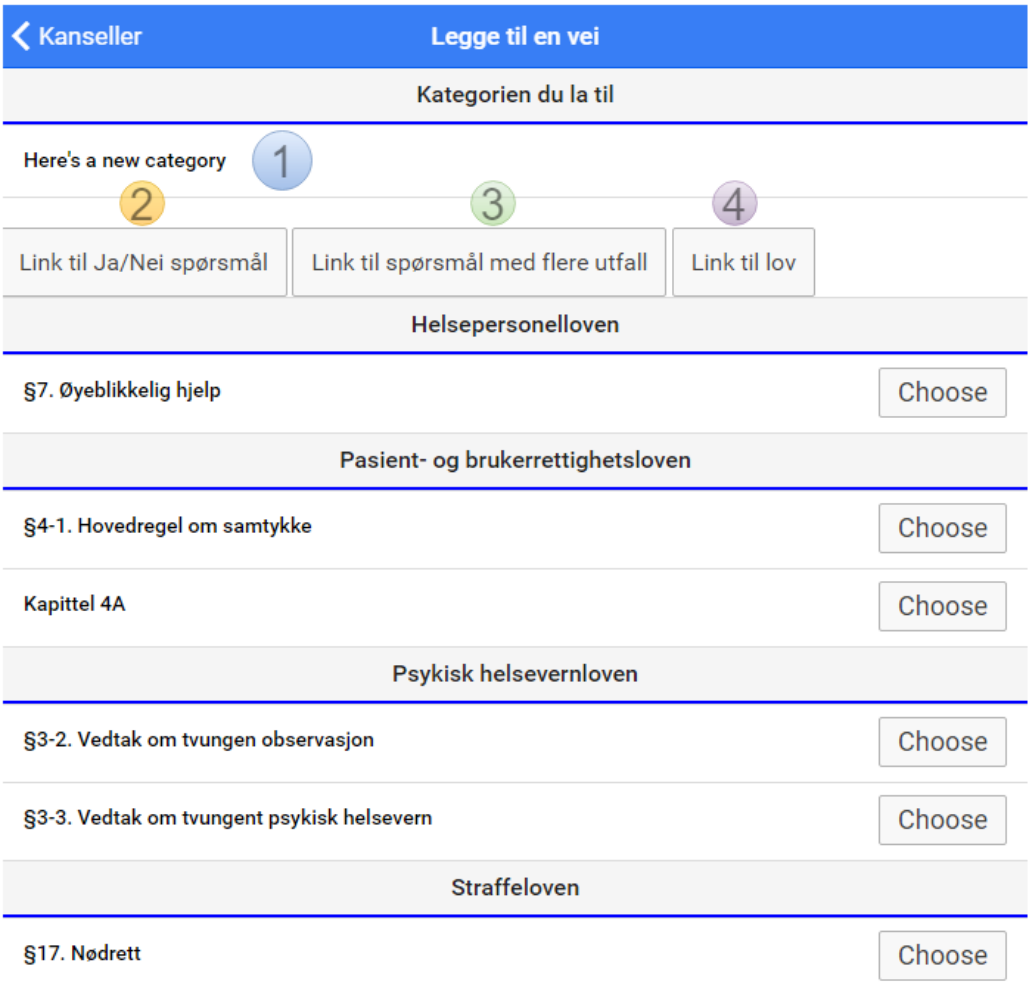


Figure J.16: Direct category

Figure J.17: The yes-no category

Figure J.18: The multiple answer category

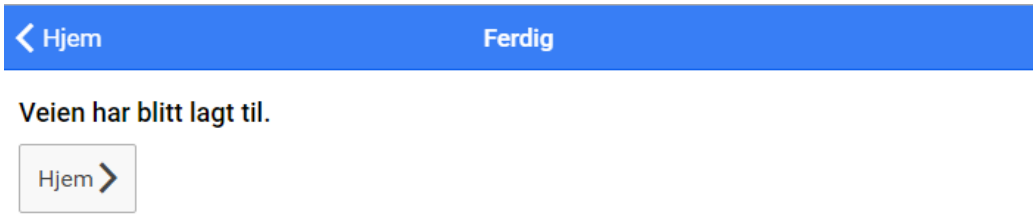


Figure J.19: The confirmation prompt displayed when submitting a new category

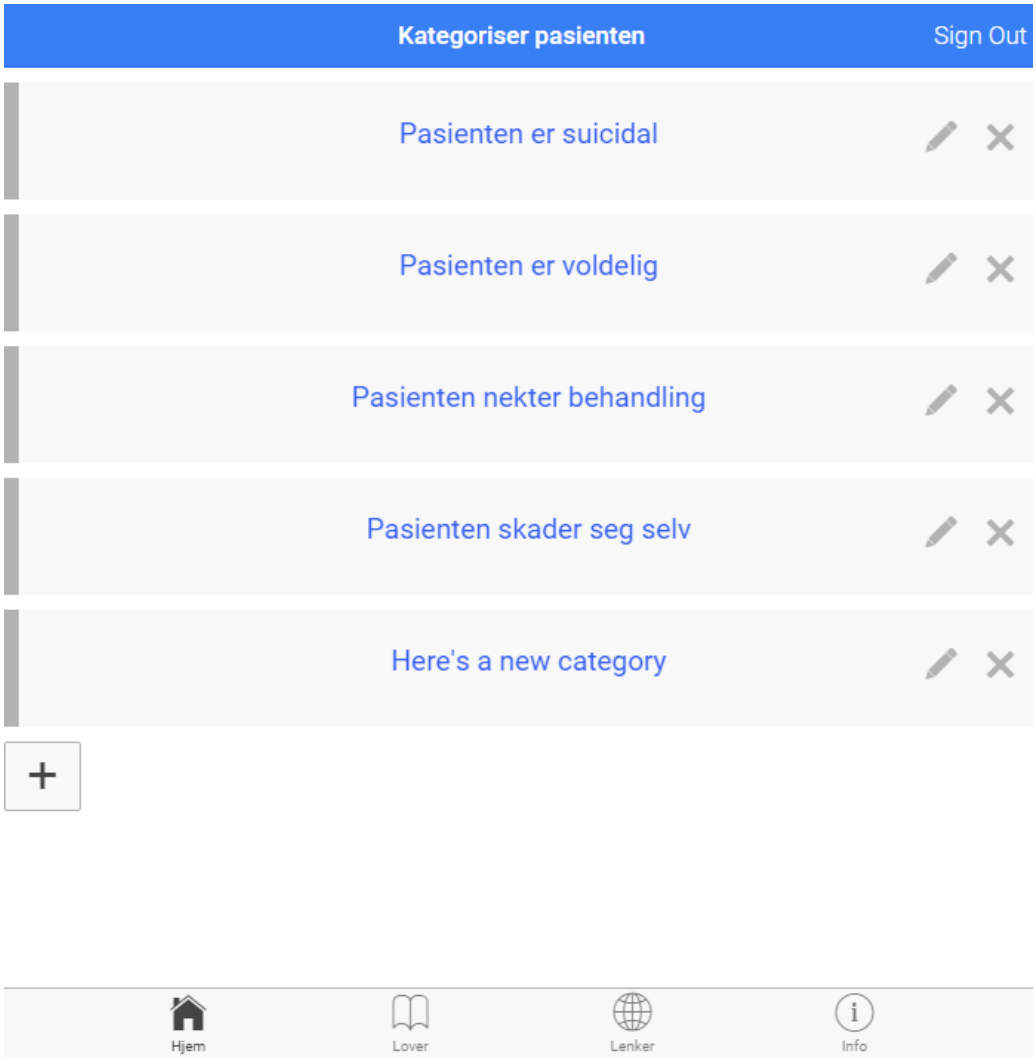


Figure J.20: The new category is now displayed

Bibliography

[1] R. Appel. Modern Apps : Mobile Web Sites vs. Native Apps vs. Hybrid Apps. https://msdn.microsoft.com/en-us/magazine/dn818502.aspx, 2014. [Accessed 11-November-2015].

[2] E. AS. EQS Kvalitetssystem. http://www.extend.no/eqs-kvalitetssystem/, 2013. [Accessed 11-November-2015].

[3] L. Bass, P. Clements, and R. Kazman. Software Architecture in Practice, 2012.

[4] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*, pages 212–5. Addison-Wesley Professional, third edition edition, 2012.

[5] B. Bos. Cascading Style Sheets home page. http://www.w3.org/Style/CSS/, 2015. [Accessed 9-November-2015].

[6] P. Bright. HTML5 specification finalized, squabbling over specs continues. http://arstechnica.com/information-technology/2014/10/html5-specification-finalized-squabbling-over-who-writes-the-specs-continues/, 2014. [Accessed 9-November-2015].

[7] D. Co. Ionic. http://ionicframework.com/, 2015. [Accessed 9-November-2015].

[8] W. Cunningham. Manifesto for Agile Software Development. http://agilemanifesto.org/, 2001. [Accessed 10-November-2015].

[9] J. D. Davidson and D. Coward. Servlet Specification version 2.2. http://www.people.vcu.edu/ wjo/servlet2_2 − *spec.pdf*, 1999. [$Accessed 10 - November$ − 2015].

[10] P. D. S. Edlich. Your Ultimate Guide to the Non-Relational Universe! http://nosql-database.org/, 2015. [Accessed 11-November-2015].

[11] Gantt.com. What is a Gantt chart? http://www.gantt.com/, 2015. [Accessed 4-October-2015].

[12] A. Gjærum. Venøs Trombemboli. http://vte.madebynice.com, 2015. [Accessed 11-November-2015].

[13] M. D. Group. Running your app on Android or iOS. https://www.meteor.com/tutorials/blaze/running-on-mobile, 2015. [Accessed 10-November-2015].

[14] M. D. Group. The JavaScript App Platform. https://www.meteor.com, 2015. [Accessed 7-November-2015].

[15] T. U. Group. LaTeX – A document preparation system. http://www.latex-project.org/, 2015. [Accessed 10-November-2015].

[16] M. N. N. Henriksen. ground:db. https://github.com/GroundMeteor/db, 2015. [Accessed 11-November-2015].

[17] A. Inc. What's new in Xcode 7. https://developer.apple.com/xcode/, 2015. [Accessed 10-November-2015].

[18] A. Inc. Designing for iOS. https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileH Last updated 2015-10-21. [Accessed 4-October-2015].

[19] A. Inc. Submitting Your App. https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributi $CH9 - SW1, Lastupdated2015 - 10 - 21.[Accessed9 - November - 2015]$.

[20] M. P. C. T. Inc. A Special Announcement: Trello Is Now Part Of Trello, Inc. http://blog.trello.com/trello-is-now-trello-inc/, 2014. [Accessed 9-November-2015].

[21] A. S. Incorporated. Adobe Photoshop. See what's possible. http://www.photoshop.com/, 2015. [Accessed 10-November-2015].

[22] L. Jaccheri. Chapter 8 - Testing. `http://folk.ntnu.no/magnlu/TDT4140%20Systemutvikling/Slides/Ch8_Testing.pdf`, 2013. [Accessed 10-October-2015].

[23] E. N. Jackson S, Joshi A. Recent Research on Team and Organizational Diversity: SWOT Analysis and Implications. http://jom.sagepub.com/content/29/6/801.short, 2003. [Accessed 4-October-2015].

[24] I. Jacobsen, M. Christerson, P. Jonsson, and G. Overgaard. *Object-Oriented Software Engineering - A Use Case Driven Approach*. Addison Wesley Professiona, first edition edition, 1992.

[25] P. Kruchten. Architectural Blueprints—The "4+1" View Model of Software Architecture. IEEE Software 12 (6), November 1995. [Accessed 4-October-2015].

[26] G. Kumparak. Slack's Co-Founders Take Home The Crunchie For Founder Of The Year. http://techcrunch.com/2015/02/05/slacks-co-founders-take-home-the-crunchie-for-founder-of-the-year/, 2015. [Accessed 9-November-2015].

[27] Microsoft. Capitalization Styles. https://msdn.microsoft.com/en-us/library/x2dbyw72(v=vs.71).aspx, 2015. [Accessed 9-November-2015].

[28] Microsoft. Integration Testing. `https://msdn.microsoft.com/en-us/library/aa292128(v=vs.71).aspx`, 2015. [Accessed 12-October-2015].

[29] J. Nielsen. 10 Usability Heuristics. http://www.useit.com/papers/heuristic/heuristic$_list.html$, 2012.$[Accessed12 - November - 2015]$.

[30] NTNU. TDT4290 Customer Driven Project. https://www.idi.ntnu.no/emner/tdt4290/, 2015. [Accessed 11-November-2015].

[31] A. Oliver. How to run your app on an Android device. https://github.com/meteor/meteor/wiki/How-to-run-your-app-on-an-Android-device, 2014. [Accessed 10-November-2015].

[32] D. E. Perry and A. L. Wolf. Foundations for the Study of Software Architecture. SOFTWARE ENGINEERING NOTES vol. 17 no. 4, October 1992. [Accessed 4-October-2015].

[33] J. Rasmusson. Scrum model. http://www.agilenutshell.com/scrum, 2015. [Accessed 10-November-2015].

[34] D. W. W. Royce. MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS. http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf, 1970. [Accessed 4-October-2015].

[35] K. Scotland. Aspects of Kanban. http://www.methodsandtools.com/archive/archive.php?id=104, 2010. [Accessed 9-November-2015].

[36] A. S. Solutions. Nice to meet you. We're Axure. http://www.axure.com/company, 2015. [Accessed 7-November-2015].

[37] C. . A. Sundar Pichai, SVP. Introducing Google Drive... yes, really. https://googleblog.blogspot.no/2012/04/introducing-google-drive-yes-really.html, 2012. [Accessed 9-November-2015].

[38] D. Svanæs. Don Norman "The design of Everyday Things". TDT4180 MMI, lectures from the course, 2011. [Accessed 12-November-2015].

[39] D. Svanæs. Gestalt principles. TDT4180 MMI, lectures from the course, 2011. [Accessed 12-November-2015].

[40] G. Team. GanttProject Team. http://www.ganttproject.biz/about, 2015. [Accessed 4-October-2015].

[41] Teamweek.com. Make plans. Change plans. https://teamweek.com/, 2015. [Accessed 9-November-2015].

[42] Unknown. JavaScript. https://en.wikipedia.org/wiki/JavaScript, Last Modified 29 October 2015. [Accessed 7-November-2015].

[43] Unknown. Application programming interface. https://en.wikipedia.org/wiki/Application$_{p}rogramming_{i}nterface, Last$M
$November - 2015$].

[44] Unknown. MongoDB. https://en.wikipedia.org/wiki/MongoDB, Last Modified 8 November 2015. [Accessed 9-November-2015].

[45] Unknown. PhoneGap. https://en.wikipedia.org/wiki/PhoneGap, Last Modified 9 November 2015. [Accessed 9-November-2015].

[46] M. Walraven. Mobile Development Install: Android on Mac. https://github.com/meteor/meteor/wiki/Mobile-Development-Install:-Android-on-Mac, 2015. [Accessed 10-November-2015].

[47] N. Wientge. meteor-ionic. https://github.com/meteoric/meteor-ionic, 2015. [Accessed 10-November-2015].