TDT4290 - Customer Driven Project

Scalpel

A clean cut in surgical planning

Grimstad, Eivind
Graneggen, Nina Røsdal
Ohlsen, Felix
Vatsendvik, Fredrik
Johansen, Jenny Stange
Grov, Magnus Austlid
Vatne, Svenn-Helge

November 16, 2017

Abstract

Planning the operation of a surgical department is a complicated task that involves several people and units, with their own needs for capacity and resources. Changes may happen quickly and lead to consequences for everyone. OpPlan is a software solution developed by HEMIT to allow for shared and concurrent planning of surgical departments, but due to it's implementation of a large set of features it does not scale well down to smaller handheld devices like phones and tablets. This project aims to deliver a proof of concept solution, targeting modern smartphones, that will work as a supplement to OpPlan and speed up organizational tasks to reduce the need for accessing a computer. The result of this will be the web based surgical planning solution Scalpel.

The team decided to develop the application using React, a popular JavaScript framework tailored to work well on both phones and computers, with the possibility of converting it to a native application should better performance or hardware utilization be desired. This also allowed the team to use a lot of existing components, resulting in less effort on the implementation. By interviews, usability tests and studying the existing system, the team settled on a feature set consisting of: a graphical timeline view for operations, a list based view, a detailed section for each operation, a personalized graphical view for operations the user is involved in, with the possibility to switch between dates and plans inside the application. Additional smaller features were also introduced that did not exist in OpPlan, such as the ability to end a phase of an operation with just a few clicks and the ability to call personnel directly from the application.

Work on the project was done following a set of the agile methods present in Scrum and Extreme Programming. The project was segmented into six sprints, with five of these acting as conventional sprints with the target of producing a minimum viable product, and the last for fine tuning the application, the report and producing the video. Each week meetings were held with both the customer and the advisor, to ensure progress for the application and the report. Each sprint started with sprint planning to decide on tasks for the backlog, including planning poker for estimating these. At the end of a sprint a sprint retrospective was held to discuss how the sprint went and how the team could improve for the next sprint.

The final web application shall be served by servers on closed networks at the respective hospitals. By communicating with an external API it will be able to exchange data with OpPlanWas, the windows communication foundation platform serving the OpPlan clients. Communication will be handled over encrypted HTTPS connections, with token based authentication using an Identity Access Management system. The application will only run on corporate devices.

Contents

Chapter 1

Introduction

1.1 Name

Scalpel - A clean cut in surgical planning.

1.2 Customer

Helse Midt-Norge IT, hereafter referred to as HEMIT. Representatives from HEMIT are Rune Grimstad, Marius Moholdt, Rolf Holte, Tore Aurstad and Øyvind Bech.

1.3 Background and Description

This project was part of the course TDT4290 Customer Driven Project at NTNU. The course should give students a deeper understanding of software development, by working with a real customer. The class was divided into 14 teams. The teams were assigned customers, each with their own case. The goal was to develop a solution for the problem in the case description, while collaborating with the customer. At the end of this development process each team was going to hold a presentation for the customer, the advisor and an external examiner whom would grade the project. The team was paired up with HEMIT, whom have described the following situation:

"Handling the logistics related to surgeries is a mission critical task for all hospitals. Hospitals in Central Norway perform this task using an application developed by HEMIT called OpPlan. This is a very large and complex application that requires the user to have access to a Windows computer. For last two years, HEMIT have been exploring options for how to build a light-weight

version of OpPlan for mobile devices, but unfortunately we haven't had the time to follow up this work. We therefore decided to suggest this as a task for the TDT4290 Customer Driven Project course." (Appendix I)

The team was going to figure out which features from OpPlan is suitable for mobile platforms, while also exploring new features. This would then be implemented, resulting in a functional prototype.

1.4 Problem Description

This section describes the problem, as it was presented to the team by the customer. The main goal of the project was to make some of the functionality of OpPlan available on a mobile phone. A big part of the task was to figure out what was suitable for a mobile platform. The customer suggested to show the timeline, which is central in today's OpPlan client. It shows an overview of the schedule of the day. Some thoughts by the customer:

- Display timeline
  - Edit, move, cancel and reschedule operations.
  - Add operations.
- Display operation theaters.
  - Show name of patient, diagnosis, operation code, duration and other parameters.
  - Filter on parameters.
  - Show operation crew.

The team should focus on showing one day at a time and one operation program at a time. Operation programs are collections of theaters usually based on their physical location. They can have up to 20 operation theatres.

The team should make it possible for anesthesia crew to use the application, by showing information based on the anesthesia tab in OpPlan. It would further be useful to show individual patients and operations with possibility for simple editing.

Working closely with end users and conducting usability testing should be a main focus for the team. Doing so will result in valuable information that will reveal what a mobile application should contain to be useful for hospital employees.

1.5 Project Goals

The goal for the project was to design and develop a mobile application which would give hospital employees easy access to the most important information from the desktop program, efficiently and on-the-go using a mobile device. This would result in less time spent on organizing and information gathering. The result would be a functioning front-end application, which could be integrated and expanded by the customer later on.

For the course, the goal was to get experience with software development and teamwork by working with a real customer.

1.6 Scope

The project lasted for 14 weeks, from 29 August until 23 November. The team consisted of seven students, each of which had 25 person hours per week. This means the total number of available person-hours was expected to be 2450.

The course description specifies that the team must deliver a functional program. The customers said that "The task was fairly open so our expectations were that the students would do a concept study and deliver a design sketch and possibly a clickable prototype." Since the customer had low expectations and gave the team an open task, it is mainly limited by the course itself.

The scope is defined by the lower limit set by criteria defined in the course description and the time frame.

1.7 Stakeholders

The stakeholders for this project are listed in Table 1.1.

Stakeholder	**Interest**
The team	Satisfy customer and complete the course
The customer	Solution to their problem
The advisor	Help the team with communication and solution
The examiners	Evaluate solution
The end users	Useful product

Table 1.1: The stakeholders

1.7.1 The Team

The team consisted of seven students from NTNU studying computer science. The interests of the team are to complete the course with a good result,

get development experience, get experience with academic writing and build connections to the industry. The members of the team are listed in Table 1.2.

Name	E-mail
Nina Røsdal Graneggen	ninarg@stud.ntnu.no
Eivind Grimstad	eivigri@stud.ntnu.no
Felix Ohlsen	felixpo@stud.ntnu.no
Fredrik Vatsendvik	fredrva@stud.ntnu.no
Jenny Stange Johansen	jennysj@stud.ntnu.no
Svenn-Helge Vatne	svennhev@stud.ntnu.no
Magnus Austlid Grov	magnusag@stud.ntnu.no

Table 1.2: The team

1.7.2 The Customer

The customer consisted of several employees from HEMIT. Rolf Holte was considered the product owner of OpPlan and this project. HEMIT provides software to the hospitals. Doctors, nurses and other hospital employees are the end users. The main interest of HEMIT is to explore new opportunities, which can make their products better. The end users want to spend more time on their core work, saving lives. The customers are listed in Table 1.3.

Name	Email
Rolf Holte	rolf.holte@hemit.no
Rune Andreas Grimstad	rune.andreas.grimstad@hemit.no
Marius Moholdt	marius.moholdt@hemit.no
Tore Aurstad	tore.aurstad@hemit.no
Øyvind Bech	oyvind.bech@hemit.no

Table 1.3: The customers

1.7.3 The Advisor

The advisor provided the team with counseling during the project. Her contact details can be found in Table 1.4.

Name	E-mail
Letizia Jaccheri	letizia.jaccheri@ntnu.no

Table 1.4: The advisor

1.7.4 The End Users

The end users are people working at the hospitals, who HEMIT are providing with software. The end users interest in the project is based on their need for software to use on a mobile device, that will make their everyday work easier. During the project the team contacted potential end users by having interviews, a questionnaire, field study and user tests. It was important for the team to understand the end users in order to solve the problem given by the customers.

Chapter 2

Planning

2.1 Choice of Lifecycle Model

For every project, a team has to decide how to organize and structure it. In this chapter the most common methods will be introduced with pros and cons. To decide which lifecycle model to use for this software project, the team looked at the possibilities and compared them to each other. The team decided on a mix of Scrum and Extreme Programming as the software engineering model.

2.1.1 Waterfall

The waterfall model is a commonly used model and has a linear structure. In the waterfall method every phase comes after a phase is completed and tasks can be divided according to phases. Every phase has a start and end point. This model was historically introduced because in some processes, like building a house, changes to certain things are not possible later in the project. The foundation can not be changed if the house is already built on it. Every phase has a specific goal and 6 phases are most common as shown in Figure 2.1.

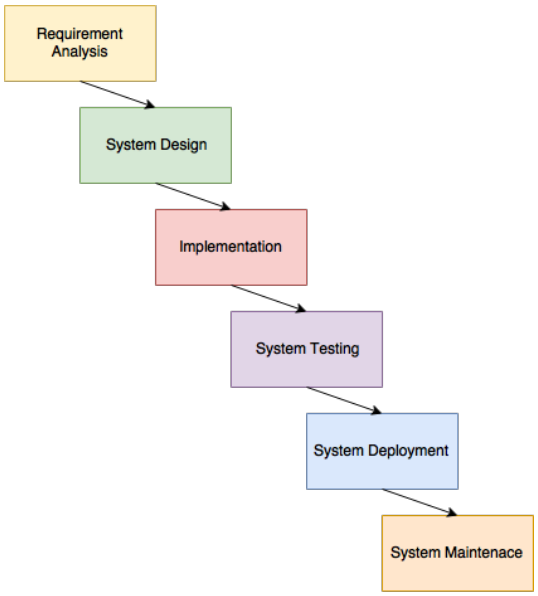


Figure 2.1: A model showing the sequence of steps of a waterfall process

The waterfall model has met some criticism. Often, the customers does not know what the final product should look like, and find it difficult to define a complete list of requirements in the earliest phase of the project. Also, making changes in the later phases requires a lot of work to be done all over again, making changes costly. The waterfall models does not normally involve frequent deliveries to the customers, that means that any misunderstandings of requirements will often not be discovered until system delivery (Dybå and Dingsøyr 2008).

According to Dybå and Dingsøyr, projects where the customers know what a final solution should include are suitable for the waterfall method. The customers must be able to define the final set of requirements at the very beginning of the project. In addition to the requirements, the time frame, resources and technology should remain the same during the project. Also, the waterfall method is convenient when the customer does not need to, or are not able to, give feedback during the project.

2.1.2 Agile

The Agile methodology is something that has reached a lot of popularity over recent years, particularly in software development where most businesses today often use agile in some form, the most popular being Scrum (Denning 2015). Agile methods are tailored to the specific project, with it's main benefits

being flexibility and adaptability. As in the waterfall method there are phases, but these phases are iterative. The cycles are incremental. Each release has more functionality that build up onto the last cycle.

2.1.2.1 Scrum

"Scrum focuses on project management in situations where it is difficult to plan ahead, with mechanisms for 'empirical process control'; where feedback loops constitute the core element. Software is developed by a self-organizing team in increments (called 'sprints'), starting with planning and ending with a review." (Dybå and Dingsøyr 2008). Scrum is an iterative project organization method. It uses sprints as the smallest unit. Each sprint has a specific length and is one to four weeks long. All tasks that should be done to complete the project are written down in a product backlog, often by the product owner. Several meetings are held each sprint by a person chosen as the Scrum Master. Every sprint begins with a sprint planning. In this meeting the team will chose a subset of tasks from the product backlog that should be finished by the end of the sprint. These tasks are put into the sprint backlog.

Before the work begins every day the team holds a daily scrum meeting. In this meeting, all team members stand up and tell what they have achieved since the last meeting and what they are going to work on this day.

After every sprint the team review their own work in the sprint review. The sprint review often includes other stakeholders, such as the customers. After a sprint is completed there is also a sprint retrospective meeting.The retrospective meeting focuses on the work process and team dynamics, and not what tasks that have been completed. The meeting includes evaluating what worked well during the sprint, what did not work well and what the team could start doing in order to improve.

Using Scrum involves frequent deadlines, which usually involves showing the completed work to the customers. This will motivate the team members and thus improve the efficiency of the team. The customers can participate in planning work and prioritize tasks. This reduces the risk of project failure, as the team can get feedback and easily adapt to changes.The retrospective meetings makes sure that the team is continuously improving their work process.

2.1.2.2 Extreme Programming

In extreme programming, XP for short, personal horizontal coordination is achieved using pair programming and co-location (Beck 1999). XP is built on communication. Good team communication is necessary for a project to work well when using XP.

As opposed to the waterfall model the customer does not need to know

the specific requirements for the system. The software can change its direction during the process. The goal is to build a working product with some core functionality and further build upon these. Within a XP team there are no strict roles, all team members are considered equal. This helps to prevent a monopoly of knowledge.

In general a user creates stories that leads to new tasks (Juric 2000). Then the team creates pairs that program together. These pairs will create a test case. The solution that will pass the test case gets programmed in the pairs. If they need help the team can organize that. If the code is too complex they will refactor the code without any hesitation. After this process, an acceptance test is held.

Pair programming is an important tool for XP, but studies have shown that "pair programming was difficult when there was a large skill differential between the members of the pairs" (Dybå and Dingsøyr 2008) which was considered to be the case in this project, based on previous experience and dialogue within the team.

2.1.2.3 Kanban

Kanban is a project management method that tries to visualize the process of creating as much as possible (Sommerville 2015). It's common to use a Kanban board. In the board there are 6 columns. Each column represents specific work that has to be done to a problem. Each index card represent a task. A column has a maximum of index cards. In the core it is like a flow from one side to the other. If there is a index card that stays longer than regular in one column the team has to look into that. A Scrum board has many similarities to the Kanban board, with the main distinction being that a Scrum board act per sprint and do not add features to maintain a certain number of them on the board. It also has no limitations on the number of tasks that can be in each category at a time as the Kanban board.

2.1.3 Selecting a Project Management Method

In the beginning of the project the team had to decide on how to manage it. First the team had to decide between the waterfall model and the agile methods. The team decided to use agile methods because the requirements of the project were not clear in the beginning, with the possibility of features being introduced as the project went along or decisions being reconsidered. This was not very well suited for a waterfall model. The customer had not fully planned what functionality the final mobile application solution should contain. The team's main task was to figure out what the end users needed. Further, the customers wanted frequent deliveries where they could give feedback. For this project, the team believes that using the waterfall method would involve a great risk

of implementing a product that would be useless for the end user. Being able to define and change requirements after the earliest phases was important to the team. The team believes that the success of the project was dependent on frequent feedback both from the customers and the end users. Also, three team members already had experience using agile methods. Another point was that the earlier teams in the course had worked with agile methods with promising results.

Kanban, XP and Scrum were the three frameworks that the team focused their decisions on. While every method has features that could help the project, the team chose to use Scrum as the main methodology. Most members had experience with working in Scrum teams, and believed it was a good way to manage the project. The constant flow of Kanban could lead to reduced motivation. The team decided to not use a Kanban board, but rather use a Scrum board with the time limitations and not the limitation on number of tasks being in the board. This can lead to a better prioritization of tasks when only considering a smaller subset each time, and also lead to improved motivation by visualizing the total amount of active tasks being reduced and the completed tasks increasing. Using Scrum, the team already had frequent deliveries and close communication with the customer. Extreme Programming was used to a much lesser extent than Scrum, but once the implementation phases began the team decided that it would be beneficial to do some work in pairs. The team also found Planning Poker to be a good method to use for estimating tasks, which is a common practice in both Scrum and XP.

2.2 Project Work Plan

2.2.1 Sprints

The team decided to split the project into six sprints. That meant that each sprint would be approximately 2 weeks. In every sprint the team began by holding a sprint planning meeting, where prioritized tasks were taken from the product backlog and put into the sprint backlog. The sprints can be viewed in Figure 2.2.

The first sprint was mostly spent on research of the requirements for the solution. This process is described in Chapter 3.3. The team needed to get a better understanding of the task. Some tasks completed in this sprint were; talking to the customer, arranging some interviews with daily users of OpPlan, reading the compendium and getting an overview of how the solution should be developed. The task was very broad, so the team tried to collaborate with the customer and interview objects to get a better understanding of what the main functionality of the application should be. The team made a draft of the requirement specification with both functional and nonfunctional requirements. There was also developed a backlog with user-stories.

The second to the fifth sprint were more regular scrum-sprints. Before each sprint a set of tasks to be done were assigned to the sprint. The goal was to complete all planned tasks within the end of the sprint. For each of these sprints the team completed the sprint planning, the sprint retrospective and the sprint review.

The last sprint was mainly used for testing, finishing the report and finishing the presentation.

2.2.2 Activities

2.2.2.1 Sprint planning

Each sprint started with a sprint planning meeting. During this meeting, tasks to be done during the sprint were put into the backlog. The amount of work required to complete each task was estimated by the team by using planning poker. In planning poker each team member guessed the amount of work for a particular task, and then the team agreed on the most accurate time estimate after a discussion. According to a study done in 2017, planning poker is shown to be an advantageous planning technique (Moløkken-Østvold, Haugen, and Benestad 2008). Described advantages include better planning due to incentivizing discussion and more objective estimates because they are revealed simultaneously. The team agreed to use 1 story point for each hour of work the task would require. By looking at how much time there are in the sprint and the time estimate for different tasks the team could choose a set of high prioritized tasks and put them into the upcoming sprint. The set of tasks chosen was approved by the customer at the next customer meeting. The goal for the sprint was to complete all these tasks within the end of the sprint.

2.2.2.2 Sprint retrospective

At the end of each sprint, a sprint retrospective meeting was held. At these meetings the team discussed what went well in the last sprint, what they should start doing in the next sprint, and what they should stop doing. Thus, the retrospective looked at how work and cooperation was done in the last sprint, and not what work that was done. After writing down all the keywords for the retrospective, the team collectively selected the most important things to focus on during the upcoming sprint. From the second retrospective, the team also reflected on whether or not the goals specified in the last retrospective were achieved.

2.2.2.3 Sprint review

Further, at the first customer meeting after the end of a sprint, a sprint review was done. In the sprint review the completed work was shown to the

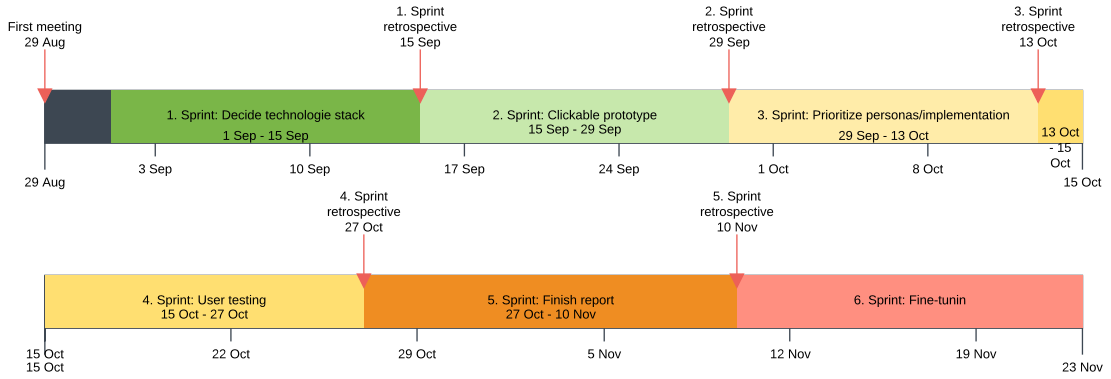


Figure 2.2: Chart showing the planned sprints, including weekly dates and dates for sprint retrospectives for the entire project.

customers. This served as and update for the customer on what was done and was a chance to notify the team if there had been any misunderstandings. Also, tasks that were not finished as agreed could be discussed. Work that was planned to be done in the sprint, but was not completed by the end of the sprint, was usually moved on to the next sprint.

2.3 Milestones

The team defined a set of milestones for the project. The goal of each sprint was to have all the planned tasks finished. By the 16th of November both the report and the video should have been be finished, leaving the team to work on the final presentation until the end of the course at the 23th November. The milestones can be viewed in Table 2.1.

Name	**Time**	**Goal**
Sprint 1 ending	September 15th	Decide on technology to use
Sprint 2 ending	September 29th	Finish clickable prototype
Sprint 3 ending	October 13th	Prioritize personas
Sprint 4 ending	October 27th	Get application ready for usability tests and conduct these
Sprint 5 ending	November 10th	Finish the application and have 110 pages in the report
Report and video delivery	November 16th	Finish the report and the video
Final sprint ending	November 23rd	Have the presentation ready
Final presentation	November 23rd	Present the project

Table 2.1: Important milestones for the overall project

2.4 Team Organization

The team had a flat hierarchy. Even though the team had roles, these roles were meant to give a direction and not cause a strict hierarchy. For example the report manager had the responsibility that the report got written, not to write it. As a result each team member had the option of delegating certain tasks within their responsibility when necessary. Figure 2.3 shows a grouping of the roles assigned to members of the team.

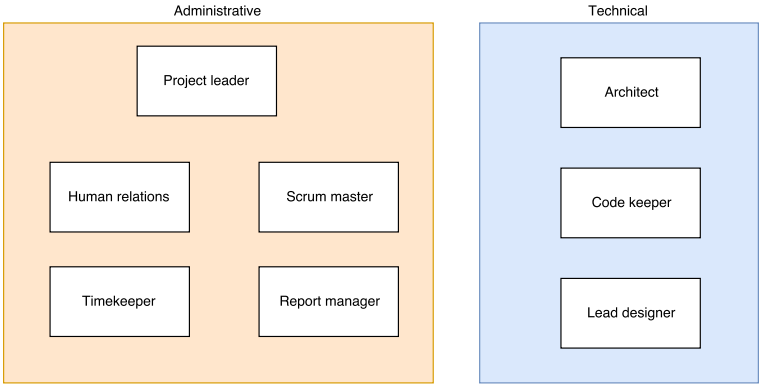


Figure 2.3: Model showing the roles assigned to members of the team.

2.4.1 Responsibilities and Roles

Everyone in the team were assigned a role with a set of responsibilities. The team also assigned a backup for every role. The task for the backup was to assure that the leader of the role were doing their job correctly and to take responsibility if the leader of the role was absent. Table 2.2 shows assignment of roles and the responsibilities of each of them.

Role	Name	Backup	Responsibility
Scrum master	Jenny	Magnus	Hold Sprint planning meeting Hold Sprint retrospective Make sure notes are taken from these meetings Make sure Scrum ceremonies are carried out
Project leader	Nina	Jenny	Lead meetings Send meeting invitations and reports Customer relations Leader meetings (report issues and keep team up to date)
Architect	Fredrik	Nina	Overall Architecture, have an overview of the different components and how they interact Keep diagrams up to date Flowcharts Use cases
Code keeper	Eivind	Fredrik	Make sure the code is clean and documented and reviewed Help people understand the code Manage use of coding conventions Responsible for setting up coding style, naming conventions
Human relations	Svenn-Helge	Eivind	Make sure everyone is feeling well and show up Task management, grouping for pair programming etc. Conflict settlements
Time keeper	Felix	Svenn-Helge	Make sure everyone is updating their time sheet Update calendar Make sure rooms are booked for work sessions Keep track of the late punishments
Report manager	Magnus	Felix	Make sure documents are kept up to date Know which documents the team has to write and what should be in them Create templates for meeting documents such as sprint retrospective
Lead designer	Felix	Fredrik	Keep track of design sketches Implement the clickable prototype Responsible for the main decisions in design

Table 2.2: The assigned team members and their backups for roles and their responsibilities.

2.5 Weekly Schedule

During this project the team had three weekly meetings. Times were set off for mandatory meetings where everyone in the team had to show up. It was decided to limit the amount of mandatory meetings, as team members often had conflicting schedules. When the mandatory meetings were finished everyone usually continued working on their assigned tasks.

Outside of meeting hours the team often scheduled voluntary working sessions. The assigned pairs for cooperation arranged their own meetings during the week as required to complete their tasks. If anyone needed help doing their task, this was agreed between each member. A typical week consisted of student meeting and lecture on Mondays, customer meeting and working at the HEMIT offices on Wednesdays, and student meeting, advisor meeting and sprint retrospective on Fridays. Later in the project this was expanded to include open work sessions at the customer offices on Mondays and Tuesdays as well. Figure 2.4 shows a view of a typical week schedule in Google calendar. In Table 2.3 a more detailed overview of the types of meeting and activities are listed.

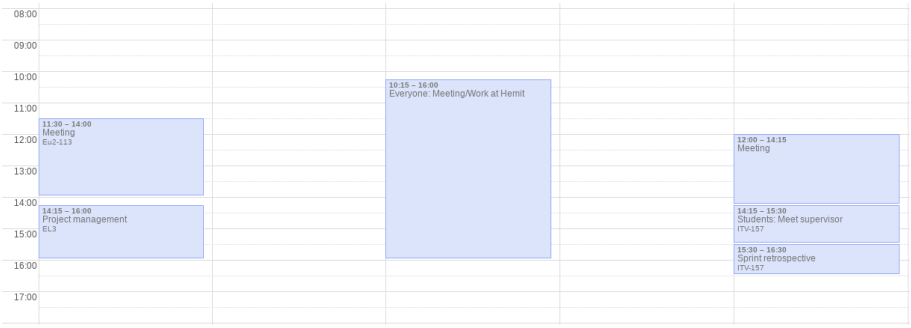


Figure 2.4: A typical week schedule.

Type	Day and time	Activity
Work session	Monday 12:15-14:00 Wednesday 11:00-16:00 Friday 12:15-14:00	All the members of the team that are available gather to work together on the project.
Customer meeting	Wednesday 10:15-11:00	The team gather at the customer offices and have a meeting with the the customer.
Advisor meeting	Friday 14:15-15:30	Meeting with the advisor. Discussion of the status report and issues.
Sprint retrospective	Friday 15:30-16:30	If a sprint is ending, a sprint retrospective is held.
Sprint planning	Monday 12:15-14:00	If a sprint has just ended the team plans the current sprint.
Lecture	Monday 14:15-18:00	Mandatory guest lecture.

Table 2.3: Detailed overview of weekly activities.

2.6 Templates and Standards

2.6.1 Meeting Templates

Templates for all relevant document types were made. This includes:

- Meeting agenda with customer
- Meeting minutes from customer meeting
- Meeting agenda with advisor
- Meeting minutes from advisor meeting
- Status report
- Sprint retrospective

These can be viewed in Appendix G. The templates helped the team save time and also made sure the documents were tidy and easy to read afterwards.

2.6.2 File Organization

Google Drive was used to organize files. There were specific folders to place documents, shown in Figure 2.5. All meeting documents are placed in the Protocols folder, in the sprint they belong to and the date of the meeting, while other resources were grouped into categories such as photos, models, research and personal folders. The team tried to keep the Google drive updated and well structured throughout the project.

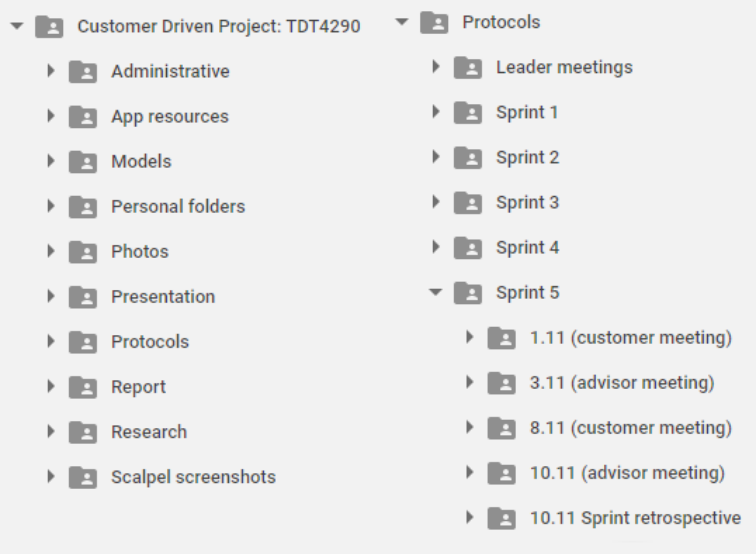


Figure 2.5: Overview of the folders on Google Drive.

2.6.3 Internal Project Meetings

Every project meeting that were mandatory had an agenda that was sent to every team member before the meeting. Such meetings were customer meetings and advisor meetings. The agenda was written by the project leader with inputs from the other team members. In the agenda the secretary were stated. This person wrote minutes for the specific meeting. The person had to upload it to Google Drive on the same day. In this way the team made sure that all decisions made at meetings were documented. Also, if anyone were absent at a meeting, they could stay updated by reading the minutes.

2.6.4 Hour Tracking

The team kept a time sheet updated during the project. Google Spreadsheet was used for this purpose. Each member was required to update their own hours.

The goal was to make sure that everyone spent almost the same amount of time on the project, and also to motivate by having the goal of 25 hours each week to strive for. The timekeeper was responsible to remind everyone to track their hours and also notify any team member if they were spending too little time on the project. An overview of the hours spent can be seen in the Appendix H.

2.6.5 Version Control Procedures

In this project the team has used git on Github for version control of code. It keeps a complete history of all changes that are made. That also includes the author of any change. In general, when a use case was handled a new branch was made. This made it easy to test a new version before it was integrated and to revert changes in case something had gone wrong. The hierarchy and merge history of branches were also kept so a complete overview could be viewed. The git workflow was in part based on the Feature Branch Worklow as it is described by Martin Fowler (Fowler 2009).

For documents Google Drive has embedded automatic version control for files. That made it easy for the team to look back at earlier versions of documents if necessary.

2.7 Risk Management

In this section, the team have studied what risks can affect this project and made strategies to overcome them, depending on the impact. The risks are placed in four risk response categories:

- Avoid: Have a plan that makes sure it can't happen
- Transfer: Transfer part of the risk to a third party
- Reduce: Actions chosen will reduce the likelihood of the risk or reduce the impact of the risk if it does happen
- Accept: The risk is accepted and managing it is more costly than the risk actually happening

The template used for considering risks can be viewed in Table **??**. The risks considered can be viewed in Tables 2.4 - 2.12.

ID	R1: Running out of time
Activity	All
Impact	H: Product quality can be reduced as some features have to be dropped. The team might have to work overtime. Might not have time to finish product and/or report.
Probability	H: It is difficult to estimate time
Strategy and action	Avoid The team always have to make sure to use the time wisely by prioritizing the most important tasks of each iteration.
Deadline	Continuous
Responsible	Scrum master, Project leader

Table 2.4

ID	R2: Team member is unable to work for a short period
Activity	All
Impact	L: Will lead to small delays
Probability	H: Someone might get sick or be busy with other work
Strategy and action	Reduce It's okay that it happens occasionally, but by staying in touch the team can get an assurance that the person is ready to come back as soon as possible. If the person's current task is urgent, the project leader has to be informed and tasks will be rearranged.
Deadline	Continuous
Responsible	Human relations

Table 2.5

ID	R3: Team member is unable to work for a long period
Activity	All
Impact	H: Can lead to large delays of important parts of the project
Probability	L: Can happen due to serious illness, dropping the course, etc.
Strategy and action	Reduce The team needs to have redundancy in their skill sets, so that reallocation of the person's responsibilities and tasks can be done between the remaining team members. Documents that belong to the project have to be available to all team members at all times.
Deadline	Continuous
Responsible	Human relations, Project leader

Table 2.6

ID	R4: Communication problems within the team
Activity	All
Impact	M: This can have an impact on the product quality and the productivity of the team, can end up doing double work
Probability	M: There is a lot of verbal and written communication
Strategy and action	Reduce Trello should be updated with the tasks every team member is working on. Tasks should be concise. Ask others if something is unclear. Communicate on Slack outside of work hours
Deadline	Continuous
Responsible	All

Table 2.7

ID	R5: Misunderstanding between team and customer
Activity	All
Impact	H: Product quality can suffer if the team do not understand what the customer wants. Impacts productivity.
Probability	M: The customer has technical competence, but the team does not have the same background and misunderstandings may happen.
Strategy and action	Reduce Have weekly meetings with the customer, make sure they are updated and included in the process. Get approval on documents written and tasks done.
Deadline	Continuous
Responsible	All

Table 2.8

ID	R6: Customer being unavailable for a short period (days)
Activity	All
Impact	M: Can lead to delays
Probability	L: There are two contact persons and it's unlikely that both will be unavailable at the same time. Customer is available for questions on Slack.
Strategy and action	Reduce Arrange regular meetings, so that the team keep in touch.
Deadline	Continuous
Responsible	Project leader

Table 2.9

ID	R7: Customer being unavailable for a long period (weeks)
Activity	All
Impact	H: The different customer representatives do not have the same knowledge, so if one of them is missing the team might not get the answers needed. This can have an impact on the product quality and lead to delays.
Probability	M: Customer represantitive can be on prolonged sick leave
Strategy and action	Reduce Make sure to plan ahead so that there is enough information to work with for the next couple of weeks. This will give the customer more time to give the team the answers needed. If both customer representatives are unavailable, the advisor should be contacted.
Deadline	Continuous
Responsible	All

Table 2.10

ID	R8: Lack of knowledge about technologies used
Activity	All
Impact	M: Affects the productivity of the team
Probability	M: The team are using technology that many of the team members are unfamiliar with.
Strategy and action	Reduce If something is difficult and/or is taking too much time, team members should ask for help from someone with more experience with the technology.
Deadline	Continuous
Responsible	All

Table 2.11

ID	R9: Low team motivation
Activity	All
Impact	M: Affects the productivity of the team and can reduce quality of the product
Probability	L: The project is pretty short which makes it easier to stay motivated
Strategy and action	Reduce Have rounds around the table where the team talk about how each member are feeling and what they are currently doing.
Deadline	Continuous
Responsible	Human relations

Table 2.12

2.8 Quality Assurance

To ensure the quality of the deliverable the team employed some quality assurance strategies. These includes both strategies for writing high quality code and for writing documentation.

2.8.1 Quality of Code

To ensure that the code written was of high quality the team conducted testing, code reviews and pair programming along with using linting and code comments.

2.8.1.1 Testing

The team made a decision early not to put too much focus into automated testing of code. Automated testing of user interface code is not a simple task. Even though there are frameworks for achieving this, it was not estimated to give enough value. Michael Cohn describes the relationship between different types of tests in his book, Succeeding with Agile (Cohn 2010). He describes the various aspects of automated software testing as a pyramid, where unit testing is on the bottom and testing of UI on the top. Items further up the pyramid are more time consuming and less useful. Additionally, UI tests are more brittle, so small changes are more likely to break the tests. This is undesirable since the team wanted to be able to rapidly test design decisions. On the background of this, the team decided to only create automated tests for some critial parts of the code, such as data transformations.

2.8.1.2 Code Reviews

Code review and manual testing before merging of new features was emphasized. A person should in most cases get a review from another team member, on both the quality of the code and the functionality. The only exception would be if the changes were very minor.

2.8.1.3 Linting

When the development environment was set up, an automated linter was added as part of the development build pipeline. When the development server was running, the linter would check for code style errors, such as mixing spaces and tabs, using semicolons, or double quotes instead of single quotes. It could also detect unused variables and functions. This made sure the team had a basic protection against code bloat and inconsistent style. Many of the choices that were made, such as using tabs, are more or less arbitrary so they were made only for the sake of internal consistency.

2.8.1.4 Pair Programming

The team did not enforce pair programming, but did encourage to work together by separating into pairs on each iteration.

2.8.1.5 Code comments

Comments in code are often necessary but should not be considered to be mandatory. It is better to write code that is easy to understand than to comment bad code. Thus, the team only used comments where the code was not clear enough by itself. The way the team considered code commenting came in part from the book Clean Code, by the renowned software engineer Robert C. Martin. He says that "the only truly good comment is the comment you found a way not to write" (Martin 2008).

2.8.2 Quality of documentation

The team made a lot of documents during the project. To be able to easily read each document, and make sure they had the correct content, templates were made for each of them. These templates can be found in G. The meeting agendas and minutes followed their templates and was read by all team members. The advisor reviewed the report and documents produced by the team regularly to make sure they followed the standards expected.

2.8.2.1 Report

To write the report the team followed the guidelines presented in the "technical writing" lecture of the course. The report was from sprint 3 regularly read and commented by the advisor to help get a clearer understanding of how the advisor and examiners would want the report to look like in the end. Tasks on writing on the report were a part of the Trello board, and were moved from 'Doing' into 'Testing' as they were completed. That meant that another team member would review it before the task was moved to 'Done'.

2.8.2.2 Team Meetings

To ensure good communication in the team had many face-to-face meetings and work sessions. For these meetings the team booked rooms at NTNU or at the customer's place and worked together. The benefits of this was that the team members could easily cooperate, ask others for help or ask them to review their work. At some of the meetings the team started with a 15 minutes daily scrum to get an overview over the situation. At that meeting each member would tell what tasks they had completed since last meeting, what they were going to work on at the particular day and if they had any issues.

2.8.2.3 Customer Meetings

The team had weekly meetings with the customers at their office. The calling for the meeting was delivered through Slack by 14.00 the day before the meeting, this made sure everyone had time to prepare. Minutes of the meetings were written and posted at Slack before 12.00 the day after. This made it possible for the customers to review the decisions made at the meetings and notify the team if there had been any misunderstandings. Also team members could read the minutes to get an update if they had missed a meeting.

2.8.2.4 Advisor Meetings

The team had weekly meetings with the advisor, where they discussed the progress since the last advisor meeting. The calling for meeting were sent out by 12.00 the day before the meeting, and included minutes from meetings, status report and relevant phase documents. These documents were discussed in the meeting and the advisor made sure that the team were on the right track. The advisor would also approve the documents, which means that they followed the standards expected.

2.8.3 Time of Response

To maintain regular communication with the customers, agreements were made on maximum response time.

- Maximum 24 hours to answer questions, e.g. on Slack.
- Maximum 48 hours to approve phase documents.
- Maximum 48 hours to approve minutes of customer meeting.

Chapter 3

Preliminary Study

3.1 Current System

3.1.1 Table of important concepts

Before discussing the current system it is necessary to introduce some terms related to OpPlan and surgeries.

Concept	Description
Operation	A surgical procedure involving one patient and crew from the hospital
Theater	A room in the hospital where operations take place
Operation plan	A set of theaters that often correspond to a type of operation
Operation phases	The different phases an operation goes through. They are - Preparation - Pretime - Surgery time - Posttime - Postoperation
Surgeon	A doctor performing a surgery
Anesthesia doctor	A doctor performing the anesthetization of the patient before an operation
Surgical nurse	A nurse assisting the surgeon during an operation
Anesthesia nurse	A nurse assisting the anesthesia doctor
Patient coordinator	A hospital employee working with coordinating patients and operations
OpPlan	The software that is currently used by Helse Midt-Norge to plan operations

Table 3.1: Concepts that will be used throughout the report

3.1.2 Overview

The team got access to a lot of information about the desktop application, OpPlan, that is currently in use. The customer provided the team with documentation and access to a running test version of OpPlan while working at their office. The following description includes the main functionality and the relevant features. As OpPlan is a large application not every aspect will be described.

OpPlan stands for 'OperasjonsPlanlegger' and is an application for planning and monitoring operations in hospitals made by HEMIT. HEMIT's goal with OpPlan has been to create a responsive, user-friendly and intuitive application which can help the users to focus on their tasks, and use OpPlan as a tool to help them in a hectic day-to-day situation. The main view of OpPlan can be seen in Figure 3.1.

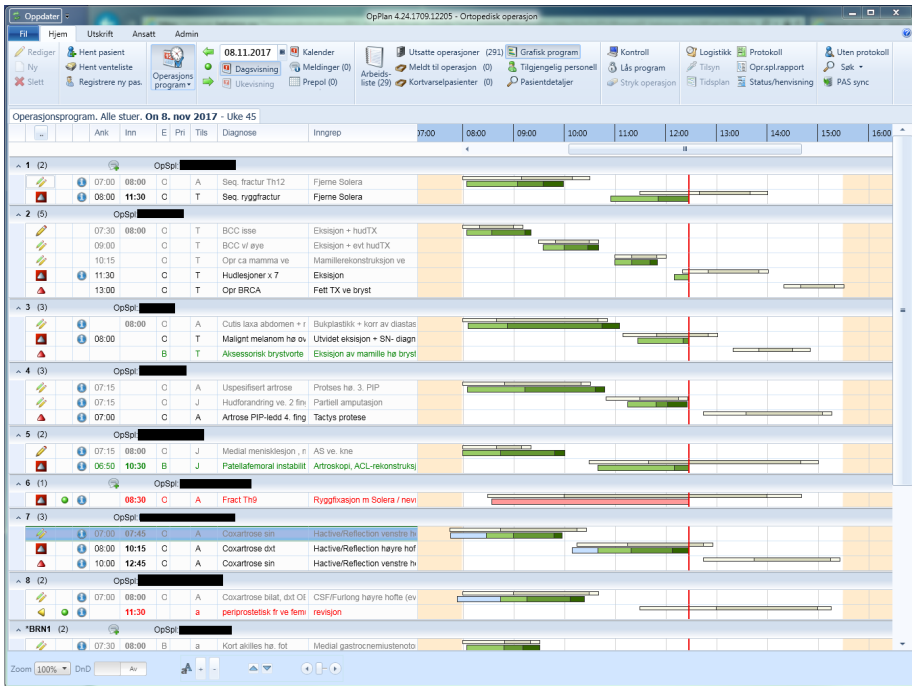


Figure 3.1: A screenshot of the main view of OpPlan.

OpPlan consists of different operation programs. An operation program has a set of operation theaters, an example can be viewed in Figure 3.2. Further, a user have to select a single operation program along with a role when logging in. The role determines the access rights, and is also stored when the user inputs data to the program. Examples of roles are ‘coordinator’ and ‘nurse’.

A theater is where an operation takes place. Usually only one patient can be in a theater at a time. Several phases of each operation are monitored in OpPlan. In OpPlan the theaters can be viewed as lists of patients. The columns for the list contains different data and which columns that are shown can be somehow changed by the user. Columns that are usually shown includes patient name, when they arrived, which unit they belong to, diagnosis, surgery and which crew that are scheduled to participate in the operation. One column that is commonly used is the one that shows the timeline.

The timeline is by the customer considered one of the most important features of OpPlan. It shows a bar view with time slots horizontally, and can be viewed in Figure 3.3. Two things are shown, the planned time and the actual time for the operation. For both, each phase are indicated by a distrinct color coding. As the operation progresses the timeline of the actual time is updated in real time. The operation crew uses check boxes to indicate which phases of

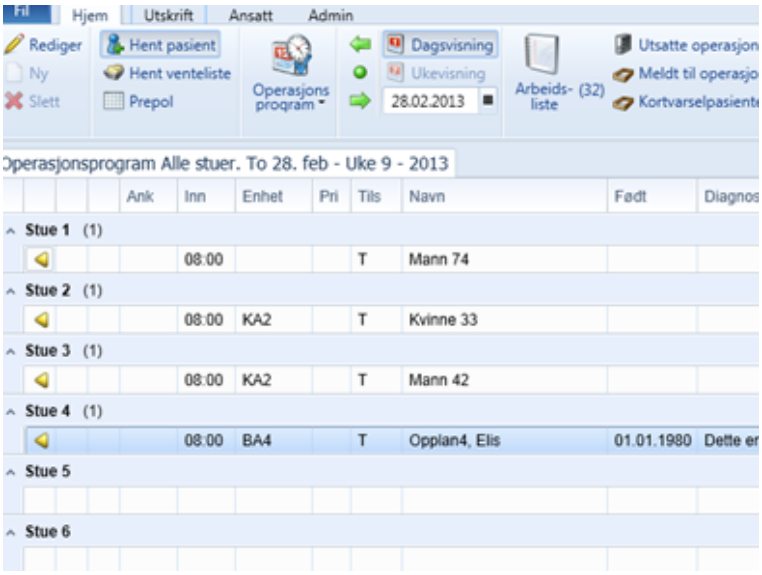


Figure 3.2: Screenshot from OpPlan showing an operation program with a set of theaters.

an operation that are completed. This is reflected in the timeline.

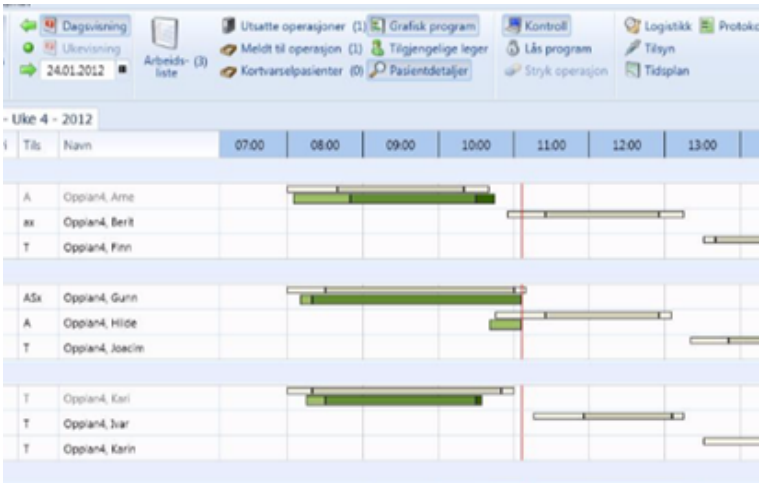


Figure 3.3: The timeline in OpPlan showing both planned and actual operation time. The red line indicates current time.

When excluding emergency operations, the operation plan for each day is scheduled in advance. When scheduling operations patients are added to an operation theater and data is inputted. This includes planned times for the phases

of the operation along with diagnosis, planned surgery and other information.

A lot of information regarding the operation is maintained in OpPlan. Different hospital employees will input and use different parts of it. Some of the tabs that are accessible are patient information, the anesthesia tab, the operation tab, the equipment tab, the crew tab and a tab that shows postponements. These are shown in Figure 3.4.

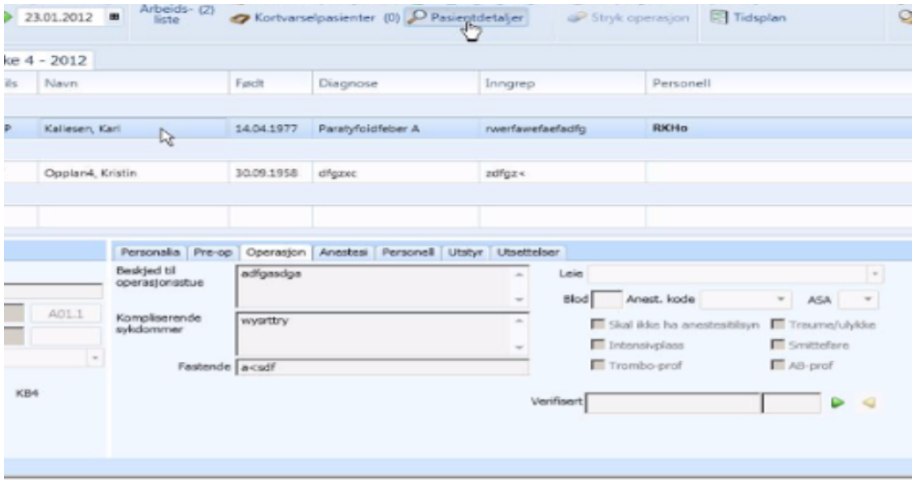


Figure 3.4: A screenshot showing the tabs with details for a selected operation.

3.1.3 Technical Architecture

The system architecture for the latest version of the OpPlan system, OpPlan 4, can from a simplified point of view be segmented into three layers, see Figure 3.5. At the lowest layer we have the databases, both local for OpPlan specific information at a particular hospital or region, and global for storing general patient information. Above this is the application service layer, OpPlanWas, built using the Windows Communication Foundation (WCF) to handle concurrent support for over a thousand clients with real time updates.

At the top layer are the actual OpPlan clients, built using Microsoft's .Net framework. These clients, to ensure that the critical information stored in the databases are not subject to incorrect handling, do not access the databases directly, but go through the WCF layer in all their actions. This includes the functionality of OpPlan to export data to other systems such as: MRS Posi, Sony Picis and the DocuLive Journal System through the OpPlanWas layer.

In addition to OpPlan WAS there is another application service layer for accessing the global patient information from the Patient Administration System (PAS), holding information about all patients that have a relation to the

hospital. OpPlan clients also access this information through the WCF layer. For securing access to all this data there is also an IAM layer, granting user access upon request through use of a token that is later verified in the WCF layer before access is actually granted.

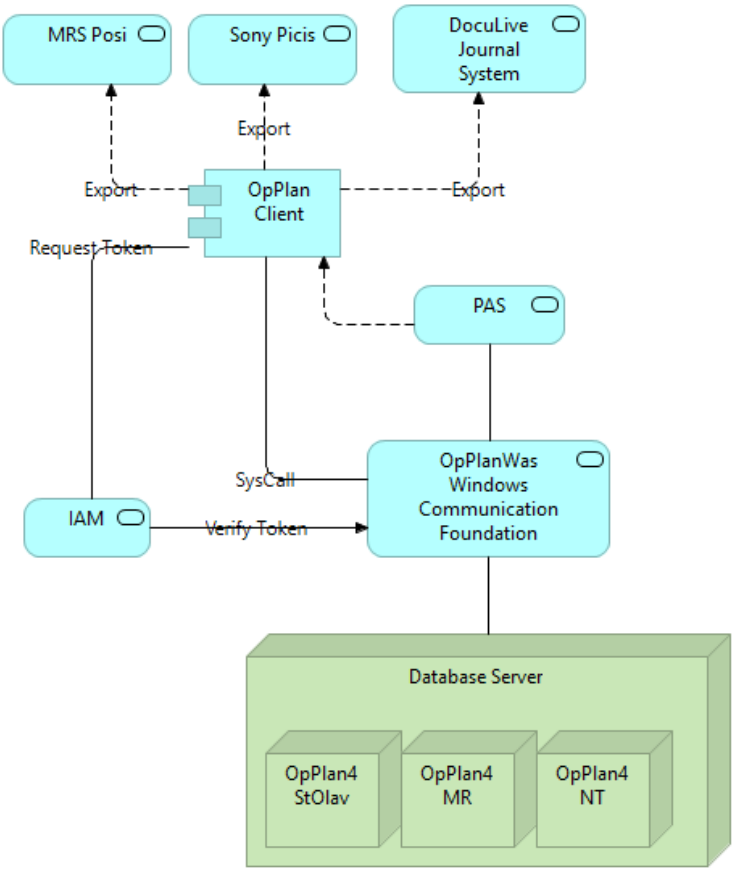


Figure 3.5: OpPlan system architecture. Here the blue boxes signify services and clients while the green signify database systems. The full drawn lines and arrows signify direct communication or system calls, while the stippled arrows signify indirect links that go through OpPlan WAS.

Access permissions are generally structured into specific roles pertaining to the occupation of the person requesting access. These limits both the observable and modifiable information of a specific plan so that for example a doctor or nurse in the anesthesia department only has access to relevant information, while a coordinator with extended access can see and modify any details. A diagram depicting this can be seen in Figure 3.6.

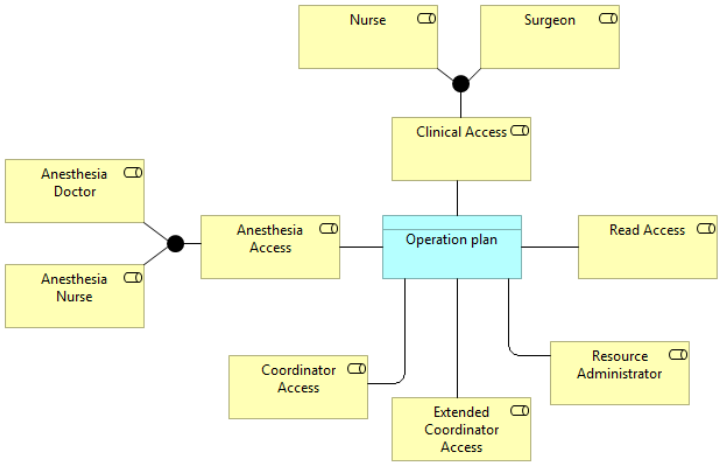


Figure 3.6: Model displaying all the different levels of access to an operation plan in OpPlan. Some typical occupations, such as anesthetic and surgical nurses and doctors, are grouped into anesthetic and clinical access respectively.

Operation plans, or programs, form the basis for the actual front-end views. These plans are largely based upon the geographical region of where they are to be used, or department based if the size of a hospital permits it. Each plan consists of a set of operation theaters in which surgeries are scheduled to be held. Each of these theaters may show in several plans, depending on the scheduled surgeries, and have associated with them one or more patients. In addition to theaters there also exist emergency lists, accessed similarly to theaters, but behaving a bit differently as the patients here are not assigned to an actual theater. Each plan only has one active emergency list, a list that corresponds to the plan that can be modified by someone with access to that particular plan. However, one may also have read access to other emergency lists not subject to the accessed plan.

As the scope of OpPlan is quite large, this leads to large databases with quite a number of relations tying operations to patients, crew, theaters and so forth. In Figure 3.8 is a simplified view of an ER diagram displaying these relations, only showing the names of tables and not their internal values.

Of particular note here are the relations going into the operation table in the top left. These tie the operation to anesthesia details, theater and hospital (through OperationUnit) on the right, as well as the operation crew members and operation events below. There are a large number of other relations here that will not necessarily tie directly into the project.

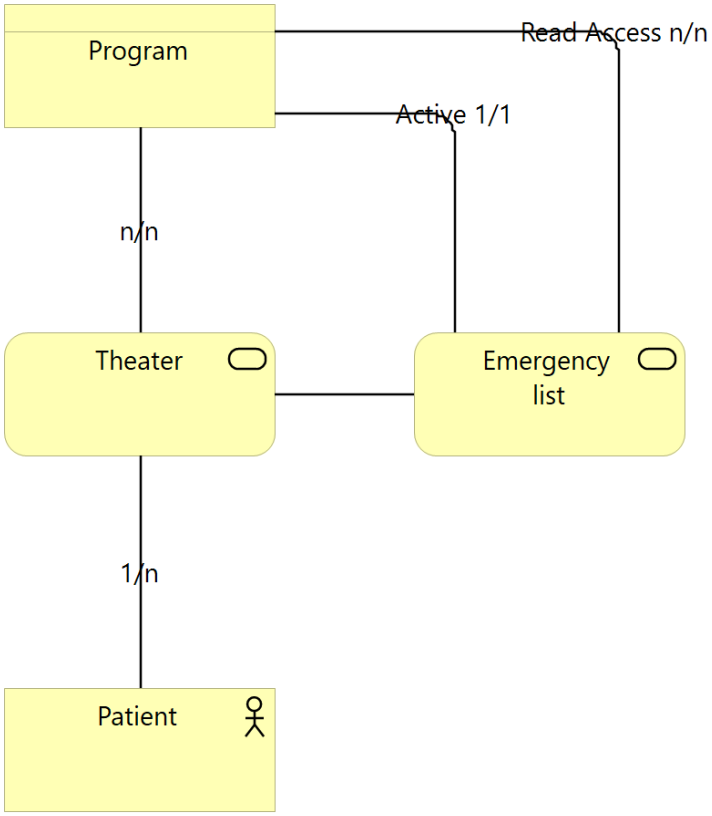


Figure 3.7: Model showing a hierarchical view of patients in relation to theaters and operation plans. The emergency list is related to the theaters in that they may show similar data, but are handled somewhat differently within a program.

3.2 Technical Solution

3.2.1 Requirements

The problem stated that the solution should be a mobile application to be used when the user did not have access to a desktop computer. The main challenge was to find the subset of functionality in OpPlan that was interesting in situations when a desktop was not available, and how to present that functionality. Research on this subject was done in four ways.

The first step was to hold interviews with users of OpPlan. This included doctors and nurses within the surgical and anesthetic departments, as well as department managers and coordinators. From these interviews a member of the team also got an opportunity for the second course of action; to follow an

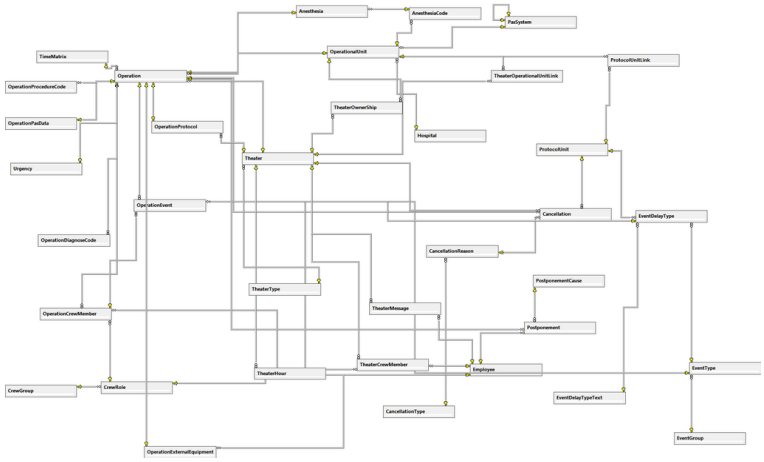


Figure 3.8: ER diagram displaying entity relations within the databases used by OpPlan

anesthetic doctor for a few hours during one of his work days and experience the existing system in use.

The third step was to create a questionnaire in Google Forms, through which the team received 61 responses from employees located at several hospitals in central Norway. After performing this research, the team worked out which functionality was most desirable.

Finally, as a fourth step, the team set up usability tests with three different users; a surgical nurse, a surgeon and an anesthesia doctor. This was done to make sure that the functionality implemented by the team were satisfactory from the user's perspective. Several changes were made after the usability tests. The final requirements decided upon are presented in Table 4.1.

3.2.2 Potential Solutions

The team considered it unnecessary to research existing technical solutions. Any such solution would still have to be modified to fit into the OpPlan infrastructure, which would likely be more work than developing a system from scratch.

With these constraints there were still technological decisions to be made. There exists several ways to implement a mobile application. Alternatives the team had to consider were:

- Pure native application: Choose between Android or iOS and implement using their platform specific SDK.

- Native web wrapper: Choose between one of the many frameworks for writing JavaScript in a native application, such as React Native or Ionic
- Pure web application: Choose between any front end web framework and design the website with a mobile-first focus.

3.2.3 Evaluation Criteria

Below the set of chosen evaluation criteria are briefly described. Performance:

- The application should be fast and responsive so that information can be extracted quickly

Target group cover:

- The application should cover as much of the target group as possible. This is mainly related to mobile operating system and browser

Flexibility:

- The application should be easy to modify to new use cases

Ease-of-development:

- The application should be as easy as possible to develop and maintain

Requirements fulfillment:

- The application should be light-weight and include the most important information in OpPlan

3.2.4 Conclusion

The three different alternatives were considered based on the evaluation criteria, see Table 3.2.

	Performance	Target group cover	Flexibility	Ease-of-development	Requirements fulfillment
Native					
Web wrapper					
Web					

Table 3.2: The degree of which the various platform choices fulfill the different criteria, on a scale where green indicates fulfilled and red indicates not fulfilled.

A native application would deliver best performance as it would run directly on the platform. However it would restrict the application to one platform and shrink the target group cover. Additionally it would be hard to evolve the application into new use cases. The biggest problem with developing a native application was the ease-of-development. Since the team members work on several different operating systems, both on the computer and on the phone, there would be challenges in developing for any platform.

Using a web wrapper was considered to be slightly better than native, overall. There would be a drop in performance, especially since the application presents itself as a native application and would then be compared to natively running apps. Target group cover is much better, as the app could cover several operating systems. Flexibility remains below perfect as the application would still be written specifically for phones. Ease-of-development is better.

Ultimately, the team chose to go for a pure web application. The performance is still not as good as on a native app, but still perceived better than on a web wrapper since users are used to web applications. The customer agreed on the implementation as an pure web application as long as it is focused on the mobile aspect.

The main concern with a web application was in the requirements fulfillment, as accessing hardware and other features of the smartphone outside the browser is harder in a webapp, with slightly less options than a fully native application. For example biometric sensors, accelerometer and NFC is unavailable in a web app (Dascalescu 2016). However, thanks to recent developments within progressive web apps, many of these issues are solved and it is possible to set up for example notifications for a web application. In addition a web application has the potential of being used on a computer and, using some underlying framework, the option of being converted to a native application down the line. As a result of this the team settled on the choice of a progressive web application.

3.3 Information Gathering

Various observation techniques were used to define the app's context of use. The goal was to get an understanding of who the users are, what they need and in which situation they need it. Information was gathered through interviews, a field study and a questionnaire. These techniques have different strengths and weaknesses, and altogether they gave a deeper understanding of the problem. The information gathered was represented through personas, which made it easier to understand the users and was used to form the requirement specification.

3.3.1 Interviews

The team wanted to get in contact with as many different hospital employees as possible, but experienced that getting in contact was difficult as people working at the hospitals typically have very busy schedules. It had to be done on their premises. The team was still able to arrange interviews with three of the main potential users of the application; an anesthesia doctor, a coordinator and a surgical nurse.

Some interview questions were planned beforehand, but it was just as important to be open about the interview subjects' thoughts and ideas, thus the interview was semi-structured (Preece, Rogers, and Sharp 2015 p.234). Advantages of doing interviews are that it gives a deeper understanding of a particular user and this user's needs, the conversation can be lead to obtain the knowledge needed. Disadvantages is that it takes time and it is not possible to know if this user's needs reflect what other users of the same occupation want.

The interviews were done at the interviewees' own offices. This had several advantages. People working at the hospital are usually very busy, so by being flexible with time and place, it was easier to schedule the interviews. By doing it in their office, it was also possible for them to show the team exactly how they interact with OpPlan during their working days, as they had computers with OpPlan running. It is not possible to run OpPlan outside of the hospitals. This made it a lot easier to talk and understand each other. Another advantage is that the interview subjects are more relaxed in their own environment, as they do not have to worry about who other people are or interact in an unnatural environment (Preece, Rogers, and Sharp 2015, p.242), this makes it easier for them to talk.

Interview questions:

- Age, gender, position, education.
- How is your understanding of technology?
- Can you describe an ordinary day?

- Can you describe a difficult day?
- What annoys you the most with OpPlan?
- How often do you use OpPlan?
- For what tasks do you use OpPlan the most?
- What other tasks are important for your work?
- For what tasks do you always have easy access to a computer?
- What tasks could you benefit from doing while not having access to a computer?
- Do you see the need of a mobile application?

3.3.1.1 Interview with Anesthesia Doctor

The first interview was done with a 39 years old male who is an anesthesia doctor and has worked at the hospital for 3 years. The interview subject had a very good understanding of technology as he had studied computer science.

How hectic his days are depends on if he is on call or not. Being on call means working in the emergency department. On a regular day, his day is predictable. When he has to read information or edit data about a patient, he does it in his office. On the other hand, his days are very unpredictable when he is on call. He uses OpPlan more frequently. The plan changes a lot because new patients arrive to the emergency and there is a lot of verbal flow of information.

The most useful feature for him is the overview of the state of all the theaters, as seen on the left in Figure 3.2. Important information for him is shown in Figure 3.9 including the symbols to the left, which the doctors are well familiar with the meaning of, and the 'tils' column which shows whether anesthesia has looked at the patient and formulated a plan. It would be useful for him to look at this while walking between two patients, so that he could get quick information about what he can expect.

When hovering over the i-symbol in column three in Figure 3.9, a popup shows up, shown in Figure 3.10. This popup contains a summary of text fields and most of these fields are useful for an anesthesia patient view, as well as the information in the anesthesia tab shown in Figure 3.11. The anesthesia tab contains the data that anesthesia doctors edit in OpPlan. The text fields are too long to edit on the go, but editing small fields and check boxes might be useful.

There is an emergency list included in the overview of theaters, which shows patients that have been put up to surgery but have not yet been assigned to a theater. This is usually where he finds patients who need to be looked at by

Figure 3.9: The symbols and information in the "tils" column are useful to have in a mobile application.

Figure 3.10: The popup showing when hovering over the information symbol. Shows a summary of important text fields.

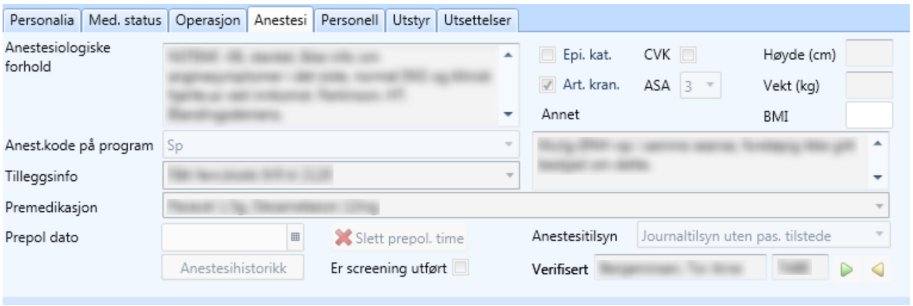


Figure 3.11: The anesthesia tab in OpPlan with the data fields that anesthesia doctors edit.

him. Sometimes the coordinators will sneakily place a patient there and not tell him until hours later when they call to angrily ask why he has not looked at the patient yet. Because of this, he thinks it would be helpful to get a notification when a new patient is added.

To summarize, the anesthesia doctor's priority in a mobile application is to have a quick overview of patient details while he is on the run between different patients. Usually, it is not convenient to edit data while on the run, but editing check boxes might be useful. Getting notifications when new patients arrive would also be useful.

3.3.1.2 Interview with Coordinators

The team also arranged an interview with two coordinators who have worked at the hospital for about 10 years and have education as surgical nurses.

The coordinators will always keep track of the time, looking at the timeline in OpPlan to monitor the progress in each theater. In OpPlan they are able to see when a patient arrives, and the color coding shows why it takes time between each operation phase. They use the timeline to see if they are on time and able to finish today's program. Both planned and actual time are important to them, as the planned time shows how much delay there is to an operation.

The coordinators think the application would be more useful for doctors. Coordinators usually have access to a computer where they do tasks that would be too complex and cumbersome to do on a small screen. The coordinators experience that doctors call them a lot to ask about their own schedule, so the app would indirectly ease the coordinators' day if doctors were able to view their own schedule on their phones, instead of calling.

Another task that would be useful for doctors, surgeons and other crew is to be able to register when they enter new phases of an operation. This will make sure that coordinators always are updated about what is happening.

3.3.1.3 Interview with Surgical Nurse

The last interview was done with a surgical nurse that has worked at the hospital since 2008. This interview subject also has a genuine interest in what HEMIT is doing and is designated by Rolf Holte to help him with testing at HEMIT. Because of this, he has a good overview of different types of users of OpPlan, and the talk considered what these users need, according to his studies. This interview was a little shorter than the other interviews, as he is very busy. During the interview, a list was made that contains the different occupations and the functionality that is considered most useful to them.

Surgical nurse:

- When they are finished with a patient, they want to be able to get a quick overview of the next patient in the theater. The information wanted is typically the diagnosis of the patient and what the status is
- If they are registered to a theater, they want to be able to see only what is going on in that theater. An surgical nurse is usually only assigned to one theater
- Get overview an of the timeline
- Subscribe to a patient and get notification when the patient state is changed. E.g. when a patient has arrived
- Click on a theater and call it
- Want to register data points for patients on the phone. This is now done on a computer

Anesthesia nurse:

- Register phase changes for patients
- A view of newly added patients
- Get notification on IP telephones

Surgeon:

- Have an overview of the surgeries they are going to perform during the day
- Want to see their schedules for the day and plan lunch breaks

He also mentioned what a room view and anesthesia view in the app should contain. A summary follows:

Room view:

- Ratio between planned and actual spent time. Useful to see if they need to reschedule if an operation is taking too much time
- It's important that time points are more accurate than whole hours
- Know what kind of operation is in the timeline. The timeline of each operation should include text of the diagnosis of the patient

Anesthesia view:

- Tab of newly listed patients

From this interview, the most important things learned is that the surgical nurses usually are assigned to one theater, which means they often also have access to a computer. Still, the main views that would be useful for them to have in a mobile application are an overview of the time matrix and some patient details. This will help them get a quick overview when needed throughout the day.

3.3.2 Questionnaire

A questionnaire was sent out in order to collect bigger amounts of data than what it is possible to do with interviews. The questionnaire can be used to confirm the conclusions from the interviews (Preece, Rogers, and Sharp 2015, p. 244). An interview is usually done with individuals and the questionnaire can be used to see whether or not others in the same occupation as the interview subject have the same needs. The team can also discover needs that the interview subject did not mention.

It was decided to do the questionnaire web-based on Google Forms, because web-based questionnaires have a fast response rate and automatic transfer of responses into a database for analysis (Preece, Rogers, and Sharp 2015 p. 249). By doing web-based questionnaires, it was possible to conduct answers from OpPlan users outside of St. Olavs Hospital, as OpPlan is used at all hospitals in Central Norway.

In order to gather the information needed, the questions must be well formulated. The team and the customers cooperated with the creation of questions. The customers had more experience within this field of study, so they helped to make sure the questionnaire made sense to the people receiving it and that they had the right alternatives to choose between.

In order to gather as many responses as possible, it was decided to send out the questionnaire in HEMIT's name to their contacts at the hospital. This was done because respondents are more likely to answer when they consider it interesting, of value and well presented (Burgess 2001). For these respondents, HEMIT has more trust than a student project and makes the questionnaire look more serious. It was sent out with a request to forward to other relevant hospital employees.

3.3.2.1 Questions

The questionnaire consists of both open and closed questions. The closed questions are useful for statistics about the user groups, but it is also important to ask some open questions so that the users have the opportunity to express

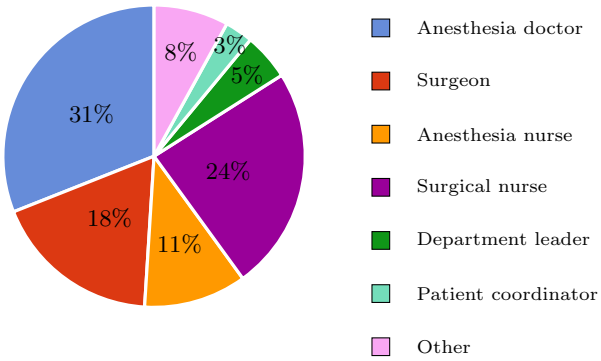


Figure 3.12: Distribution of occupations.

their ideas, if they want. At the same time, it was important to avoid the mistake of asking too many questions, as this can contribute to a poor response rate (Burgess 2001). To make sure this did not happen, the open questions did not require an answer.

The questions can be viewed in appendix B.

3.3.2.2 Responses

Before sending out the questionnaire it was decided that the goal was to conduct at least 50 responses. This goal was sat due to people working at the hospital usually are very busy. The questionnaire got responses from 61 people at HEMITs hospitals. This result is satisfactory, because the respondents were of different genders, ages, occupations and from different hospitals. Having this is mind, the most important user groups were well represented.

3.3.2.2.1 Gender 52,5% female, 47,5% male. Both genders are well represented.

3.3.2.2.2 Occupations The best represented occupations were anesthesia doctors (31,1%), surgical nurse (24,6%), surgeon (18%), anesthesia nurse (11,5%), see Figure 3.12. There were also responses from three department leaders, two coordinators, two general nurses, one orthopaedist and one secretary.

3.3.2.2.3 Hospitals All the hospitals are represented, 49,2% of the responses were from other hospitals than St. Olavs, see Figure 3.13.

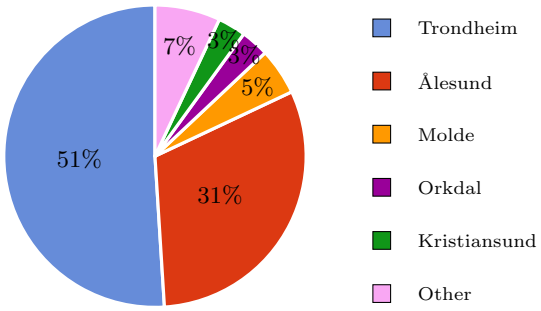


Figure 3.13: Distribution of work places.

3.3.2.2.4 Age 55,7% of the responders were 30-50 years old, 44,3% over 50 years old.

3.3.2.3 Analyzing data

Responders were grouped by their occupations to examine whether or not the groups use OpPlan in a similar way and if they have the same needs within the group. The data was also used to confirm or decline conclusions from the interviews.

3.3.2.3.1 Anesthesia Doctors As seen in Table 3.3, most anesthesia doctors do not spend a lot of time by a computer each day. Most of them use OpPlan for 2 hours or less and all of them use OpPlan for short intervals. In this group is usual to use OpPlan to get an overview of and complete tasks within the same day, use it for orientation, checking progress and for some documentation. 26,3% see the benefit of a mobile application right now, and a lot of them are unsure.

	Anesthesia doctors
Number of responses	19
Question	**Responses**
How much time do you spend by a computer per day in your job?	- 0.5-2 hours: 68.4% - 2-5 hours: 26.3% - More than 5 hours: 5.3%
How much time do you spend using OpPlan during a day?	- Less than 0.5 hour: 10.5% - 0.5-2 hours: 84.2% - 2-5 hours: 5.3%
How do you use OpPlan?	- Short intervals: 100% - Passive use/reading (OpPlan in the background): 26.3% - Long continuous periods: 10.5%
What are you using OpPlan for?	- Get an overview of tasks: 84.2% - Get an overview of and complete tasks for today ("dagen i dag"): 89.5% - Time scheduling: 15.3% - Lookup of orientation and progress: 68.4% - Schedule patients ahead of time (admissions office): 21.1% - Documenting: 47.4%
Do you see the benefit of a mobile application as a supplement to OpPlan?	- Yes: 26.3% - Maybe: 26.3% - No: 31.6% - Don't know: 15.8%

Table 3.3: Questionnaire response statistics about anesthesia doctors.

Wanted features in a mobile application for anesthesia doctors:

- A quick and easy overview of operations with clinical info about patients and progress
- See key info (for anesthesia doctors) for each patient
- Notification about operations ending soon, to start preparing for the next one
- Possibility to fill in and change the anesthesia part of the operation
- Replace operation scheme on paper with a digital version
- Being able to follow the operation theater you have responsibility for in detail

Surgical nurse	
Number of responses	15
Question	**Responses**
How much time do you spend by a computer per day in your job?	- 0.5-2 hours: 20.0% - 2-5 hours: 60.0% - More than 5 hours: 20.0%
How much time do you spend using OpPlan during a day?	- Less than 0.5 hours: 13.3% - 0.5-2 hours: 40.0% - 2-5 hours: 26.7% - More than 5 hours: 20.0%
How do you use OpPlan?	- Short intervals: 60.0% - Passive use/reading (OpPlan in the background): 6.7% - Long continuous periods: 46.7%
What are you using OpPlan for?	- Get an overview of tasks: 66.7% - Get an overview of and complete tasks for today ("dagen i dag"): 80.0% - Time scheduling: 33.3% - Lookup of orientation and progress: 73.3% - Schedule patients ahead of time (admissions office): 46.7% - Documenting: 66.7%
Do you see the benefit of a mobile application as a supplement to OpPlan?	- Yes: 26.7% - Maybe: 40.0% - No: 20.0% - Don't know: 13.3%

Table 3.4: Questionnaire response statistics about surgical nurses.

Conclusions from the interview with the anesthesia doctor are that he is usually on the run and would like to have a quick overview of operations and patient details and a possibility to edit small data fields. The data collected from the 19 anesthesia doctors can confirm that they use OpPlan in a similar way; they use it for short intervals to have an overview of tasks and progress and do not spend much time by the computer during a day. Some of the anesthesia doctors also reported that they want the same feature; an overview of operations and patient details, and some editing of data.

3.3.2.3.2 Anesthesia nurse 7 anesthesia nurses answered the questionnaire, but none showed any interest in having a mobile application. None submitted any wanted features and 85,7% did not see the benefit of having a mobile application, while 14,2% did not know. Because of the low interest, this user group was not analyzed further.

3.3.2.3.3 Surgical nurse As seen in Figure 3.4, it varies a lot how much the surgical nurses use OpPlan. As most of them use the computer 2-5 hours a day it seems like this user group spends more time by the computer than anesthesia doctors. This could be because they usually are assigned to one theater for a whole day. Both long continuous periods and short intervals are usual. What tasks OpPlan is used for also varies a lot, but most use it for getting an overview and for completion of tasks.

Wanted features in a mobile application for surgical nurses:

- The timeline
- Logistics overview
- Notification when a patient arrives to a preparation room
- Possibility for quick registration of time
- The content in patient details

After the interview with the surgical nurse, it was concluded that the most important views for surgical nurses is the timeline as well as patient details. This is also what surgical nurses from the questionnaire reported. Also, registering of the time when an operation changes phase could be useful to include.

3.3.2.3.4 Surgeon The surgeons answering the questionnaire do not use OpPlan for long continuous periods and none of them use OpPlan for more than 2 hours each day. It is most usual that they use OpPlan to get an overview of tasks, complete tasks for today and lookup of orientation and progress. Surgeons are generally very positive about having a mobile application, 63,6% answered they see the benefit and none are negative.

Wanted features in a mobile application for surgeons:

- See when a patient is called for
- See when anesthesia is done
- Possibility to see the equipment field
- Show personal plan
- Login with touch ID
- Overview of operation program and progress

Surgeon	
Number of responses	11
Question	**Responses**
How much time do you spend by a computer per day in your job?	- 0.5-2 hours: 45.5% - 2-5 hours: 54.4%
How much time do you spend using OpPlan during a day?	- Less than 0.5 hours: 27.3% - 0.5-2 hours: 72.7%
How do you use OpPlan?	- Short intervals: 90.9% - Passive use/reading (OpPlan in the background): 45.5% - Long continuous periods:
What are you using OpPlan for?	- Get an overview of tasks: 100.0% - Get an overview of and complete tasks for today ("dagen i dag"): 45.5% - Time scheduling: 9.1% - Lookup of orientation and progress: 72.7% - Schedule patients ahead of time (admissions office): 27.3% - Documenting: 27.3%
Do you see the benefit of a mobile application as a supplement to OpPlan?	- Yes: 63.6% - Maybe: 36.4%

Table 3.5: Questionnaire response statistics about surgeons.

Since it was not possible to get in touch with a surgeon for the interviews, this data will rather be compared with the studies from the surgical nurse who has a knowledge of surgeons' work day, see 3.3.1.3. From his point of view, the most important views for the surgeons is to have a personal plan where they can track their own surgeries for a day, and also use it to schedule a lunch break. The responses from the questionnaire can confirm that surgeons want to be able to view such a personal plan, and also other details about patient and anesthesia, as well as the timeline. This user group is interesting, since they are positive about the idea. Because of this, some surgeons will be contacted for usability tests, where it might also be possible to have a talk to understand them better.

3.3.2.3.5 Coordinators The questionnaire only received answers from two coordinators, so it was too few to make useful statistics about the user group. Both of the coordinators use OpPlan for long continuous periods, which confirms what coordinators from the interview said. They use OpPlan for various tasks, including time scheduling and documentation, which might be cumbersome to do on a small screen. Both might see the benefit of having a mobile application. One of the coordinators also suggested that the mobile application should have a personal plan, showing an overview of operations for each doctor, both for today and coming days. This is also something the coordinators from the interview suggested.

3.3.2.3.6 Department leaders According to nurse and product owner Rolf Holte, department leaders are employed as managers, but can also have coordinator tasks. The questionnaire received answers from three department leaders, which are too few to determine meaningful statistics surrounding the user group. The department leaders had some suggestions for features that the mobile application should contain:

- Read the timeline
- Read ASA group and anesthesia type
- Move patients between operation theaters
- Read patient details
- Change between operation programs
- Calendar with limited information
- For doctors it would be helpful to have access to operations they will work on

These features were considered to be added to the mobile application. It is interesting that also department leaders think it would be helpful if doctors had a personal plan with their own operations.

3.3.2.3.7 Others Two general nurses, a secretary and an orthopaedist also responded. There is not enough data to draw any conclusions, but they suggested some features that they think the mobile application should contain:

- Overview of patients
- Progress in the operation departments
- Possibility to change data
- Notifications before operations

3.3.3 Field study

A field study was arranged with an anesthesia doctor. The advantage of doing field studies is that it "can help fill in details about how users behave and use the technology, and nuances that are elicited from the other forms of investigation may be observed" (Preece, Rogers, and Sharp 2015, p. 253). The goal of the field study was to understand how anesthesia doctors use OpPlan during a regular work day.

One of the team members got to observe the leading anesthesia doctor working at the emergency department. The team member joined from the very start of the shift and got to follow along during anesthetization and wake-up. The team member stayed until the dinner break.

In the beginning of a shift the lead anesthesia doctor gets the emergency phone from the previous shift doctor. Then he goes to the anesthesia base. He starts by getting an overview of the situation in OpPlan, before he takes a quick look in the journal system to see if there are any small internally referenced jobs. Nurses and the assisting anesthesia doctor drops by and talks about today's situation. The oral sharing of information is essential according to the doctor.

When there is a task the anesthesia doctor is generally notified either by looking at OpPlan, face to face or through the emergency telephone. The prioritizing of tasks is generally done by surgeons and orthopedists.

In operation theaters there were three computers, two of which were running OpPlan. Nurses at in the operation theaters have different roles. This is important to keep it as clean as possible around the patient. Generally they are

divided into one sterile nurse, one that can access drawers and cabinets and one that can go outside of the operation theater.

During the field study the team member got to talk to many different people. They had a lot of thoughts and information to share. Some of the recurring thoughts about OpPlan were:

- It would be nice if there was an indicator for healthy patients
- The gaps between operation parts should be entered
- Changing between plans (centres) should be quicker
- People are not entering as much information as they should

Apart from the explicit information that was gathered, the intuition for the team member improved drastically. This was utilized during discussions and it was very useful when making the questionnaire and when planning the interviews.

3.3.4 Personas

Making personas is a way to represent the information gathered through the interviews, the field study and the questionnaire. Personas do not describe real people, but a synthesis from a number of real users who have been involved in data gathering (Preece, Rogers, and Sharp 2015, p. 357). In this project, the personas were used in order to understand what functionality the application should provide to ease their working days.

Personas were made for anesthesia doctors, surgical nurses and coordinators. A persona for surgeons was not created, because there was not enough data about this user group. The team only got information from the questionnaire.

3.3.4.1 Anesthesia Doctor

The persona is presented in Figure 3.14[1].

3.3.4.2 Surgical Nurse

The persona is presented in Figure 3.15[2].

3.3.4.3 Coordinator

The persona is presented in Figure 3.16[3].

[1] https://tinyurl.com/ycv295qz
[2] https://pixabay.com/en/doctor-dentist-dental-clinic-1149149/
[3] https://www.pexels.com/photo/woman-girl-sitting-computer-132681/

Arne

Age: 41
Occupation: Anesthesia doctor
Relationship status: Married, 2 children
Education: "Medisin profesjon" at UIO

"I'm always doing my best"

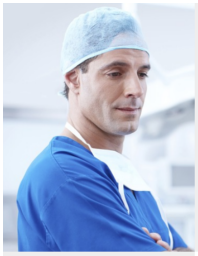
Understanding of technology:
Generally good. Arne uses different technology at work and also uses social media when he is off.

Typical working day:
If Arne is not on call, his day is predictable. There are not a lot of changes in the plan and he has time to do changes and stay updated in his office. Arne's days are a lot more stressful if he is on call, the schedule will change often and he uses OpPlan frequently. He gets a lot of verbal information.

How, when and where to give information:
Reads information on the way between different theatres. Wants to get a quick notification when a new patient arrives. He is busy and information needs to be easily accessible.

Concerns:
Having enough time to be updated on the next patient before meeting them. When the coordinators place a new patient in "akuttliste" and does not notify him, they call to angrily ask why he has not looked at the patient yet.

Figure 3.14: Anesthesia doctor persona.



Daniel

Age: 35
Occupation: Surgical nurse
Relationship status: Married, 3 children
Education: Bachelor of nursing at Nord university

"If there is anything, you will find me here"

Understanding of technology:
Good understanding, but low interest. Daniel uses his phone to send sms and call his kids, and he also uses a lot of different apps.

Typical working day:
He usually is assigned one or a few operation theatres and works there through the day, taking care of the patients coming in.

How, when and where to give information:
Daniel wants to get quick information about the next patient coming to the theatre he is assigned to, their diagnosis, status and get a notification when they're arriving.

Concerns:
Sometimes when he wants to call an operation theatre he uses a lot of time to find the mobile number.
He always wants to know the status of the next patient.

Figure 3.15: Surgical nurse persona.

3.3.4.4 Prioritization of Personas

Prioritization of personas was done in a meeting with the team and the customers present. It was agreed upon that Anita is of secondary concern, as she usually has access to a computer and is doing complex tasks that would be cumbersome to do on a small screen. Then the discussion revolved around Arne and Daniel. It was decided that Arne will be the highest prioritized user

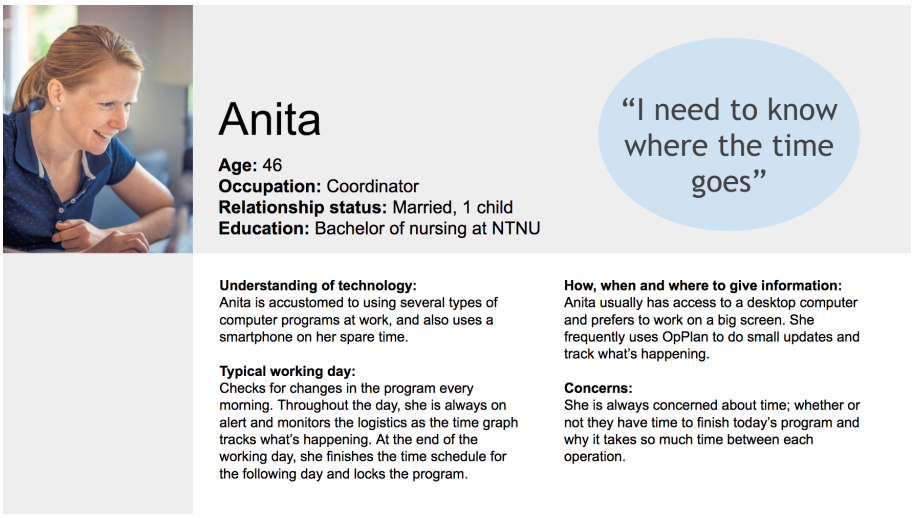


Figure 3.16: Coordinator persona.

of the mobile application, as he was the initial idea of the application from the customers' side, and he is also the one who is on the run, thus it would be most useful for him to have access to OpPlan while walking between operation theaters. Daniel is also considered interesting, so it would be good if the mobile application also covered some of his needs.

3.3.5 Conclusion

The team had to analyze the information gathered to decide what functionality was most useful for the users to have in the application. In this section, all the major functionalities suggested by the users will be considered.

3.3.5.1 Timeline

The timeline was the first functionality that was presented to the team by the customers as it is one of the major functions of OpPlan. It gives a good graphical overview of the operations and every user the team were in contact with were well familiar with the timeline. After having done interviews and the questionnaire, it was clear that the timeline had to be implemented in Scalpel.

3.3.5.2 Operation Details

Anesthesia doctors were interested in having access to information about patients while on the run, and this was confirmed from both the interview and the questionnaire. Surgical nurses want to have a quick overview of the next patient coming to their operation theater. All of this information can be presented

in an operation details view in Scalpel, where the most important information is presented first. It was decided that this is one of the most important views for the application, since it would ease the work day for the highest prioritized personas. A view like this is also easy to fit into a small screen and easy for the users to read, so it is a good functionality to have on a mobile phone.

3.3.5.3 List view

A list view is an alternative to the timeline to show operations. It is a list of operations sorted by their start time. In a list view, it is possible to show textual information that it is hard to show in the timeline view. Some of the users the team were in contact with mentioned that they would be interested in having a list view in addition to the timeline view. Different users have different preferences, so it would be nice to be able to choose what way to view operations. It is considered easy to show operations in a list view on a small screen, and it is not very difficult to implement it in addition to the timeline.

The team did not see any drawbacks to adding a list view. A big advantage is that more employees at the hospital might see the value of the application when they are able to view the information the way they want. Because of this, it was decided to implement a list view in the application.

3.3.5.4 Personal Plan

Various users have suggested to have a personal plan in the mobile application, where it is possible to see the operations they are assigned to. Coordinators, from the interview and the questionnaire, suggested that doctors should be able to view their own schedule instead of calling the coordinators to ask. The surgical nurses want to view what is happening in the theaters they are assigned to, and this is what would be shown in a personal plan. The surgical nurse interviewed, see 3.3.1.3, mentioned that surgeons want to have an overview of their own surgeries through the day, as was also confirmed by surgeons in the questionnaire. A department leader also suggested to have a personal plan.

Because of the high interest by employees of various occupations, it was decided that a personal plan is a view that should be implemented in Scalpel.

3.3.5.5 End Phases

Coordinators have mentioned that employees forget to end operation phases in OpPlan, so the timeline does not reflect what is actually happening in the operation theater. They have suggested to add a possibility to end the phase in a mobile application. This functionality can be added to the timeline view, and it would be easy for users to end a phase with just a few clicks. This is a functionality that is simple to implement and gives much value, and will therefore be implemented in Scalpel.

3.3.5.6 Notifications

There have been suggestions about getting a notification when a patient arrives to a preparation room, about operations ending soon and when a patient is added to the emergency list.

It was decided that this function would have a low priority because it would take a lot of time to investigate what kinds of notifications were useful and the notifications would have to be dependent on the occupation of the person receiving them. It is a lot of work, but would not make a significant change in the user experience.

3.3.5.7 Editing Data

Some have suggested to add a possibility to edit data in a mobile application. The anesthesia doctor wanted a possibility to edit small data fields, but thought bigger fields would be difficult to edit on a phone.

There are many drawbacks about editing data in a mobile application. Many of the users have said the screen is too small, so they would still use the desktop computer for these tasks. The customer's priority was on showing data rather than editing data. Because of this, it was decided not to add a possibility to edit data in Scalpel.

3.3.5.8 Calling Personnel

Surgical nurses have mentioned they want a possibility to call personnel, as calling is used a lot and is not possible to do from the desktop application. The possibility to call is easy to implement, easy for the user to do on a phone, and because of this, it was decided to add this functionality to Scalpel.

The suggestions for functionality made by the users are summarized in Table 3.6.

Suggestion	Addressed
The diagnosis of the patient and what the status of operations	Yes
See the operations you are assigned to	Yes
Show the timeline	Yes
Subscribe to a patient and get notification when the patient state is changed	No
Click on a theater and call it	No
Change operation phases	Yes
A view of newly added patients	No
View their own plan for the day	Yes
See both planned and actual time for operations	Yes
A quick and easy overview of operations	Yes
See anaesthesia information about each patient	Yes
Notification about operations ending soon, to start preparing for the next one	No
Possibility to fill in and change the anesthesia part of the operation	No
Replace operation scheme on paper with a digital version	No
Notification when a patient arrives to a preparation room	No
See when a patient is called for	No
See when anesthesia is done	Yes
Possibility to see the equipment field	Yes
Login with touch ID	No
Change operation plan quickly	Yes
Read ASA group	Yes
Move patients between operation theaters	No
Read patient details	Yes
The gaps between operation parts should be entered	Yes

Table 3.6: Summary of suggestions for functionality from the information gathering process

Chapter 4

Requirements Analysis

This chapter contains the specification of the requirements for the system. Also, to better visually present them, a use-case diagrams with different actors of the system were produced.

A big part of the task was to research which part of OpPlan's functionality that was suitable to put on a mobile device. Therefore, the customer did not present the team with a task with a full requirement specification for the solution. The customer had a different approach. HEMIT allowed the team creative freedom and options to explore the feature set for the application themself. This meant that the team had to do a lot of research to explore which features should be implemented, this is presented in Chapter 3.

The requirement specification was updated each time the team discovered a new feature the team and the customer found useful. The fact that the team did not have a requirement specification from the start meant that estimates for the user stories in the requirements specification could not be made in the outline planning phase.

First, a few high-level user stories were made. The application should realize those. Later, the team divided these tasks into smaller user stories that could be added to the backlog and added to each sprint.

4.1 Overview

The solution should be an application that works on mobile devices and has some of the already existing functionality of the desktop application OpPlan, along with some new features. It should mainly consist of presentation of data that is relevant and appropriate on a mobile device. The goal for the project

was to create a functional prototype. The final product would show confidential data and a back-end already exists for the desktop application. Considering this, it was not required to integrate the team's solution with real data from an actual back-end.

4.2 Clickable Prototype

A prototype-diagram for the application was made incrementally in an application called InVision. This was based on the agreed upon system requirements. During development of the prototype, it was regularly presented to the customers who gave the team feedback. In the early phases of the project, the prototype was very helpful for discussing and getting a better understanding of what a final solution should look like. When the prototype was done, the customers approved it and the application was developed based on it. Although the prototype was a good indicator for what the final application should look like, the team had to do some changes to it during development. The clickable prototype can be viewed in Appendix D.

4.3 Functional Requirements

The functional requirements are presented in Table 4.1.

These requirements were later divided into smaller user stories. These are presented in Table 4.2. For each user story there was assigned an id. This was to easier reference them in other parts of the project. An estimate for the amount of person hours it takes to finish the task was also made. In the last column the importance of the task was estimated.

4.3.1 Textual Use Cases

The team made some more detailed textual use cases for the requirements with more detailed information. These are shown in pairs in Tables 4.3 - 4.7 and Figures 4.1 - 4.5.

ID	Name	Description	Priority
B1	Operation timeline	As a user I want to view a version of the timeline view from OpPlan, which shows scheduled and performed operations distributed over a time axis. This should include both the planned time and the real progress of the ongoing operations	High
B2	Operation list	As a user I want to view a version of the operation list view from OpPlan, which lists scheduled and performed operations and their status	Medium
B3	Operation details	As a user i should be able to show details of an operation, such as patient data and current status	High
B4	Personal calendar	As a user i should be able to show a timeline for the selected day with the operations involving the currently logged on user. This includes both the scheduled times and the progress of the ongoing operations	Medium

Table 4.1: Functional requirements

ID	User story	Story points	Priority
A1	As a user I would like to see when each phase of an operation starts and ends	5	High
A2	As a user I would like to input when a phase I am involved in starts or ends	8	High
A3	As a user I would like to see the details of an operation	13	High
A4	As a user I would like to see all operations in a list view	8	High
A5	As a user I would like to have specific options in the header per view	3	Low
A6	As a user I would like to change between operation programs	3	Medium
A7	As a user I want to scroll horizontally through theaters in the bar view	2	Medium
A8	As a user I would like to view operations that overlap in time side by side to compare them	8	Medium
A9	As a user I would like to change the details of an operation	13	Low
A10	As a user I would like to easily switch between the day viewed for a specific operation program	3	High
A11	As a user I would like to view my plan for the day	8	Medium
A12	As a user I would like to see the age and the gender of the patient in the list view	1	Low
A13	As a user I would like to view a small timeline in the operation details	3	Low
A14	As a user I would like to get a tutorial the first time I log in	10	Low
A15	As a user I would like to login with a username and password	10	Low
A16	As a user I would like to see when each phase of an operation was planned to be in the timeline view	2	High
A17	As a user I would like to be able to call personnel directly from the detailed view	1	Medium

Table 4.2: User stories

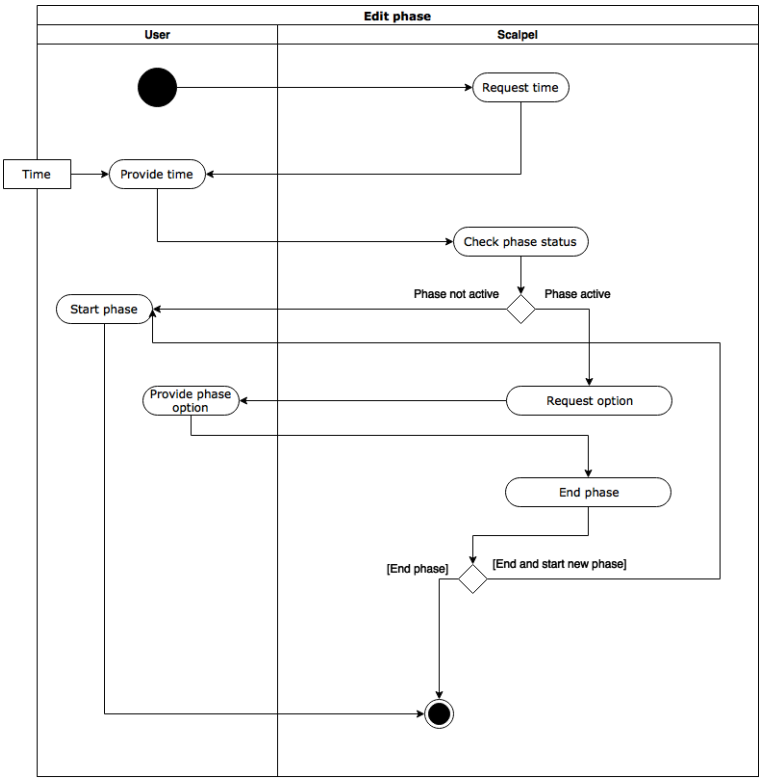


Figure 4.1: Use case A2

Name	As a user I would like to input when a phase I am involved in starts or ends
Description	After a phase in an operation, for example that anaesthesia is given, users want to register that this phase is complete. User long clicks on an operation, a pop up will emerge
Actors	Anaesthesia doctor Surgery nurse Anaesthesia nurse
Preconditions	- User is logged in to the application - User is in bar or list view - User clicks on a operation
Basic flow	1. User clicks on the time-picker field if they want to edit the time 2. User clicks on "Phase completed"-button 3. Popup closes
Alternate flow	User clicks on "Phase completed"-button and the current time is registered and the popup closes User clicks on "Start new"-button and the current phase is ended and set with the current time. If the operation has not started only the "start"-phase button is shown
Post conditions	The time of the completion of the phase should be registered in the system or the starting of a new phase shall be registered

Table 4.3: User story A2

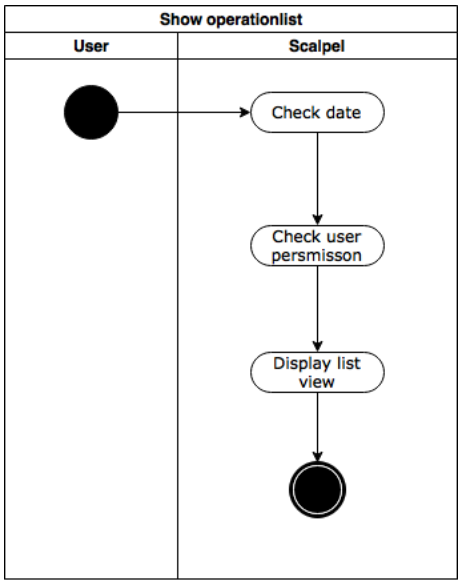


Figure 4.2: Use case A4

Name	As a user I want to view the operation list view from OpPlan, which lists scheduled and performed operations
Description	The list view should be a summary of the operations in the plan. It contains one tab for each theater. The most useful information should be displayed, the information will be the status symbols, the diagnosis and surgery. It should be sorted by operation time. If a list item is clicked a detailed view of the operation is shown
Actors	Anaesthesia doctor Surgery nurse Anaesthesia nurse Surgeon
Preconditions	- User is logged in to the application - User chooses list view in the header menu
Basic flow	1. User clicks list view mode in the menu 2. The list of operations is shown
Alternate flow	User clicks on list item and a detailed view of the operation is shown

Table 4.4: User story A4

Name	As a user I want to view the timeline view from OpPlan, which shows scheduled and performed operations distributed over a time axis. This should include both the planned time and the real progress of the ongoing operations
Description	The timeline view should be a summary of the operations in the plan. It should look like a gantt-diagram. It will show the actual and the planned time. The bars should be clickable to show more detailed info. A line with the current time will show. The first column will consists of time. The rest will have theater names on the top and bars with the planned and actual surgeries
Actors	Anaesthesia doctor Surgery nurse Anaesthesia nurse Surgeon
Preconditions	- User is logged in to the application
Basic flow	1. User opens the application or user chooses bar view 2. The timeline is shown

Table 4.5: User story A1

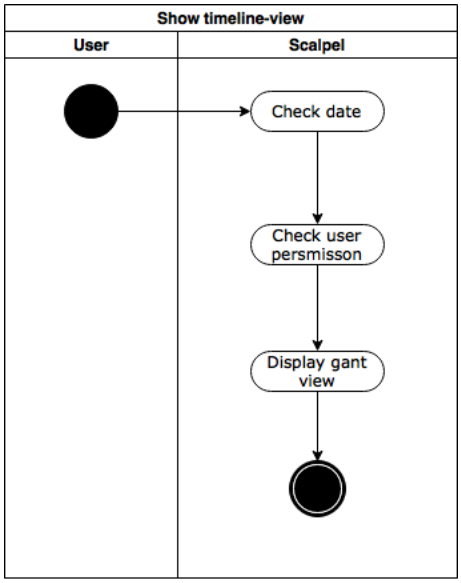


Figure 4.3: Use case A1

Name	As a user i should be able to show details of an operation, such as patient data and current status
Description	The operation details should contain several tabs with all relevant information about the selected operation
Actors	Anaesthesia doctor Surgery nurse Anaesthesia nurse Surgeon
Preconditions	- User is logged in to the application - User chooses an operation
Basic flow	1. User click on a operation in the timeline or in the list view 2. The operation details view is shown

Table 4.6: User story A3

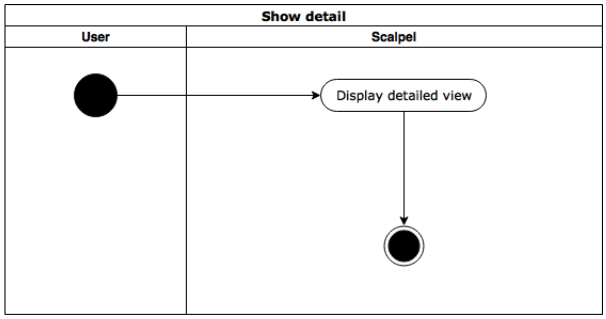


Figure 4.4: Use case A3

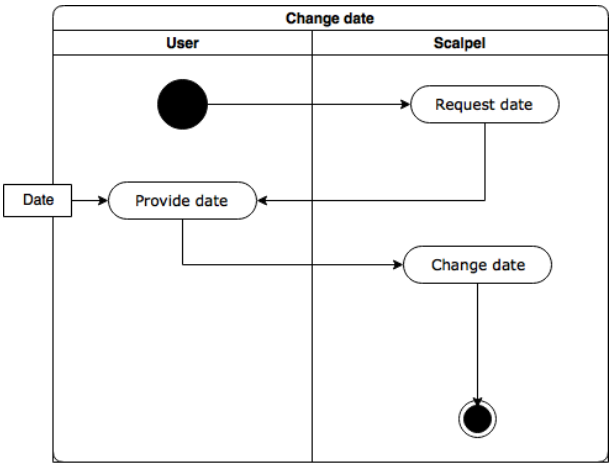


Figure 4.5: Use case A10

Name	Change date
Description	The user can change the date of the view to the past or the future for better planning. In the app header the user can press the date to changing it. A calendar is used for selecting day
Actors	Anaesthesia doctor Surgery nurse Anaesthesia nurse Surgeon
Preconditions	- User is logged in to the application
Basic flow	1. Listview or Barview is chosen 2. User clicks on the date 3. A calendar opens 4. User selects another date
Post conditions	The state of the application is updated to the selected date

Table 4.7: User story A10

4.4 Non-functional Requirements

4.4.1 Performance

The performance of an application is measured in how responsive it feels. When an item is clicked the application should quickly update. Further, the application should load quickly when navigated to. The team and the customers

did not set any goals or limits on performance, but usability testing should be conducted to reveal whether improvements are needed.

The desktop application has about 1000 simultaneous users, which means that the mobile application should be able to handle the same system load, and still be reliable and responsible. The team will make sure to build the application using scalable frameworks that can handle many simultaneous users. Load testing involving a lot of concurrent users should be conducted before releasing the application to analyze the system behaviour and reveal any issues.

It is important that the application does not stop working at any time. The desktop application has some trouble with crashing occasionally, which will require a restart. From the questionnaire the team understood that a lot of users of the desktop application were dissatisfied by that. During development it is not easy to reveal issues that might occur when the application is released, but the team will make sure to write high quality code and do necessarily testing to avoid producing bugs.

4.4.2 Usability

Usability is defined as how easy it is for a user to complete their tasks (Preece, Rogers, and Sharp 2015). While usability refers to the ease of performing tasks, it does not measure which tasks or the number of tasks that the application can be used to perform. To make sure to develop an application with high usability, it is important to have a user-centered approach with frequent user communication and feedback.

As described by Preece, Rogers and Sharp usability is often split into three aspects; discovery, learning and efficiency. Discovery refers to the ease of finding a particular task in the application. Learning defines how quickly the user understands how to perform the desired task. How long it takes to perform a task when it is learned defines the efficiency. To measure these things, a usability test can be performed. By giving a user a set of tasks to perform, the team can look at how easily the user discovers where a feature can be found and how many errors the user does before completing the task successfully. During development efficiency can be handled by limiting the number of 'clicks' required to perform tasks. The team will strive to have more frequently performed tasks more efficient to complete. Together with the customer the team can look at the results of usability testing and decide whether or not the they are satisfiable.

4.4.3 Maintenance

This project has a time limitation and the team will deliver a product that will be modified and developed further in the future. That means that what the team delivers must be easy to read and well documented for future developers.

Regarding code, it must be scalable, easy to understand and well documented. This will be addressed by defining a set of code style guidelines (see chapter 6). Further, this report will contain a lot of support documentation for the product, such as for example system architecture.

4.4.4 Security

Information used in the hospitals is highly confidential data. This means that security is a prioritized requirement for the product. Despite this, the main goal of this project was to create a functional prototype that can be connected to almost the same back-end as the desktop application is currently using. Thus, the team and the customers agreed that security issues did not have to be addressed during the project, but that proper handling of authentication is a minimum requirement before an actual release of the application.

Chapter 5

Tools and Technologies

5.1 Technology choice criteria

The team looked at several aspects when choosing which technologies to use. The main criteria was naturally to what degree it fit to the needs of the team. But another important aspect was the licensing and pricing of the technology. First, the team had limited financial capital so free solutions were always considered first. Then, open source solutions were also favored. This aspect was most important because open source software is favoured in security and testing due to its nature. As described in the paper "The Aspects of Choosing Open Source versus Closed Source" (Atieh Khanjani 2011) there is also no central management. That fits well to the project.

5.2 Management and Communication Tools

5.2.1 Trello [1]

Trello was used to manage the backlog and keep track of progress. Trello is an online tool that works in the browser and also native on Windows/Mac. The team members had to register. The team got three months premium for free, since the person inviting gets one month of premium for every new user. The main task in Trello is to make cards. People can assign themselves to cards, comment and add deadlines. This all is included in the free version. The premium features that helped a lot were planning poker, Github and Card Numbers. With these additions the team could manage the workflow even better.

[1] Website: `https://trello.com/`

5.2.2 Google Drive [2]

During the project the team produced a lot of content. The team needed a platform to store and share all the files produced.

5.2.3 Google Calendar [3]

The team used Google Calendar to plan meetings and schedule work times. Since all of the team members had a Google account, it could easily be integrated into their normal workflow. It also enabled easy access to the place a meeting should be, and how long it would last.

The team had some options to choose between. To store small files like meeting documents, design files and other things produced during the project the team concluded that a file sharing system operating in the cloud were needed. The different options were platforms like Dropbox, Microsoft OneDrive and Google Drive. Everyone were good file-sharing systems. OneDrive offered a lot of space through a deal with NTNU. But it lacked access through multiple interfaces due to the fact that it doesn't support Linux, which some members of the team used on their computers. Dropbox was considered but it does not have the a good enough concurrency solution. So the team chose to use Google Drive.

On google drive all of the documents that was produced during the project was stored. This included the report, protocols, research and hour records.

5.2.4 Slack [4]

Slack is a communication tool. An alternative is Facebook, but there are some drawbacks. On Slack the administrator can create infinity channels. People can talk to each other private and in every channel people can exchange data. In Slack the search function is advanced and helps to find what people are looking for. Another main part is the integration of application outside from Slack. In Slack there are many additions for other, big tools like Google Drive, DropBox and Github. These help to connect every information in a micro cosmos that bundle up every information for a certain project.

The team also got access to a premium channel within the HEMIT group on Slack from the customer. This removed the restriction of 10000 messages in the free version. Also for free, Slack would be the first choice of the team due to the great management of data.

[2] Website: `https://drive.google.com/`
[3] Website: `https://calendar.google.com/`
[4] Website: `https://slack.com/`

5.2.5 GitHub [5]

GitHub is a web-based Git or version control repository and internet hosting service, mainly used for code. It offers all of the distributed version control and source code management of Git as well as adding more features. Through GitHub users gain access control and collaboration features such as bug tracking, feature requests, task management and wikis. GitHub offers plans for private and free repositories on the same account, and is commonly used to host open-source software projects. With more than 20 million users and 57 million repositories as of April 2017 GitHub is the largest host of source code in the world. The team utilized GitHub to create repositories both for source code and architectural models, creating separate branches for each feature to work on collaboratively. This was an obvious choice due to the team members' previous experience with the tool, its reputation and overall ease of use.

5.3 Documentation Tools

5.3.1 Google Docs [6]

Google Docs is a common tool in the education sector. The team appreciated the ability to work on documents together in real time. This argument was also part of the decision process to choose this instead of other tools. The report was first written in Google Docs, as it is simple to navigate and easy to get comments from the advisor and customers. ShareLaTeX did not have as good concurrency handeling. Another big plus was that it is free. For managing all documentation in the project there were no notable alternatives in the market.

5.3.2 PBworks [7]

PBworks is a wiki tool that was used mainly as a lookup table for the resources used by the team. On the wiki the team gathered documentation on what technology that were used and created links to other wiki-pages to create a overview.

5.3.3 ShareLaTeX [8]

ShareLaTeX is a server based LaTeX editor to be used online in a browser, allowing for real-time collaboration and online compilation of LaTeX projects down to a PDF format. The site follows a freemium pricing model, adding premium features to paid plans. The team used ShareLaTeX for structuring the

[5] Website: `https://github.com/`
[6] Website: `https://docs.google.com/`
[7] Website: `http://www.pbworks.com/`
[8] Website: https://sharelatex.com

final version of the report, working under student licenses provided by NTNU to amongst other things avoid the limitation of maximum two collaborators per project.

5.4 Frameworks and Development tools

5.4.1 Visual Studio Code [9]

Visual Studio Code is a source code editor made by Microsoft to run on Windows, Linux and macOS. It is light weight, open source and highly customizable while offering support for debugging, embedded Git control, syntax highlighting, intelligent code completion, snippets, and code refactoring. Visual Studio Code was the team's editor of choice for programming the application, offering the option for some automated management of clean code. The editor also has a simple JSON format for configuring settings, so they could be added to the code repository. That made sure every developer had more or less the same environment for writing code, which was helpful.

5.4.2 npm [10]

npm is a package manager for JavaScript and the default package manager for the JavaScript runtime environment Node.js, which the team installed as a means using npm. npm consists of a command line client, also named npm, and an online database of public and private packages called the npm registry. The available packages can be browsed and searched through using the npm website, and installed through the command line client. The team used npm to install multiple packages and manage the dependencies the software solution relied on. Another commonly used alternative is Yarn, produced by Facebook in 2016.

5.4.3 React [11]

React is a JavaScript library for building interactive user interfaces for the web and mobile (through React Native). React targets to make it effortless to design simple views for each state in an application and efficiently update and render only the right components when the data changes. React is component based, allowing the user to build encapsulated components that manage their own state and compose them together into complex UIs. The team used React to develop a progressive web app, mainly targeted towards mobile users.

[9] Website: `https://code.visualstudio.com/`
[10] Website: `https://www.npmjs.com/`
[11] Website: `https://reactjs.org/`

5.4.4 Redux [12]

Redux is a state container for JavaScript applications. It creates a centralized store containing the state of the application, so there is a "single source of truth". It also limits the ways in which state can be manipulated, through actions and reducers. React works well with Redux, since Redux state and actions can be mapped directly to props in React components. In that way, the application no longer has to pass down properties and callback functions to child components, since they come directly from the Redux store.

Using Redux simplified development of Scalpel significantly, since Scalpel has a lot of data to manipulate and present. Once the state of the application was defined, each component could extract only the data they needed from the state, without having to be passed the data from its parent component.

5.4.5 Webpack [13]

Since a web application is hosted on the web, it is important that the files that are transferred are as small as possible. Therefore, it is normal in modern web development to use a bundler, that concatenates all code to one file, and minifies and uglifies it. Webpack is the most popular JavaScript bundler currently, and was included in the starter kit the team used to bootstrap Scalpel.

Another very useful feature of Webpack is its hot reload development server. It enables developers to immediately see the changes they have made in the browser, without having to reload the server.

5.4.6 Babel [14]

JavaScript is a language that changes quickly, but it takes a long time for new features to get good browser support. Many of these features makes development in JavaScript much more comfortable and efficient. In order to use them, they have to be translated into old JavaScript code with better support before it is used in the browser. The most common tool to do this is called Babel, and it can compile modern JavaScript code into older, better supported JavaScript. Babel can be used as a plugin for Webpack, so it can be integrated into the build process.

[12] Website: `https://redux.js.org/`
[13] Website: `https://webpack.js.org/`
[14] Website: `https://babeljs.io/`

5.4.7 Material-UI [15]

Material-UI is a resource for React Components that implement Google's Material Design. It is free to use with it's v1 version currently in beta, but older stable versions available. The team settled on using the beta version as it is a full rework and likely to support effortless updating to later full releases. Through the use of Material-UI the application was developed to conform with Google's Material Design guidelines and attempts to offer a service in a way that the user already has some familiarity with.

5.4.8 D3.js [16]

D3.js is a JavaScript library for producing dynamic, interactive data visualizations in web browsers, bringing data to life through use of HTML, SVG and CSS. With emphasis on web standards D3 gives the user the full capabilities of modern browsers without being tied down to a proprietary framework. The power of D3 comes from its ability to connect elements in the DOM directly to data points, so it can efficiently react to changes to the data. D3 was chosen as the solution for creating a dynamic, data-driven timeline for viewing scheduled, ongoing and completed operations.

5.5 Design Tools

5.5.1 Sketch [17]

Sketch is a tool to perform design tasks. It is specific tailored for the task of creating application and website screen designs. There are many free resources of UI design elements that can be used. Sketch is only available for Mac and cost regular 100$ but there is a student discount that offer the same product for 50$. Alternatives would be Balsamiq3 but the team decided to work with Sketch because it offers a way to create a more detail end product.

5.5.2 Affinity Photo [18]

Affinity Photo is a competitor of Adobe Photoshop. It is available for Mac and Windows and cost 50$. There is no student discount. It offers in a way the same toolkit like Photoshop. Where it differs is the interface. In contrast to Photoshop there are no monthly fees, that was the main argument. Affinity Photo is a tool to manipulate Photos and create design elements that can be used in Sketch.

[15] Website: `https://material-ui-next.com/`
[16] Website: `https://d3js.org/`
[17] Website: `https://www.sketchapp.com/`
[18] Website: `https://affinity.serif.com/en-gb/photo/`

5.5.3 InVision [19]

InVision is a prototyping tool. It helps to visualize the way of interaction with an app that was designed in Sketch. InVision provides an application for Sketch. With this app the team was able to create the prototype directly inside sketch. That improved the workflow a lot. The prototype was uploaded to the InVision infrastructure and could than be sendt to customers or end users. They could then straight deliver feedback. On the InVision website there were able to comment to prototype.

5.6 Modelling Tools

5.6.1 Archi [20]

Archi is a free and open source modelling tool, targeted towards all levels of enterprise architects and modellers, for creating models and sketches. Paired with the Model Repository Collaboration Plugin (currently in early access) this allows for collaborative work on ArchiMate models through sharing and versioning of models in a repository. The team utilized this tool for constructing technical models of the existing system architecture for OpPlan as well as for the designed solution, and for collaborating on these through the use of a git repository.

5.6.2 draw.io [21]

draw.io is a free online diagram editor built around Google Drive. It enables the creation of flowcharts, UML entity relation, network diagrams, mockups and more. The diagrams are then stored directly to Google Drive, with no additional third-party that stores the data. The team used draw.io to create the less technical diagrams, typically relating to user experiences and offering of features, such as use-case diagrams.

5.7 Research tools

5.7.1 Google Forms [22]

Google Forms is a great way of creating surveys to people. While the team already worked with Google Drive and Google Docs it was a no-brainer to choose Google Forms. It is also completely free. There are some alternatives but none is as well integrated as Google Forms. Another argument is that all members

[19] Website: `https://www.invisionapp.com/`
[20] Website: `https://www.archimatetool.com/`
[21] Website: `https://www.draw.io/`
[22] Website: `https://docs.google.com/forms/`

of the team can concurrently work together on surveys. It also helped a lot in regarding of sharing the survey. The customer could take a look at the surveys and if needed do changes without any technical problems.

Chapter 6

Design

During the design and work processes the team encountered different sources that changed the point of view regarding the design. The first step was to choose between different design guidelines. After that, every screen was designed in Sketch and was changed iteratively based on feedback and usability testing.

6.1 Design Choice

To design a software system there are many decisions that have to be made. One of them that needs to be addressed at the beginning is choosing the design language. Which design language the software system will follow is changing the way of interacting with it.

6.1.1 iOS and Android

There are two main design directions that an application can follow. The iOS design language from Apple (*Designing for iPhone X* n.d.) and the Material design language from Google (*Material Design* n.d.). To decide which language the project could the team did research and gathered information. These differences are listed in Table 6.1.

Table 6.2 shows that Android is much more common (*Smartphone OS Market Share, 2017 Q1* 2017). If the assumption is that more people use Android on a daily basis it is unambiguously the people can adapt to a new application easier if it is designed using Material design as Android is.

The numbers are quite clear in 2017, and is very unlikely that it will change a lot in the near future when looking at the market share of each operating system over the past few years as shown in Figure 6.1.

iOS	Android
Apple devices	Android devices
Consistent app design in the eco app system	More freedom of choice with the design elements
Only on IOS apps	Material UI elements are well implemented in React
	Is also web standard
	Can be and is used on IOS apps

Table 6.1: Differences between iOS and Android.

Year	Android	iOS	Windows Phone	Others
2017	85.1%	14.7%	0.1%	0.1%

Table 6.2: Smartphone market share.[1]

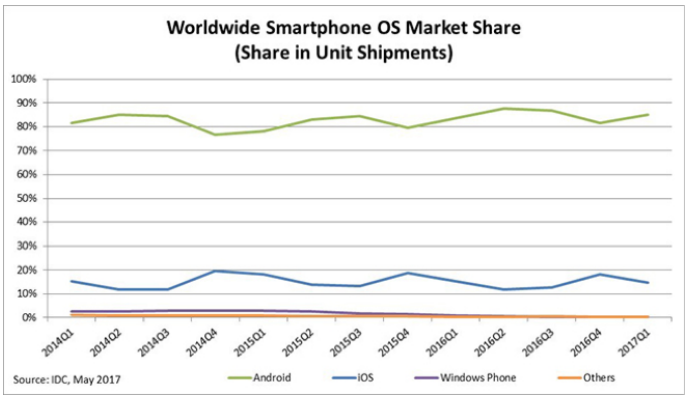


Figure 6.1: Graph depicting the market share of various mobile operation systems between 2014 and 2017 [2]

To design for other systems were not an option as Windows phone only have 0.1% of the market share and other phones have 0.1%.

6.1.2 Users and Their Habits

Regarding this topic, the team also performed a survey to gather more details about the users, the distribution of phone operating systems used is shown in Figure 6.2. The surprise was that 57% of the users that were asked use iOS. That is a big contrast to the world average. While on average only 15% of the users are using iOS, in the hospital nearly 4 times more were using iOS.

[1] `https://www.idc.com/getdoc.jsp?containerId=prUS42628117`

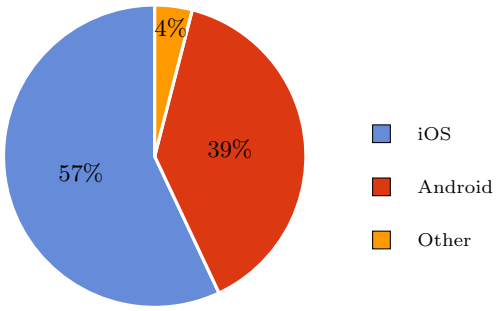


Figure 6.2: Distribution of phone operating systems.

The survey was answered by 61 hopital employees, while OpPlan can have up to 1000 simultaneous users. Thus the result might not be completely accurate. Further, in Norway the amount of iOS users are almost the same as the amount of Android users [2]. Many in the target group were wealthy people, which tend to buy more expensive products, such as Apple phones. No matter how the team chose to design the application, this was important information about the group of users.

6.1.3 Differences Between iOS and Material Design

Both design for iOS and Material Design have their own standards. The team would have to choose one of these to follow. Understanding the differences helps to create an application that feels natural to the user. Considering that the different users were used to different standards, it was important to look at both to make the application understandable no matter what background the user had.

6.1.3.1 iOS Design

iOS design 11 has no specific release date. It is the standard that Apple uses to create their own apps. It is also called iOS flat design. The colors are more muted and it is much more important to use white and grey colors. Blue is the accent color. It is common among designers to call the iOS design more minimalistic. Many designers are customizing iOS design a lot.(*Developing for Android vs. iOS: Material vs. Flat Design* 2016)

[2]Carlsen and Zakariassen 2014

[2]`https://www.idc.com/promo/smartphone-market-share/os`

6.1.3.2 Material Design

The design from Google was introduced in 2014 and is the standard for Google applications on the web and on Android devices. It has a bright color pallet. Shadows are used to create a depth or height. Further, it uses more square shapes than rounded corners. Google says that it is "inspired by the study of paper and ink" (*Material Design* n.d.).

6.1.4 Conclusion

Taking these numbers into account there are two perspectives that tells two different stories. The team had to decide right away which direction to take, as this had implications for the design and implementation from a very basic level.

While the iOS system is not compatible with Android, the Material Design is compatible with the iOS operating system. There are applications that use Material design in the iOS app store that are successful. The customer also had the wish to expand the functions of the application in a way that people can view the data also on a big screen.

iOS was more common in the environment that the application might be used in and the design is overall more consistent, but the Material design clearly wins. It can not only be expanded but it also is working well on both iOS and Android phones. Another strong argument is that React is working well with Material design and the team would not have to put too much effort into creating basic things. In the end, the team chose to use Material design.

6.1.5 Update: New smartphones for employees at the hospitals

While the team was in the fourth sprint, the managing directors decided that the hospital employees would get 6000 new Android smartphones and make these the standard work phone. While the first questionnaire clearly stated that the users are used to iOS rather than Android, this will change in the future. This new information confirms that choosing to design for Android was a good choice.

6.2 Preliminary Design Ideas

Before the team dived deeper into the design process they first created sketches that were built upon their own expectations of how the problem might be solved. While doing this the team did not get external influence from users. After some interviews the team got the feedback that resulted in design sketches that in the beginning looked innovative, but not optimized for everyday use. The operation list view is not considered in this chapter because it was pretty

straight forward, and stayed almost the same during the project. All sketches can be viewed in Appendix C, and the final clickable prototype can be viewed in Appendix D.

The design as shown in Figure 6.3 adopted the bar design but not in a horizontal way as in OpPlan. The design was also capable to show more than one room. It had a view into the near future, meant to be a personal view of operations in a short time period. The end user was familiar with horizontal bar design and later the team realized that adapting to a circle design could lead to a more complex learning process than using already familiar structures. As a result of the feedback this design was not implemented in the application. Users said it wasn't intuitive enough for them.

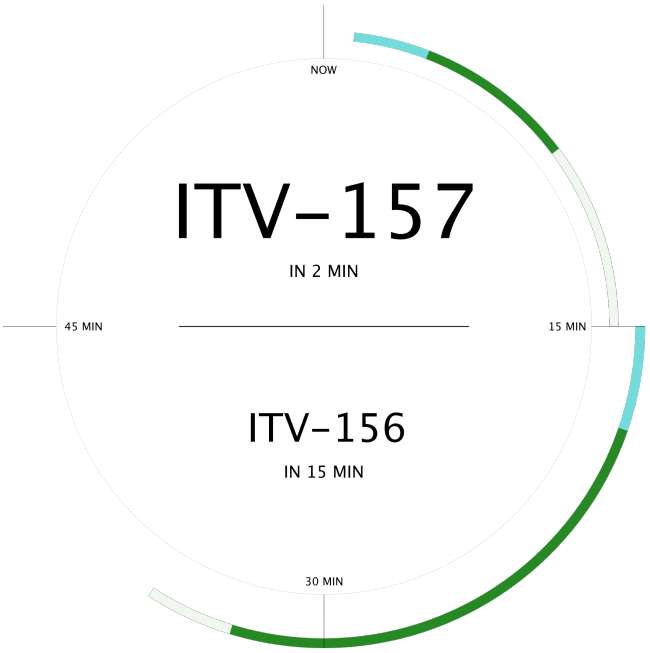


Figure 6.3: First sketch of a personal view

6.3 Operation Details View

The Operation details is a page with a lot of information. It is meant for all user groups that will use the application. It will be visited on a regular basis. Users get to this view if they touch an operation in the timeline or on one entry in the list view.

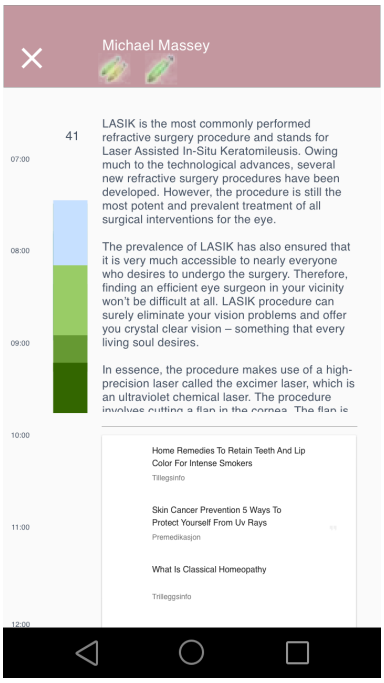


Figure 6.4: First sketch of the detailed view

The first design was based on the needs that the customer told the team. At this point the end users were not taken into account. What stands out is something that is not seen on the screenshot. It is the way of getting to this page. The team focused in the beginning on small details like animations. That caused some problems in the structure. Not only the motions that were a core of this design were against the material design guideline, also the design itself were not right considering the team had the goal to follow the guidelines. The motion would have worked like this: If a user touched the timeline for a specific operation, it would scroll to the far left of the screen. The information about this operation would appear with a scroll from bottom to up the screen. The motion would take at least 1 second, which could lead to a not fluid design. The first design can be seen in Figure 6.4.

Taking all these points into account the team redesigned this screen at a very early stage. The new design can be viewed in Figure 6.5.

Guideline	Feedback	Improvements
An interaction shouldn't keep the user waiting longer than necessary *Material motion - Motion* n.d.	The user would have to wait for every operation at least 1 second.	Open the Window without any animation.
Any customization to your apps motion experience should be consistent throughout the application *Material motion - Motion* n.d.	The motion would be only in this case and would have nothing to do with the rest of the app.	Open the Window without any animation.
Large UI areas and elements should be colored with your primary color *Color - Style* n.d.	The top application bar is red, the color is not used anywhere else in the app.	Change the color the the primary color.
Metrics and keylines *Metrics and keylines* n.d.	The application bar is too high, the margins are wrong.	Reduce size of application bar. Rearrange the elements
The Up button returns users to the previous screen they viewed. It navigates upward in the app's hierarchy until the app's home screen is reached *Navigation* n.d.	The behaviour is right but the but icon in the top left corner is wrong.	Replace the icon in the top left corner with a left arrow icon.

Guideline	Feedback	Improvements
"There is a lot to scroll" Questionnaire	The user had to open both tabs for certain information. Could cause a lot of scrolling/time.	Use a tab bar instead of expandable horizontal tabs.

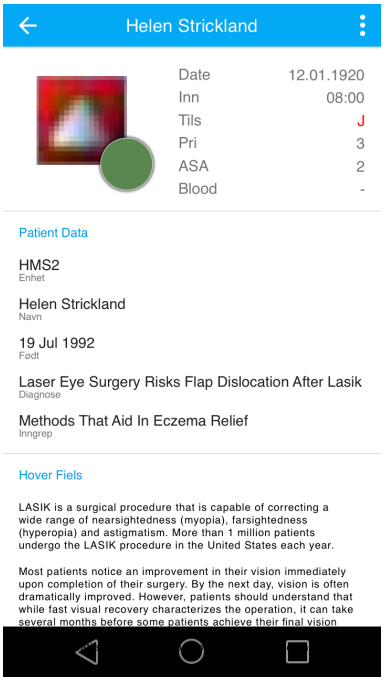


Figure 6.5: Further improvements.

The new design looks much cleaner and follows the design language. The primary colors are changed to the HEMIT logo color. The animation was removed. The top left icon was replaced and the information were ordered like the guidelines recommend it.

The team did the first user test with this design. Also there were more information in the guidelines that had to be addressed. With these two sources the team could create the final prototype.

The last prototype that was created in Sketch had all mentioned previous improvements. The timeline was kept but were rotated by 90 degrees to use the screen more efficient. The information was kept in horizontal tabs, to reduce scrolling. If the user clicked these they would expand, showing the corresponding information. The team then decided to start implementing it.

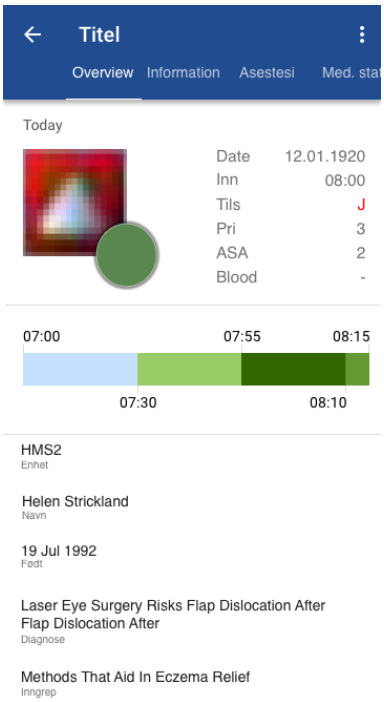


Figure 6.6: Final sketch of the details view.

The last changes were made based on the last usability testing. These changes were not made in Sketch, they were done directly in the code. The changes were limited to rearranging the information. The final sketch can be viewed in Figure 6.6

6.4 Timeline View

The timeline is one of the core elements in the application. It is the first screen that a user sees when they log in. It shows every operation in an operation plan with the phases in colors codings. The user can also see which day that is currently shown and what operations that are going to happen. The design was created with the base of the timeline that existed in OpPlan (Figure 3.3) and then optimized for smart phones. The major changes to the design were done to make better use of the limited screen space on a smart phone. In OpPlan the timeline is horizontal, but the team decided that inverting the axes would be better for a smart phone. Another difference is that in OpPlan, each operation is given its own row. This would lead to much wasted space on a phone screen, so the team decided to place all operations within one theater in the same column. In the beginning this seemed like a reasonable decision, but it

depended on the assumption that operations in one theater could not overlap. No usability testing was involved in the process of creating this screen design. The first design can be viewed to the left in Figure 6.7.

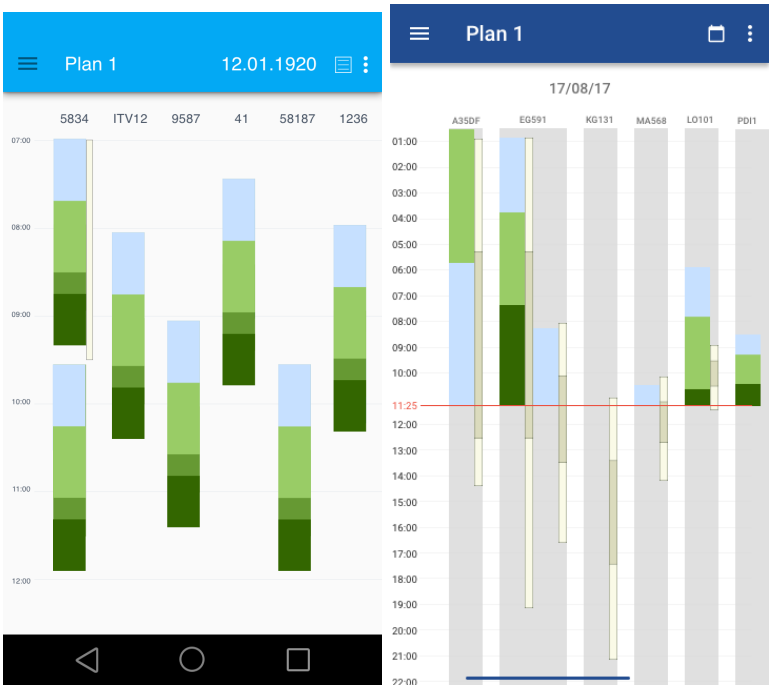


Figure 6.7: The first design and the final design of the timeline

After further research the following feedback was provided:

- The margins and the height/width are not right according to the guidelines from Google.
- The time is not detailed enough
- The use of the top right icons are not clear
- The screen real estate is not used optimally, there could be more information displayed
- There is no indicator for the current time
- Operations can overlap
- It is not easy to separate the different theaters

Taking these arguments the team created a new design based on the old one. The metric were changed, the buttons were arranged in a better way and an indicator for the current time were included. This design can be viewed to the right in Figure 6.7.

The indicator for the horizontal scrolling at the bottom was added later after the usability testing. This was done because some of the users did not realize that they could scroll to the right and see more rooms.

Chapter 7

System Description

7.1 Overview

The mobile application includes the most important features from the desktop application, along with some new functionality. To use the application the hospital employees should log in with their already existing username and password. In the desktop application the user would have to select a single operation program with their role, but the mobile application provides the opportunity to easily switch between operation programs and the user's role in that program. That is done by selecting desired program and role in the left menu, as shown in Figure 7.1. As starting the desktop application is slow the team believes this switching possibility can be a great advantage. The application will automatically start up on the current date, and the user can easily switch to any day using the date picker, also shown in Figure 7.1.

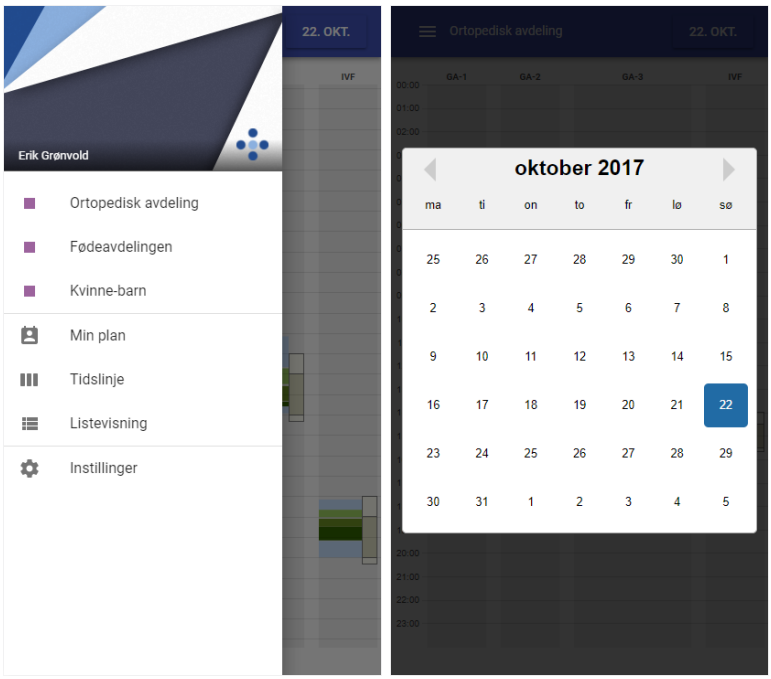


Figure 7.1: The menu drawer and the date picker.

From the background study, the team understood that the timeline was an important feature. Considering this, a lot of work was put into making it usable and easily understandable. The finished implementation of the timeline is shown in Figure 7.2. It shows one column for each operation theater in the selected operation plan. As in the desktop application the timeline includes the different phases for both the planned time and the real time for each operation. Unlike the desktop application the time is shown along the y-axis. Every operation in the timeline is clickable, providing the possibility to change the phase of an ongoing operation, see some details of each operation and easily navigate into the operation details view.

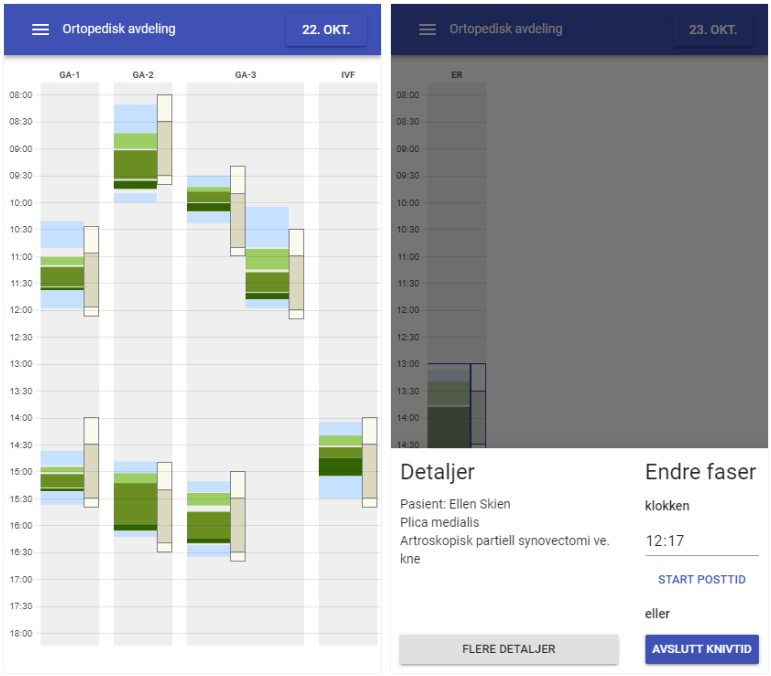


Figure 7.2: The timeline view.

To see all operations for different theaters the user can either view it in the timeline, or in a list view, shown on the left side of Figure 7.3. The list view contains a tab for each theater and lists the operations. From the list view the user can see the diagnosis, the planned procedure and the gender and age of the patient. By clicking such a list item the user is directed into the operation details view of the selected operation, shown on the right side of Figure 7.3.

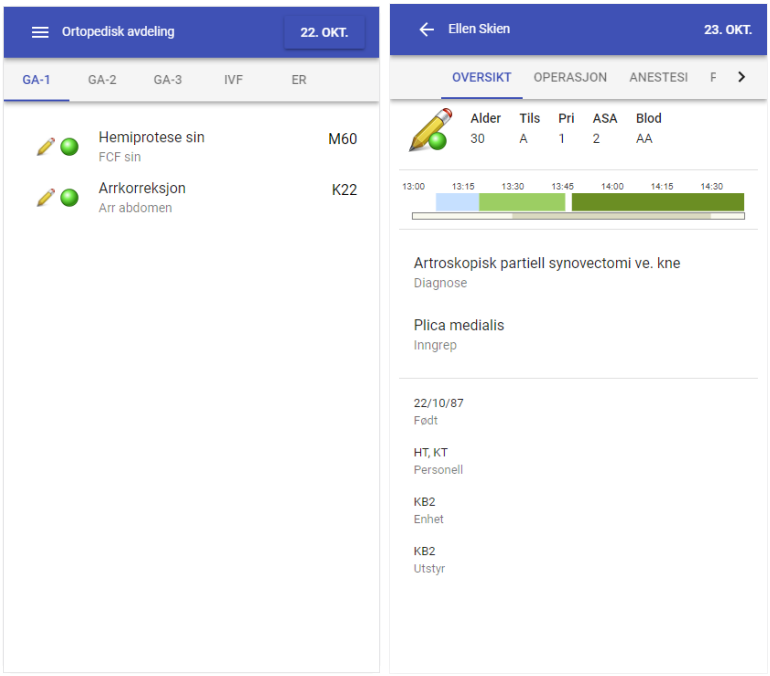


Figure 7.3: The list view and operation details view.

By selecting a operation in the timeline or the list view the user can view a lot of information about it in the operation details view. The operation details view consists of several tabs of information. The overview tab, in Figure 7.3, includes the most important details about the patient and the operation. This includes a small horizontal timeline with all the phases. The other tabs are shown in Figure 7.4. The operation tab and the anesthesia tab contains more details, and is mostly structured as the corresponding tabs in the desktop application. The last tab of the operation details view is the crew tab. It contains all information about the crew that is supposed to conduct the operation. This provides the opportunity to call any crew directly from the application. From the results of the background study this is something that can be very useful, as calling is used a lot and it is not possible to do from the desktop application.

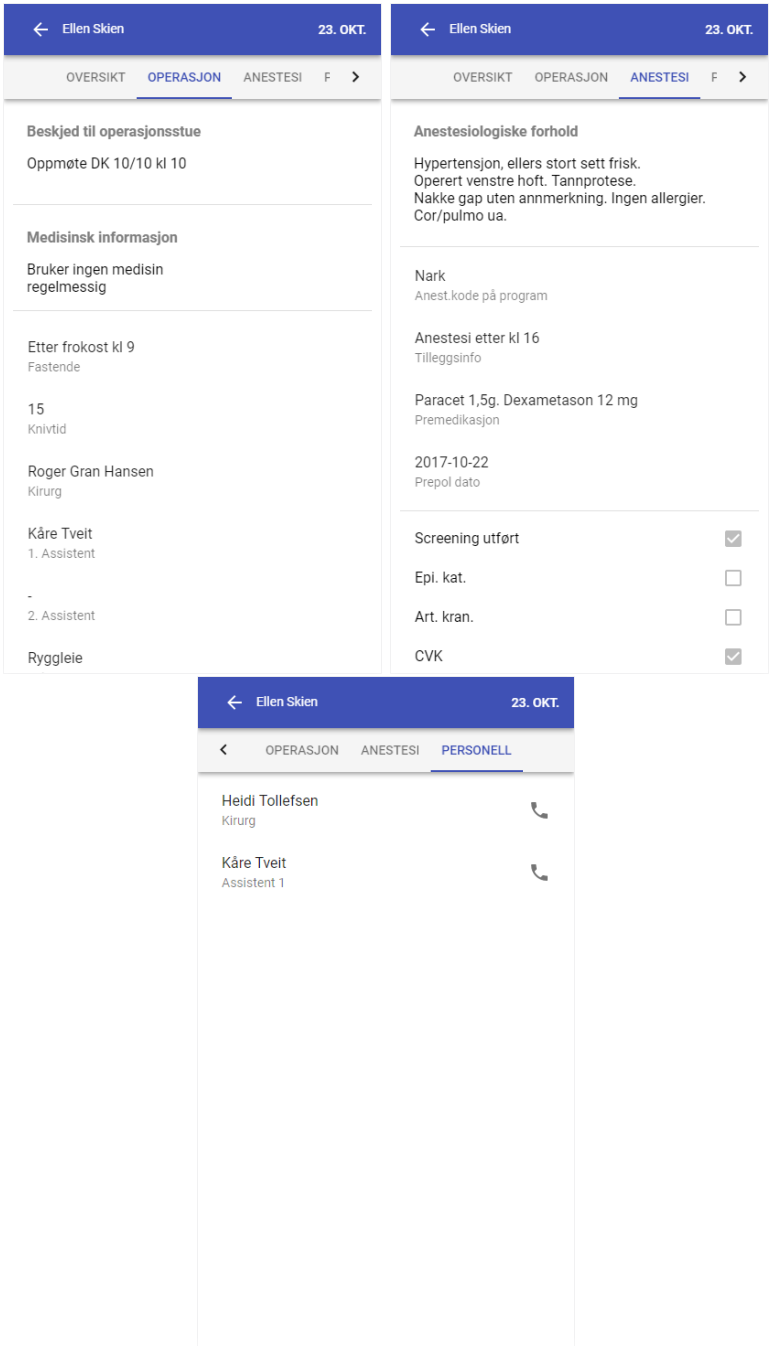


Figure 7.4: The remaining tabs of the details view.

The last view, Figure 7.5, is the personal plan for the day. It shows a timeline of all the operations that the currently logged on user is scheduled to be a part of. The timeline is shown vertically with some information beside each different operation. When the user clicks an operation, they are presented with the same drawer as in the timeline view (Figure 7.2). This consistency is described by Don Norman as one of the 6 most important principles in design (Norman 1988).

A personal plan is not something that can be found in the current desktop application, but is made based on the results from the background studies. Showing the plan as a timeline makes it easily understandable, and the user can get a quick overview of when they have things to do, and when they can have lunch. As the application is supposed to be used on the go, the team believes that the user will find it useful to always have access to their plan. The plan will include a real time update, so that the user can see the actual progress of operations against the planned progress.

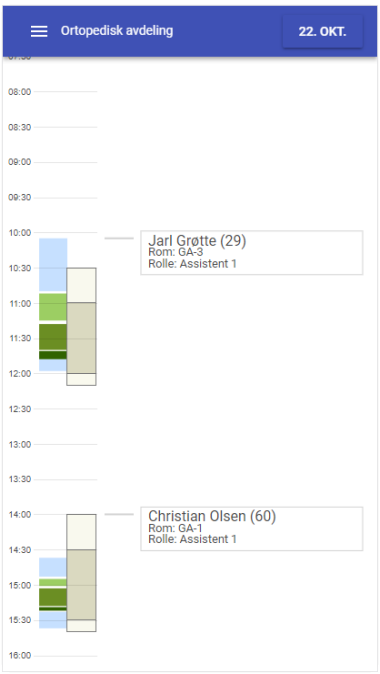


Figure 7.5: The my plan view.

7.2 System Architecture

7.2.1 Data Management

At a fairly early stage in development after choosing React, it was decided upon utilizing Redux for the application. What this means is that the state of any component is derived from a single source of information - the store. To initiate a state change, a component dispatches an action. Through a function called a reducer this action translates into a new state that is handled in the store, as is illustrated below. Each previous state is maintained, so no changes are made directly to the state of your application. Utilization of a centralized store rather than manual propagation of data between components makes for a noticeably easier management of data flow in the application. The store of Scalpel is shown in Table 7.1, while the available actions are shown in Table 7.2.

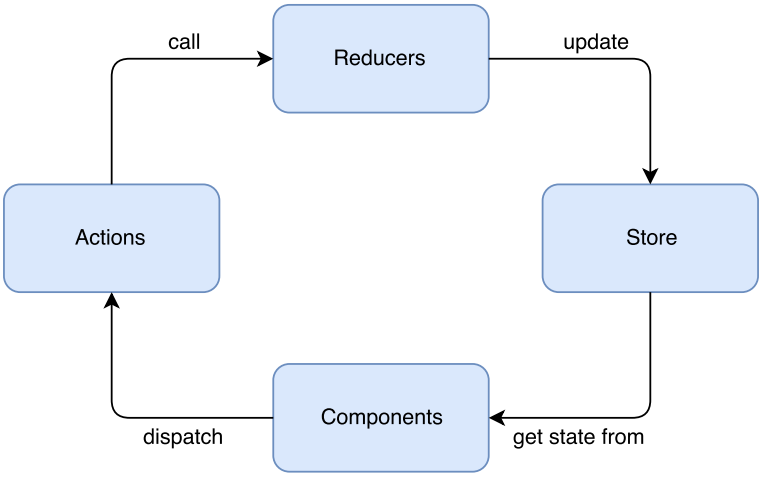


Figure 7.6: The flow of data through Scalpel.

Name	Data type	Description
operations	array	All operations
theaters	array	All theaters
operationPlans	array	All available operation plans
operationPhases	object	Metadata for each operation phase
selectedOperationPlan	number	Id of the selected operation plan
selectedDate	moment	The selected date
persons	array	all users
loggedInUser	number	id of the currently logged in user
menuDrawerOpen	boolean	is the menu open or not

Table 7.1: The data store of Scalpel.

Name	Description
setSelectedDate(date)	Updates the currently selected date
openMenuDrawer()	Opens the menu drawer
closeMenuDrawer()	Closes the menu drawer
finishOperationPhase(id, time)	Ends the currently active operation phase
startNextOperationPhase(id, time)	Ends the currently active operation phase and starts the next
setSelectedPlan(plan)	Updates the currently selected operation plan

Table 7.2: The actions available to components.

Then it would be up to each component to filter out and manipulate the data to suit its way of presentation.

Using Redux allowed the team to separate the data fetching mechanisms from the rest of the application. In a complete application the Redux store would be populated with data from an API, but the application would only interact with the store, with no knowledge of where the data originated from. That enabled the team to go ahead with implementation without having an API ready. Instead, the Redux store was populated with static data that was formatted to look as close as possible to the data that would come from OpPlan.

7.2.2 Component Hierarchy

Figure 7.7 outlines the component hierarchy of Scalpel. The Router component comes from the React-Router library and matches the URL and chooses which components to show. Some of the components are red because they are wrapped in a Redux container component. The dashed lines mean that the parent components each create their own copy of the component. Initially the header was placed above the route components, as a direct child to the app component, but the team found that as each of the views needed to show different information in the header, it was easier to render it separately for each route component. In this way, the route components could themselves choose the properties of the header. Since both the Timeline, Operation List and My Plan routes needed much of the same information, the header was wrapped in a new container component, Menu Header.

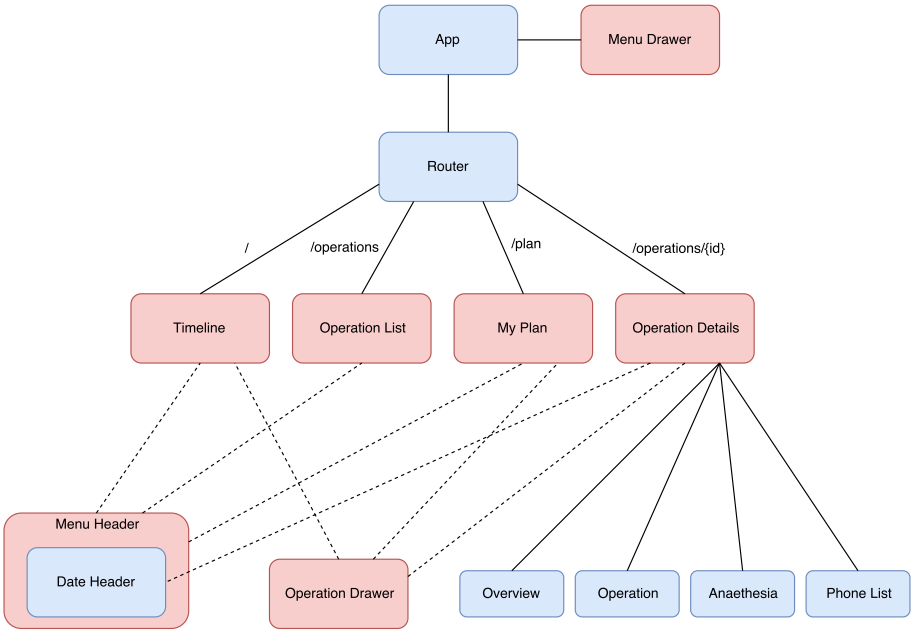


Figure 7.7: The component hierarchy of Scalpel

7.3 Implementation

7.3.1 Boilerplate

To start on a modern JavaScript application, there are many choices that have to be made. Choosing a UI library, bundler, test framework and linter. After choosing to use React, the team decided to use a React starter kit, to

reduce the number of decisions that had to be made. The team chose *create-react-app* [1] as the starter kit. It is developed by Facebook itself and is widely used, as can be seen by the number of stars on GitHub. This set up the Webpack and Babel configuration, added Jest, a testing framework, and created npm scripts for running, building and testing the application. It enabled the team to start developing the features of the application immediately, without having to think much about how to set up the boilerplate. Since the team wanted to focus on the design, this simplicity was very welcome.

7.3.2 Reusing software

Through npm, the team got access to a large repository of reusable software components. It is in fact the largest ecosystem of software modules out there [2]. This enabled the team to reuse a lot of good software, so the focus could be on developing what was specific for our application. The book Software Engineering (Sommerville 2015) describes benefits of software reuse, like increased dependability due to using tested and open source software, reduced risk and accelerated development.

React and Redux were the main npm modules the team used, but several other libraries were also used. Since JavaScript has a rather small standard library, the team used an npm module for handling time [3], and one that gave access to useful utility functions for manipulating collections [4]. Another library that was used, Material UI, is described in the next section.

7.3.3 Design

To implement the design components, the team used Material UI, as described in the Technologies chapter. This enabled the team to quickly implement Material Design components to test out our designs. The library has many contributors and is under active development, so the team considered it safe to use. It also made it possible to describe the color theme of the application easily. Once a primary and secondary color was chosen, it would generate a theme according to Material Design coloring.

7.3.4 Timeline

The timeline view was arguably the most complex view in the application, both to design and implement. Extensive research was done on existing solutions, but none was found that fit the exact use case. So the team ended up implementing the timeline using D3.js. That gave a lot of freedom, but it is also

[1] https://github.com/facebookincubator/create-react-app
[2] http://www.modulecounts.com/
[3] https://momentjs.com/
[4] https://lodash.com/

low level so a lot of code was necessary to get the right behaviour. The power of D3 comes from its ability to connect data points to elements in the DOM, so refreshing and reacting to changes to data is very efficient, which fit the use case well.

Before the data could be handled to the view, it had to be filtered and transformed correctly. It would filter to get only the operations on the selected operation plan and operation day, and only show the theaters that has operations in them. A problem that the team discovered was also that operations in the same theater could overlap in the timeline. Because of this, the timeline could not assume that each theater would only cover one column. To distribute the operation across as few columns as possible, the Greedy-Activity-Selector algorithm was used to repeatedly select non-overlapping operations until all operations were selected. The correctness of this algorithm is described at page 421 in Introduction to Algorithms (Cormen 2009). From this transformed data, the operations would be placed in a grid which could be both zoomed and panned.

D3 is however not optimized to be used on mobile, and the zooming and panning operations are quite computationally heavy, as it has to calculate new positions of every element in the timeline. This is apparent on some smartphones and browsers where panning and zooming becomes jittery. However, it was considered to be the best solution available, and the team focused on optimizing the solution.

Once the code for displaying the timeline was figured out, it could be reused in both the personal plan view and the operation details view, with a few modifications.

7.4 Servers and API

Towards the last quarter of the project the team got introduced to some work that has been done on an API that should eventually allow for phones to request data from OpPlan. Working together with HEMIT the team started the process of setting up servers to run Scalpel on the internal network and provide data directly from the test environment of OpPlan. Despite work on this not being entirely finished due to the late introduction of the API and the time constraint, what follows is a technical description of the architecture and security features of the designed implementation.

Restrictions of access on the medical networks as well as HEMIT made it impossible to just set up a simple server that communicates with OpPlan and Scalpel directly. Instead, for testing purposes, two servers were set up at HEMIT. One of the servers is able to communicate with the Windows Communication Foundation, OpPlan WAS, and to handle the acquisition of an authorization token through IAM. However, the team did not have access to the

network this communication occurred on, which resulted in there being set up a 2nd server with which a phone could communicate with over the BYOD (Bring Your Own Device) network at HEMIT. This network could be accessed from devices the team owned, using a set of credentials given, without the requirement of having a PULS authentication card. The 2nd server is as of yet not able to exchange data with OpPlan WAS, requiring opening of ports to be reachable through the firewall. Using a developer computer at HEMIT these servers were setup to run Scalpel using Microsoft IIS (Internet Information Services) web servers. The operation data from OpPlan is sent as JSON files over an HTTPS connection.

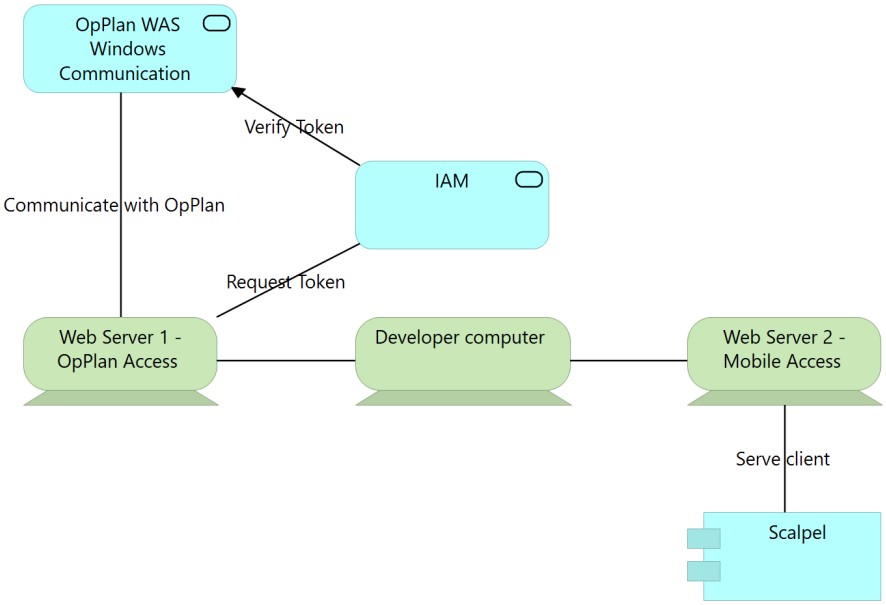


Figure 7.8: The server architecture

7.5 Security

The handling of security at this stage of the implementation is far from optimal, with the application and API in it's current state, simply working as test environments. This section will discuss several of the security aspects that would have to be handled before an actual release build should be made.

In the API there is no proper handling of login credentials, they are simply part of the query string of the URL when issuing a HTTPS request. In addition the API is currently bypassing the session management that OpPlan offers, where a user would for example lose access when connection to IAM is lost and the authentication token can not be renewed. These issues lead to the possible

security issues presented by broken authentication, session management and security misconfiguration. The HTTPS encryption, or specifically TLS 1.2 or 1.3 that is used here, is considered safe and has not as of yet been broken. This level of encryption should therefore be sufficient to prevent sensitive data exposure.

The test API is not yet enforced to exclusively use HTTPS, with requests also being possible over HTTP. For a real implementation this is not sufficient and leads to the possibility of cross-site request forgery (CRF). There are surely more issues building under the underprotected status of the API, but proper authentication handling and encryption would have to be managed before the customer can use the application in hospitals.

Some security concerns are to a large extend handled by the application layer of OpPlan, such as SQL injection and broken access control. The application layer handles what data a user should see and how to request and provide it.

Additional security concerns exist on the application side however. Being a web application that builds on several pre-made components, there exists the possibility of including weaknesses that may exist in these components. These can be very difficult to discover and manage internally, but more likely to be exploited if the components are widely used. On the other hand, the application will not run on the open web, but on the closed networks of the hospitals. This helps mitigate security issues such as Cross-site scripting (XSS), since all resources are from the closed network, and insufficient attack protection, which is otherwise of concern with the API. There should be systems present to detect and block off attackers.

The users will not be able to use their personal phones to run the application, but there is the possibility of a user for example taking a screenshot of sensitive data. This issue naturally does also exist for the current system of OpPlan, but should nonetheless be considered.

Chapter 8

Work Process

In this chapter the organization and execution of the work process will be presented. Each section will describe the sprint and the results it produced. A table of the sprint backlog is shown in every sprint. It shows the category of the issue, a description, the estimate that was made during the sprint planning, and the status of the issue after the sprint was complete. First the meetings performed each sprint is described, the sprint planning meeting and the sprint retrospective meeting.

8.1 Sprint Planning

A sprint planning is a meeting that takes place in the beginning of a sprint. The people attending this meeting should be the whole development team and the product owner. During the meeting the highest priority user stories are taken from the backlog and broken down in smaller tasks and added to the sprint backlog.

At the end of the planning meeting there should be two defined artifacts:

- Sprint goal
- Sprint backlog

A sprint goal is a short, one- or two-sentence, description of what the team plans to achieve during the sprint. It is written collaboratively by the team and the product owner.

A sprint backlog is a list of the product backlog items the team commits to delivering plus the list of tasks necessary to deliver those product backlog items.

All the tasks are also estimated.

8.2 Sprint Retrospective

A sprint retrospective is an event that occurs at the end of a sprint and before a sprint planning. During this meeting the scrum master present the development team with three questions. The development team then discuss these questions and make a list with answer to all questions.

- What shall we continue doing?
- What shall we start doing?
- What shall we stop doing?

To finish the meeting the scrum master take a final round and asks each member what they think is the most valuable item and the item the team shall focus on in the next sprint in the newly made list and picks one of them. When everyone around the table have decided on an item the two most voted items are declared as main goals to focus on during the next sprint. Then the team will discuss how these two goals can be achieved by suggesting concrete actions and giving responsibilities.

8.3 First Scrum Sprint

During the outline planning phase the team had discussions about what technology that should be used, roles were assigned with responsibilities to team members and some time were spent on learning React, as this was very unfamiliar for most of the team members. The team also started doing some design sketches and got feedback on from the customers on them. The team got contact information to possible interview subjects, and were able to interview one anesthesia doctor to get an understanding of what he needed. Some work on the project plan was also done.

8.3.1 Goal

Decide on technology stack

8.3.2 Sprint Backlog

Tasks	Status
Make meeting templates	Done
Create team contract	Done

Decide on technologies	Done
Finish timesheet	Done
Decide model types	Done
Set up wiki	Done
Set up git repository	Done
Create design sketches	Doing
Work on project plan	Doing
Do React tutorial	Backlog

Table 8.1: Sprint 1 backlog

The sprint 1 backlog can be seen in Table 8.1.

8.3.3 Sprint Retrospective

- Continue doing
  - Having good communication
  - Sharing knowledge
  - Work in pairs
- Start doing
  - Meet timely. Some members of the team did not show up at the right time for the customer meeting, and the rest of the team did not know when they were coming. This delayed the meeting.
  - Divide responsibility better. Make sure that someone is responsible for each task.
  - Talk with your responsibility backup about how things are going, e.g. on Fridays.
  - Have a clear meeting agenda with a meeting leader.
  - Choose someone to write meeting minutes before the meeting starts.
  - Have clear goals and deadlines. They should be small and achievable.
- Stop doing
  - Be late to meetings
  - When the team has meetings, people should stop talking at the same time. There are no clear line between meeting and discussion.

The team decided to focus on setting clear goals with deadlines and to structure the meeting in a better way. The first issue would be addressed by delegating tasks on Trello with concrete deadlines. The latter would be achieved

by having the team leader delegating responsibilities during customer meetings. The team has to report to the project leader if they have something they want to discuss with the customer. The meeting leader makes sure the team follows the agenda and can stop people if the discussion goes too far. Different topics should be presented by the most appropriate team member. Further, the team leader would select a team member to write minutes during each meeting. This will be stated in the meeting agenda.

8.4 Second Scrum Sprint

During the second sprint, the team focused on understanding the different types of potential users of the application, this resulted in some personas and user stories. The team made a clickable prototype and started implementing the application.

8.4.1 Goal

Have a clickable prototype

8.4.2 Sprint Backlog

The sprint backlog can be seen in Table 8.2

Type	**Task**	**Status**
Research	Field study with anesthesia doctor	Done
Documentation	Create a general enterprise model(ArchiMate)	Done
Research	Go through React tutorial	Done
Design	Patient List	Done
Design	Menu	Done
Design	Timeline	Done
Research	Create questionnaire	Done
Research	Field study with anesthesia doctor	Done
Research	Create digital notes from interview with Dani	Done
Documentation	Finish meeting minutes template	Done
Implementation	Set up skeleton code	Done
Implementation	List view	Done
Report	Create report outline	Done
Report	Choice of documentation	Done

Table 8.2: Sprint 2 backlog

8.4.3 Application screenshots

This sprint was the first sprint with implementation and only a very basic version of the application was ready at the end of the sprint, shown in Figure 8.1. There were two views, the list view and the timeline. The timeline could not be zoomed, and when panning the axis labels would disappear. Also one could not see the phases of an operation. The list view only showed the name of the patient and could not be clicked.

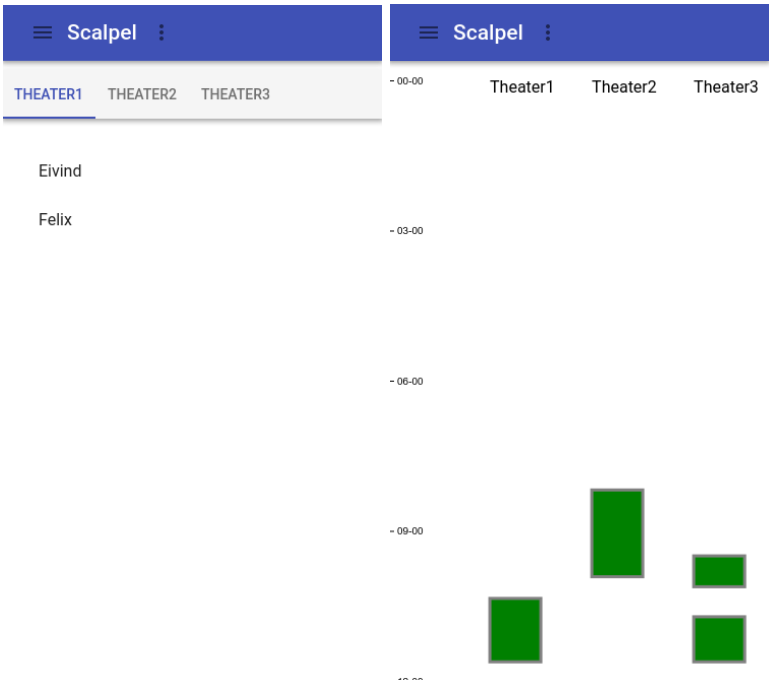


Figure 8.1: State of the implementation at the end of Sprint 2

8.4.4 Sprint Retrospective

The team reviewed the goals that were sat in the last retrospective meeting. The structure of the meetings had become much better as the team were more focused on following the agenda and leave any discussions until the end of the meeting. The goal of setting clear goals with deadlines turned out to not work very well, as the end of a sprint was working as a deadline for finishing all tasks in the backlog. It turned out that the team felt like this dues were enough.

- Continue doing
  - Be on time

- Start doing
  - Organizing work
  - Keep Trello updated
  - Pair programming
  - Task planning
  - Answering on slack
- Stop doing
  - No particular suggestions were made for this.

The team decided that organizing work was most important to focus on. The following actions were considered useful in order to achieve this goal.

- Sprint planning: Use more time on the next sprint planning. We're going to use planning poker to estimate the scope of each task.
- Programming: Dividing into groups of 2, 2 and 3 that can work together during the next sprint. This will make it easier to find time slots to meet. The human resources responsible will handle the division into pairs.
- Design: The team are going to be better at commenting the design decisions, so that there is something to discuss when the team meet.
- Report: The team are going delegate report tasks better. Everyone will work individually, but the team will discuss if there are issues.

8.5 Third Scrum Sprint

The team prioritized personas in a discussion with the customers and decided that the focus will be on the anesthesia doctor, as he is the person who is always on the run and would benefit most from having the application available at all times. The team also focused on the implementation of the application.

8.5.1 Goal

Prioritize personas

8.5.2 Sprint Backlog

The sprint backlog can be seen in Table 8.3

Type	Task	Est	Status

	Send prototype to Tor-Arne	0	Done
Documentation	Create personas	1	Done
Design	Settings	2	Done
Design	Operation status pop-up	2	Done
Design	Onboarding	5	Done
Design	Operation data	3	Done
Report	Risk management	5	Done
Implementation	As a user I would like to see when each phase of an operation starts and ends	5	Done
Implementation	User would like to long press on an operation to change the status	2	Done
Implementation	User would like to press on an operation to get to the details view	1	Done
Implementation	Add Redux	5	Done
Implementation	As a user I would like to easily switch between the day viewed for a specific operation program	3	Done
Implementation	As a user I would like to have specific options in the header per view	3	Testing
Implementation	As a user I want to scroll horizontally through theaters in the bar view.	5	Testing
Implementation	As a user I would like to view operations that overlap in time side by side to compare them	8	Testing
Implementation	As a user I would like to see the detailed view of any selected operation	13	Doing
Implementation	As a user I would like to input when a phase I am involved in starts or ends	8	Doing
Report	Finish preliminary study	3	Doing
Implementation	As a user I would like to change the details of an operation	13	Backlog
Report	Make a test plan	8	Backlog
Report	Requirement specification	8	Backlog
Implementation	As a user I would like to view my plan for the day	8	Backlog
	Figure out communication with the backend	21	Backlog
	Analyze results from questionnaire	3	Backlog
Documentation	Design architectural models for scalpel	5	Backlog

Table 8.3: Sprint 3 Backlog

8.5.3 Application screenshots

After the third sprint, many improvements had been made to the application, as is shown in Figure 8.2. The timeline now included both the separate phases of an operation, and its planned schedule. The timeline also showed the current time, and could be zoomed. This sprint also yielded the first implementation of

the details view.

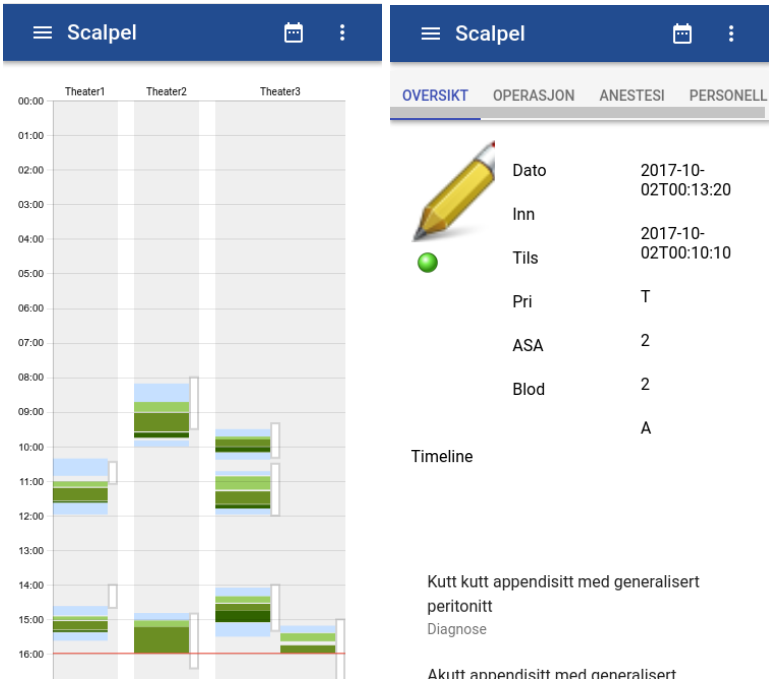


Figure 8.2: State of the implementation at the end of Sprint 3

8.5.4 Sprint Retrospective

The meeting began with reviewing the goals sat in the last retrospective meeting. The planning poker for estimation of tasks had worked well, as the team could more easily plan their work. Commenting on design decisions had also improved as the team members were more focused on giving feedback on others work. Report tasks were better handled in this sprint, as the Report manager delegated tasks. Division into pairs had been done, but had not worked optimally as there had been some misunderstandings on whether the intention was to do pair programming or to cooperate. The team decided to continue dividing into pairs, but the pair would themselves decide which tasks that was suitable for cooperation.

- Continue doing
  - Updating Trello
  - Working in pairs
- Start doing

  - Schedule breaks
  - Tracking time spent on tasks
  - Stand-up meetings
  - Pull requests
  - Define a clear goal on each sprint
- Stop doing
  - Expanding scope
  - Using computers and phones during meetings when not relevant

The team decided to focus on continuing to work in pairs, as mentioned, defining a clear sprint goal and tracking time spent on tasks. The sprint goal would be defined in each sprint planning meeting. The team had some vague goals before this sprint, but it was decided to make them clearer to make it easier to focus and work hard to reach them. The time tracking would be useful to get better at estimating story points for tasks, as the team would be able to see the estimated time spent against the real time spent on each task. To address this, it was decided that the person completing a task on Trello would leave the hours spent as a comment.

8.6 Fourth Scrum Sprint

A usability test plan was made and responders from the questionnaire who wanted to participate in usability tests were contacted. Usability testing was completed, and the results were analyzed in order to improve the application. The team also worked on the design, report and implementation.

8.6.1 Goal

Have application ready for usability testing and complete usability testing

8.6.2 Sprint Backlog

The sprint backlog can be seen in Table 8.4

Type	Task	Est.	Status
Design	Indicate what date is currently shown in timeline and list view	2	Done
Implementation	As a user I would like to see the detailed view of any selected operation	13	Done
Implementation	Update menu with correct names etc	1	Done
Implementation	As a user I would like to click any part of an operation	2	Done

Research/Testing	Plan usability test	3	Done
Implementation	As a user I would like to see when each phase of an operation was planned to be in the timeline view	2	Done
Implementation	As a user I would like to input when a phase I am involved in starts or ends	8	Done
Implementation	As a user I would like to navigate between operation plans	2	Done
Implementation	Create realistic test data	1	Done
Implementation	Format list view	1	Done
Implementation	Handle empty data fields	1	Done
Implementation	As a user I would like to view operations that overlap in time side by side to compare them	8	Done
Implementation	As a user I want to scroll horizontally through theaters in the bar view.	5	Done
Report	Android/IOS differences and design	3	Done
Report	Explain the project methods (XP, Scrum, Kanban)	3	Done
Urgent	Create digital notes from field study (with Tor Arne)	1	Done
Report	Requirement specification	8	Done
Report	Document the test plan	8	Done
Report	Write sprint chapters	5	Done
Report	Summarize results from usability testing	3	Done
	Server setup at HEMIT		Done
Research/Testing	Research how logging in works	3	Done
Research/Testing	Analyze results from questionnaire	3	Doing
Implementation	As a user I would like to view the timeline in the operation details	3	Doing
Report	Finish the preliminary study	3	Doing
Implementation	As a user I would like to see the age and the gender of the patient in the list view	1	Testing
Implementation	As a user I would like to see the most important fields on top of the operation details view	1	Testing
Implementation	Fix d3-zoom when navigating	2	Testing
Implementation	As a user I would like to see what day I am viewing in the timeline	2	Testing
Implementation	As a user I would like to have specific options in the header per view	3	Ready
Presentation	Check possibilities for borrowing equipment for video	2	Ready
Implementation	As a user I would like to see whether or not there are more theaters than I can currently see on the timeline	3	Backlog
Implementation	As a user I would like to view my plan for the day	8	Backlog
Design	Indicate whether or not horizontal scrolling can be done on the timeline	1	Backlog
Documentation	Design architectural models for scalpel	5	Backlog

Video	Make storyboard	5	Backlog
Implementation	As a developer I would like to generate operations for the current day and time for easier development	2	Cancelled

Table 8.4: Sprint 4 backlog

8.6.3 Application screenshots

After the fourth sprint, support for selecting different operation plans was added. The currently selected operation plan was shown in the header (Figure 8.3). The other major change was in the list view, which now showed the status of the operation and the procedure and diagnosis of the operation.

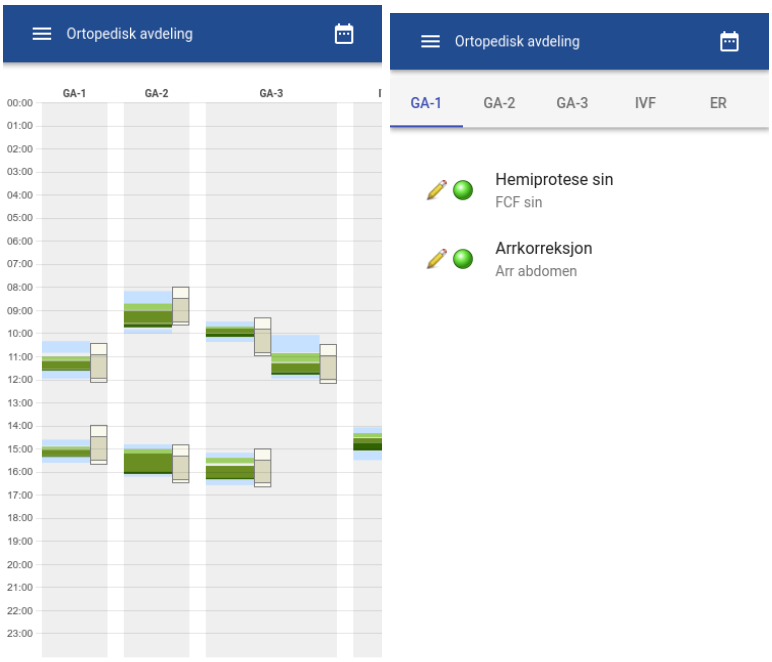


Figure 8.3: State of the implementation at the end of Sprint 4

8.6.4 Sprint Retrospective

From the last retrospective the team evaluated the pair work as a success. The team members liked the flexibility of having someone to cooperate with but also being able to do task individually if they wanted to. A sprint goal was decided in the sprint planning, and worked well. The tracking of time spent on tasks was forgotten by many of the team members, and was evaluated as less

important to focus on in the next sprint, as there were not many sprints left in the project.

- Continue doing
  - Taking planned breaks
  - Pair work
- Start doing
  - Inform missing team members
  - Chop down tasks into smaller units
  - Make sure that all tasks are on Trello
  - Put Trello up when working
- Stop doing
  - Communication problems when team members were not present
  - The usability test were not well enough prepared
  - The responsibilities were not divided well in this matter

All team members thought that better task delegation and and stop doing last minute preparations were the most important tasks to focus on. To improve task delegation it was decided that larger tasks should be divided into smaller units, optimally not bigger than 1-2 hours of work. Further, all tasks should be on Trello and kept updated with correct worker and where it is in the in the process. The goals after this sprint review were related, and the team concluded that by better task delegation last minute work would be avoided.

8.7 Fifth Scrum Sprint

The fifth scrum is the penultimate sprint and therefore the team would focus a lot on the report. At the start of this sprint the team had 65 pages, the team set a goal of 110 pages for this sprint. The deadline is approaching and the team wanted to fine-tune the report after the fifth sprint.

Implementation wise the team did one last major step in order to get a good looking and working prototype for the customer. According to research the user need a view for personal plan. This feature is the last that was implemented in Scalpel.

8.7.1 Goal

Finish and stabilize the features in the application, and have 110 pages in the report

8.7.2 Sprint Backlog

The sprint backlog can be seen in Table 8.5

Type	Task	Est.	Status
Report	Screenshot descriptions	3	Done
Report	High-level system description	2	Done
Report	Evaluation on further work	2	Done
Report	Evaluation on improvements	2	Done
Report	Document final product using text and images	2	Done
Report	Finish the preliminary study	3	Done
Report	Setup ShareLaTeX document	2	Done
Report	Evaluate advisors	2	Done
Report	Describe the technical architecture of the system	5	Done
Report	Write about use cases	3	Done
Report	Write a introduction chapter	2	Done
Report	Evaluate customers	2	Done
Report	Evaluate task	2	Done
Implementation	Fix d3-zoom when navigating	1	Done
Implementation	Specific options in the header per view	3	Done
Implementation	Divide operation details with horizontal lines	1	Done
Implementation	Show which operation that is selected in timeline	1	Done
Implementation	Swipe anywhere in swipeable views	1	Done
Implementation	Refactor overview tab in operation details	1	Done
Implementation	Unit tests for timeline data	1	Done
Implementation	Show age and gender in list view	1	Done
Implementation	Show current date anywhere in application	1	Done
Implementation	Indicate that scrolling is possible in timeline	3	Done
Implementation	Show a timeline in the operation details	3	Done
Implementation	Show role of crew members in ‘personell’ tab	1	Done
Implementation	View of personal plan for the day	8	Done
Video	Check possibilities for borrowing equipment	2	Done
Video	Make story board	5	Done
Implementation	Show symbol of ‘smittefare’	1	Doing
Documentation	Design architectural models	5	Doing
Video	Make footage	10	Doing
Presentation	Create presentation	7	Doing
Implementation	Add help-popup for each page	3	Backlog
Implementation	Indicate which operation program that is currently selected	1	Backlog
Implementation	Improve date picker design	1	Backlog

Table 8.5: Sprint 5 backlog

8.7.3 Application screenshots

After sprint 5 was complete, the application had one new view, the plan view, and an improved details view. Screenshots of these are in Figure 8.4.

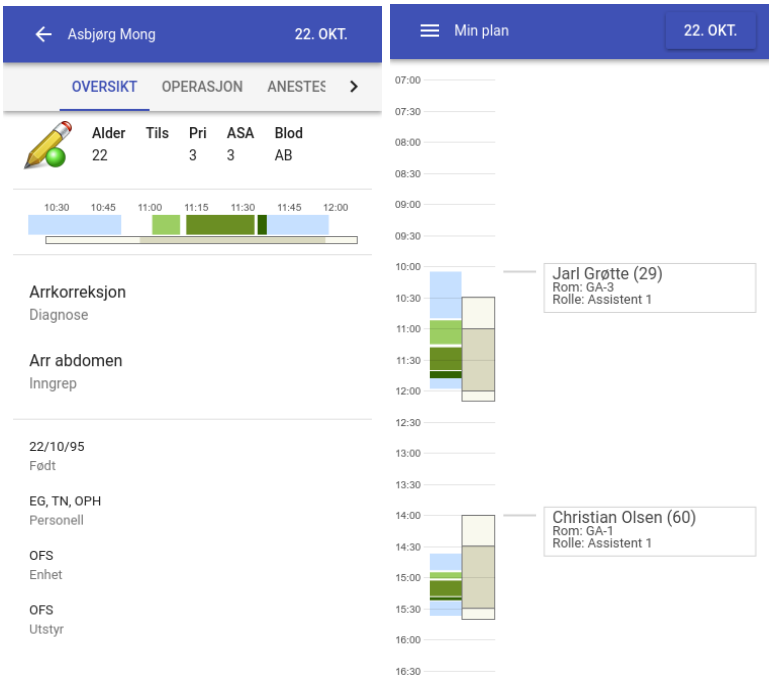


Figure 8.4: State of the implementation at the end of Sprint 5

8.7.4 Sprint Retrospective

The improvement of task delegation was a success in order to avoid last minute work, the team did not experience the same problems this sprint.

- Continue doing
  - Good communication
  - Keeping Trello updated
  - Having fun
- Start doing
  - Notify about whether or not you are attending work sessions
  - Being more decisive

  - Having backup solution, e.g. if a person is sick on the presentation day, there should be a backup

- Stop doing
  - No particular suggestions were made for this

8.8 Sixth Scrum Sprint

8.8.1 Goal

Delivery of the report and video on the 16th of November. Deliver the final presentation on the 23rd of November.

8.8.2 Summary of the Sprint

This sprint was seen as the formal closure of the project. Work was almost extensively done on finishing the deliverables and handing these over to the advisor and the customer. Whatever time remained for specific members of the team after the first deadline could then be spent on bug fixes and optimization of the application.

The tasks that were finished during this phase:

- Finishing the report.
- Finishing the presentation.
- Presentation at HEMIT.
- Presentation at NTNU.
- Finishing the video.
- Finishing the product.

8.8.3 Video

The course specified that the team should deliver a 1 minute video showcasing the project. The initial idea the team had for a video involved filming a story in the hospital, showing a use case of the application. The team storyboarded this idea, shown in Figure 8.5. When the day of filming dawned, however, the contact person that would let us in to the hospital could not come, and the filming had to be aborted. At this point there was only a week until delivery, so filming at a later point was considered to be unfeasible. The team had no other choice but to come up with a new idea. Something else that we

had earlier considered was creating an animation based video using PowToon [1] and voice-over. Given the situation, this solution was considered to be the best one, and a student license of PowToon was purchased. Making a video in PowToon turned out to be a very smooth process, and the result turned out better than expected. The finished video can be watched on YouTube [2].

[1] Website: `https://www.powtoon.com/`

[2] `https://youtu.be/00zCJbjRE9I`

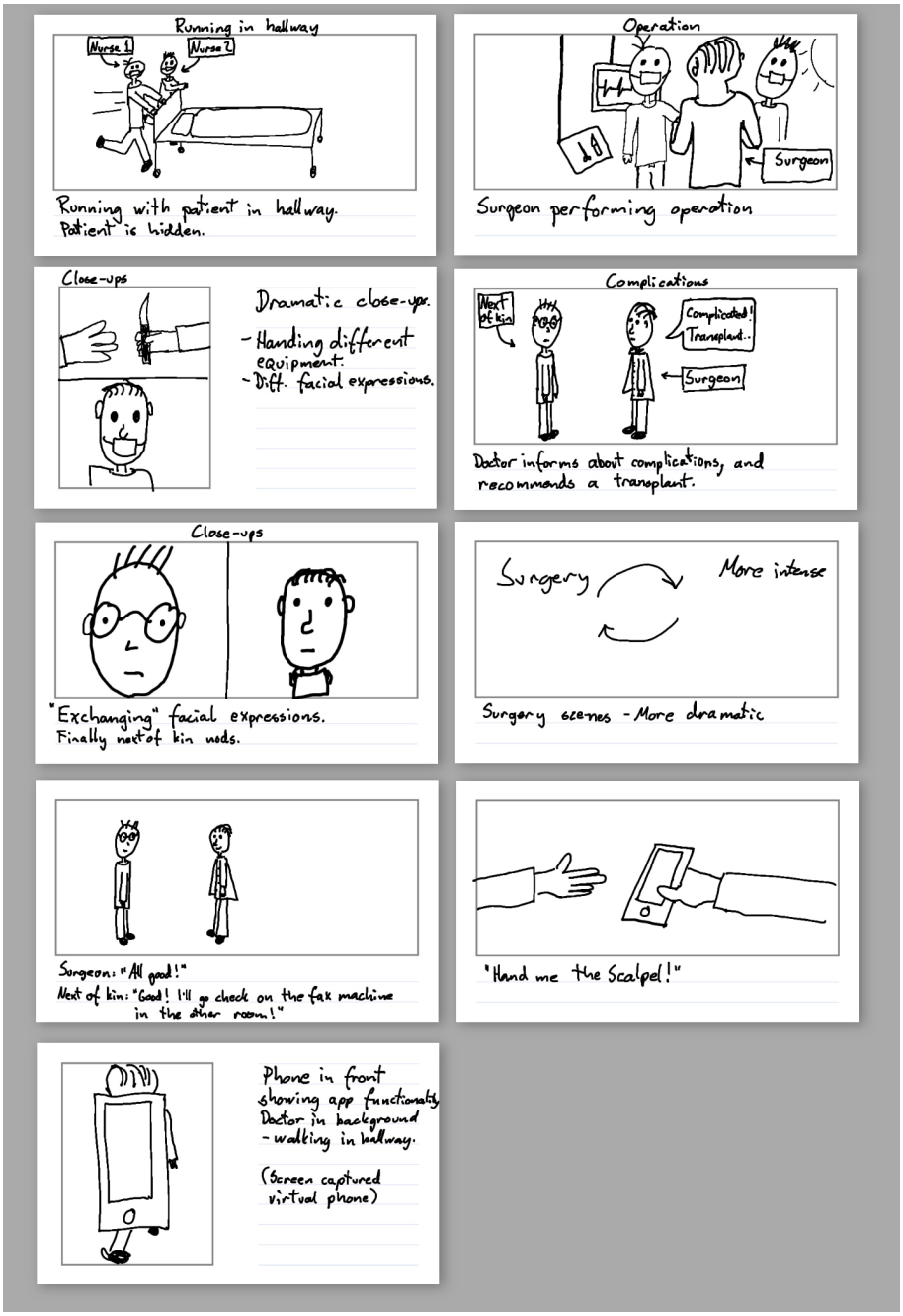


Figure 8.5: Storyboard for original video idea

8.9 Unimplemented User Stories

For various reasons some of user stories in the backlog were not implemented. This user stories were 'As a user I would like to get a tutorial the first time I log in' and 'As a user I would like to log in with a username and password'. The main reasons for this were limited time on the project. The team overestimated how many user stories that could be implemented during this project. But these user stories were considered low priority and it would not be crucial to have this functionality in the final product.

The customer also had a wish for our prototype to edit, move or cancel/reschedule operation in the start. But throughout the project the team together with the customer concluded that this functionality should not be a requirement for the product. The main reasons for this were that the mobile screen is too small to do edits, the hospital crew would like to do this on a computer. The application also is not yet able to communicate with OpPlan application layer, requiring more work on the customer side for setting up access between servers and the client, so methods implemented to edit, cancel or reschedule operations would have to be reworked in a real application. Because of these reasons and limited time on the project other functionality was prioritized.

Chapter 9

Testing

9.1 Overall Test Plan

Due to the open nature of the problem statement, the success of the application depended greatly on understanding the context of use and the needs of the users. The project goal stated that the application should "show the most important information from the desktop program quickly, efficiently and on-the-go", without a description of what this most important information is. The team had no prior experience of working in a hospital, so getting enough domain knowledge to make informed design decisions was the primary concern.

In addition, in the early talks with the customer it was decided that there was no expectation of a fully functioning implementation, and that the focus should be on the design and its usability, rather than the stability of the implementation.

With this in mind, the team decided that the main focus of the test plan should be usability testing. The team wanted to start testing the usability of design ideas early. Prototypes were sketched and given to the customer. In these early tests there were no checklists for fulfillment of requirements since the requirements were not specified yet. The team got feedback, and reacted quickly on them. In the later usability tests, more concrete and rigorous tables were made, as are described in the usability test paragraph.

While prototyping was being done, the team also started implementing. To get a design up quickly the team decided not to write automated unit tests while programming. The team did however use the available tools for doing manual testing when merging new changes into the main code base. In this process the reviewer did both a superficial manual unit testing, integration testing, and

code review.

9.2 Automated Testing

The team decided during planning to not put much emphasis on automated code testing. Read more about this in section 2.8.

9.3 Merge Testing

When a new user story was ready to be added to the main branch of the code, it first had to pass through a review. The story was first placed on the testing column on the scrum board, which indicated to the team that it was ready for testing. At the same time, a pull request was made on GitHub. Then a reviewer would look at the implementation. Usually this reviewer would be the code keeper.

In this project, integration testing was done continuously as new components were finished and ready to be merged into the main code base. When a new feature was ready for integration it would be moved to the testing column of the Scrum board and a pull request would be made on GitHub.

9.4 Functional Testing

9.4.1 Usability Testing

Usability testing was done during the fourth sprint. A set of representative users were selected to test the mobile application. It was not easy to get a lot of test subjects because people working at the hospital have a tight schedule. The subjects included one representative from each of the target groups; an operating nurse, a surgeon and a anesthesia doctor. The test subjects were given a set of tasks to complete during the test. These tasks can be viewed in Table 9.1. The testers were asked to think out loud while navigating through the application. While doing these tasks, no help was provided. One observer took notes and one assistant were filming the test. After completion, a discussion between the team and the testers took place. The results from the usability testing were carefully analyzed and discussed. The complete list of how the tasks were handled by the test subjects can be viewed in Appendix E.

The test leader followed the 10 steps of performing usability tests (Svanæs 2001):

1. Introduce yourself.
2. Describe the purpose of the test.

ID	User story	Scenarios
1	A12	Change the date to the 22nd of October 2017.
2	A1, A3	Find the patient on the operation in GA-2 starting at 08:11.
3	A5, A7	Find the blood type of the patient on the operation in GA-3 starting at 15:11.
4	A2	Show the operations as a list, change back to bar view after.
5	A18	Change the plan to Fødeavdelingen.
6	A8	Change date back to the 23rd of october. End the phase of the operation in ER starting at 13:07 today.
7	A24	Call up one person from the personnel of the operation occurring today.

Table 9.1: Scenarios for usability testing. Note that the 7th task was set up after the first usability test, as the first test subject requested this service without realizing it was already available in the application.

3. Tell participants they can quit when they want.
4. Describe the equipment and limitations of the prototype.
5. Teach how to think out loud.
6. Explain that you can not offer help during the test.
7. Describe the task and introduce the product.
8. Ask if they have any questions and run the test.
9. End the test by letting the user express themselves.
10. Use results.

The set of scenarios can be seen in Table 9.1.

9.4.1.1 Usability Testing with a Surgical Nurse

The first test object was a surgical nurse. She was mostly happy with the application, and really liked the timeline view. To show more information on such a small screen she wanted to flip the phone horizontally when viewing the timeline. Further, the ability to call directly from the app was important to her. As she normally used an iPhone she had some trouble with Android-specific elements. That is elements as the timepicker and to navigate back from the operation details using the back button that Android phones have. Another issue was to know which date the application was showing. Considering these issues, the team decided to add the date to the header of the application and

to add a back-button in the operation details view. As the hospital mainly will use Android phones as work phones, no further work will be done to make the application follow iOS standards.

During the test the surgical nurse did not try to zoom on the timeline. This is an important feature, the team decided it would be a good idea to add a manual to the on-boarding that will show up when the application is installed for the first time. The tester made some suggestions on how to improve the detailed view of the application. She also wanted to be able to move patients between theaters, especially between theaters in different operation views. The surgical nurse thought it was important that the application gave some additional value to her and her colleagues, and not just presented another way to do the same tasks as they could do with the dashboard application.

9.4.1.2 Usability Testing with a Surgeon

The second test session was with a surgeon from the movement center at the St. Olavs hospital. As an Android phone user he had more experience with the utilization of Android specific UI elements, such as the back button. Despite this several elements of the design was not immediately apparent, most notably the calendar button, the use of the menu in the top left, pinching to zoom in the timeline view and the implementation for ending and starting new phases. He noted fairly early on that as a surgeon he did not make much use of the graphical timeline view, much preferring the list view and finding this to work better overall.

When it came to the detailed view the user found the ordering of fields to be suboptimal, requesting that names and birthdates be moved up top, pushing procedure and diagnosis below these, as the most important thing to know for a surgeon was that he was indeed working on the right person. He also mentioned that the operation tab of detailed view showed more useful information, and that this could alternatively be the main view. The scrolling of details and swiping between tabs was not something that he found to be immediately known, but rather found by accident, and he also requested that one could traverse between tabs by clicking specific fields (for example when clicking the personnel field in the overview). The user found the option of calling personnel from the detailed view to be a nice addition as this was not something currently available in OpPlan, noting that several used the computer or catalogues for looking up numbers, but he would like the roles to display in the personnel tab rather than the names.

Overall he found the application to not be polished or functional enough when compared to OpPlan, and that too much information was initially hidden, before for example scrolling a view. In his opinion, as the application was to work as a supplement to rather than a substitute for OpPlan, it was a better idea for an early version to focus on adding new useful features that was not

available in the current system, and that users would then have an incentive to get used to the different interfaces and feature sets. In addition to this he suggested a feature of having notifications delivered for when for example an operation theatre was ready to receive the surgeon, an idea that was considered by the team when deciding on the framework. If there are delays due to miscommunication between the personnel responsible for different phases of an operation this could lead to problematic situations. At the end of the session the user was also questioned whether or not he would have been interested in a feature that displayed only the operations the user of the application was involved in, which he then responded to with interest.

9.4.1.3 Usability Testing with an Anesthesia Doctor

The third test was with an anesthesia doctor. The test went well, and she succeeded on six out of the seven assigned tasks, only failing to end the phase of the operation happening at the day of the test. Through the test some flaws of the application were discovered.

She thought some aspects of the application looked fine, but she had some problems understanding which parts of the application's user interface were clickable. During the test it was observed that the test person tried to click on static text, as well as not realizing that other parts were clickable. For example when wanting to call a crew member she did not understand at first that she could click on the telephone to make the call. She also attempted to click on the title in the header to go back to the starting screen.

Other problems she faced was with the material-ui guidelines. For example she did not think that the material-ui appbar looked like it was clickable or that you could scroll further or show other tabs. She also did not like that the title goes beneath the text in the operation details. This was default behaviour for the material-ui components.

The user also had some minor tweaks she thought was necessary. This included making it easier to tell that you could press on an operation in the list view of the operation to go into the detailed view. She also wanted an indicator on how far you could scroll on the x-axis on the timeline. The last thing she suggested was that the ratio between the widths of the rectangles of the actual and planned bars should be one.

9.4.1.4 Suggestions for Improvements

A complete list of suggestions for improvements made by the test subjects are shown in Table 9.2. It also shows whether or not the suggestions were addressed during the project.

The team did not address all of the suggestions. Changing operation details

and moving operations between theaters were considered out of the scope of this project, because the team did not have enough time to implement this. The customer wanted the team to focus on displaying data before implementing any possibility to change data. One user suggested to show messages like it is done in OpPlan, this possibility was not known by the team before this user test. Therefore it was decided to not implement this without doing further investigation. When clicking an operation the pop-up was improved to be more easily understandable, but the team decided not to include the start time of the operation and rather include more important information. The team found implementing notifications interesting, and believes that it could be an important feature to add in the future.

Some issues that the users experienced were due to the Material design and Android standards. Having the title under the text and buttons not looking clickable were design decisions carefully made by the team. It should not be changed without knowing that this is an issue for a lot of users.

Suggestion	Addressed
Include possibility to call personell	Yes
Move operation to different theater(also in different operation programs)	No
Show most important information on top in operation details overview	Yes
Input if patient is called and when it's 30 minutes left of an operation	No
Show 'smittefare'	Yes
Show 'hastegrad'	Yes
Add a 'back'-button to the operation details	Yes
Show the patient name as the header in the operation details	Yes
Show messages for all and messages per theater	No
Fit application better to 'flipped' phone	Yes
See start time and current phase when clicking operation	No
Add patient details to listview (gender and age)	Yes
Receive a notification when an operation you are a part of is ready	No
The "change phases" in the operation popup is not very good, at least it should have a different name	Yes
Show the time of each operation in the list view of all the operations	No
The ASA classification should be easily viewable	Yes
In the detail view it is strange that the title is under the information	No
It would be nice to have the information about where the patient is "hjemmeværende". Where he is staying	No
Show an indicator on how far it is possible to scroll on the x-axis in the timeline view	Yes

Table 9.2: Suggestions for improvements made by the users in the usability tests.

9.5 Non-functional Testing

9.5.1 Performance Testing

Considering that this application potentially can be used by several thousand users at a time, and also involves real-time update of data, the performance is expected to be reasonably good. If updating the application takes too long, it will not have any value for users that wants to follow operation progress in real time. To make sure that the application was able to handle frequent data updates and a large amount of users, React was used. Then the application can change data and states without reloading. When doing 'merge testing' the

team also checked that the new functionality did not decrease the performance notably. That was done both by observation and also by using the Chrome Developer Tools to analyze the performance. The team had some trouble with the responsiveness of the timeline, but managed to get it working fine when running it on test data. During this project the team were not able to do any tests involving many concurrent users, mainly because a test server were not sat up until the end of the project. Also, it was hard to gather enough users to make sure that several thousand users could use the application at the same time. This load testing is definitely something that should be done before releasing the product.

9.5.2 Security Testing

Although security is important for an application that contains confidential information, the main focus for this project was to create a user friendly front-end. As the customer already had their own security system for sending and retrieving data for the dashboard version, their plan was to integrate that system with the application after delivery. Considering this, the team did not spend time implementing security functionality, neither was time spent on security testing. Even though the security functionality already exists, it should be carefully tested after integration with the mobile application. That will be done by the customer after delivery.

9.5.3 Acceptance Testing

At the end of the project, acceptance testing was done by the customer. The goal was to determine that the requirements specified at the beginning of the project was met. Because the requirements consisted of desired functionality, the acceptance test mainly involved testing of functionality and not for example performance testing and security testing. Table 9.3 shows the functionality that was tested and accepted by the customer.

The customer had the following to say about the application: "The finished app is a much more polished product than we had initially expected. The app is not finished, it's missing a proper authentication process and some polish, but it is a very nice starting point. We are already looking at options for how to offer the app to the clinical personnel at our hospitals." (See appendix I)

Req. ID	Functionality	Approved
B1	**Operation timeline** As a user I want to view a version of the timeline view from OpPlan, which shows scheduled and performed operations distributed over a time axis. This should include both the planned time and the real progress of the ongoing operations. - View operation details by selecting an operation - Change operation program - Change date - Change phase of an operation - Zooming - Horizontal scrolling	Yes
B2	**Operation list** As a user I want to view a version of the operation list view from OpPlan, which lists scheduled and performed operations and their status. - View operation details by selecting an operation - Change operation program - Change date - Switch between different theaters	Yes
B3	**Operation details** As a user I want to be able to show the details of an operation, such as patient data and current status. - Switch between different information - Call operation crew - Change phases of the operation	Yes
B4	**Personal calendar** As a user I want to be able to show a timeline for the selected day with the operations involving the currently logged on user. This includes both the scheduled times and the progress of the ongoing operations. - View operation details by selecting an operation - Change date - Change phase of an operation - Zooming	Yes

Table 9.3: Functionality approved by the customer in the acceptance test.

Chapter 10

Evaluation

10.1 Team

The most important part of a functional team is communication, something that worked well from an early stage. The first thing that was done in the team was agreeing to set up a Slack channel for communication. This was a good decision as Slack is far more organized than for example a Facebook group. In hindsight it would probably have been a good decision to have two channels, one specific only to the students working in the team, and one for communication with customer. The customers created a Slack channel for the team, that they also were a part of. While they assured the team that they could use the channel as they saw fit, this still led to a natural sense of restraint in regards what team members were comfortable posting.

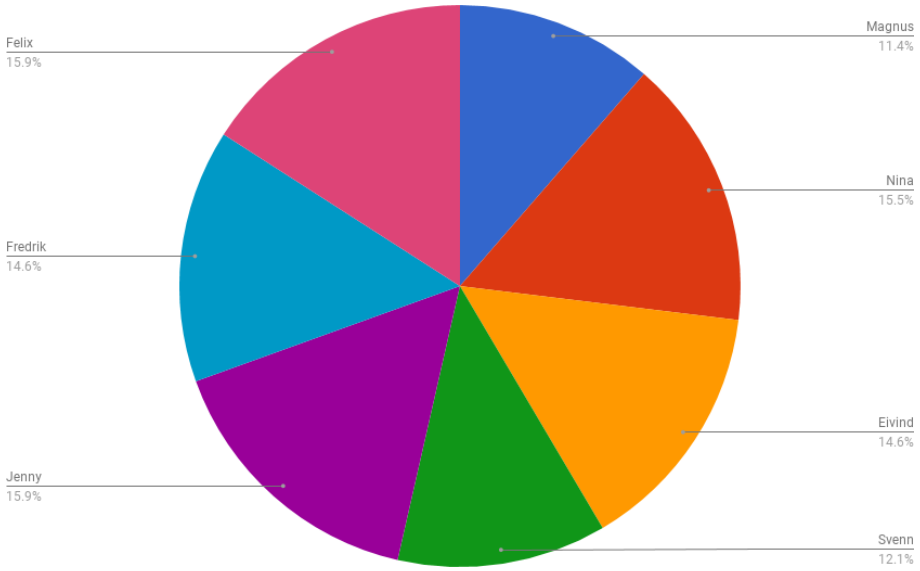


Figure 10.1: Time worked in the whole project in percentage

Overall team dynamics were also very good, with everyone taking their tasks seriously and not having issues with who the were assigned to work with. The team member responsible for human resources made sure to arrange team building activities and schedule breaks for the entire team during the project, so that the team members got to know each other outside of work hours. This led to a good working environment, where people respected each other and were not afraid to voice their opinion during disagreements. These measures improved the work balance between team members, as it is shown in the Figure10.1.

In the early stages in particular it was very hard to gather the entire team for internal meetings. This slowed down the decision making process considerably in the early sprints. This could perhaps have been mitigated by someone taking responsibility for listing the decisions that had to be made and asserting deadlines, so that everyone would be more engaged in the decision process early on.

Likewise, the handling of protocols was also considerably improved by creating templates and specifying who was supposed to write the protocol for each meeting. This should have been done from the very beginning. While meetings throughout the project took up a large amount of the working hours, they were not necessarily as intrusive as at first thought, often serving as very important factors to gauge the progress and adjust accordingly.

To ensure that everyone was committed to showing up on time for meetings, a team contract was created and signed by all members of the team, among

other things asserting what course of action should be taken if team members were late. This in turn led to the team being more careful in specifying the times that members were required to show up, where in the early stages there was ambiguity towards whether or not times specified were in academic minutes or not (for example 12:00 versus 12:15). As a result there was less waiting for meetings to start, not knowing whether someone would show up at the correct time or not, since attendance had to be specified the day in advance. There was some ambiguity regarding this statement in an early revision of the contract, and the penalty could have been more carefully considered, but overall this helped improving attendance to some degree.

The planning phases for each sprint is something that definitely improved a lot during the course of the project. Whereas the first sprints were poorly planned, with a vague goal and no estimation on tasks, the later sprints were much more carefully planned. This led to better estimates, better designation of tasks, and a common understanding of the progress of a sprint as we came along. Additionally good planing also could be a factor to encourage the team to work more. This is also shown in the Figure 10.2. While the team improved the planning phases the team also improved the raw working hours.

Something to learn from this is to take planning very seriously in early stages of the project, despite it seeming like we do not really know what we need to do. This can considerably speed up the process and improve the quality of the production in the early phases as well. Once the team were accustomed to Trello it worked very well as a planning tool for each sprint. While the full set of tasks were generally not completed during a sprint, the overall progress of the sprints, especially late in the project were satisfactory.

From sprint three to five the team divided into work pairs. This was to improve the quality of work by encouraging people to learn from each other and discuss solutions to problems together, improving quality assurance. This worked well, and was interleaved with individual work so that members of the team would have at least one person readily available if they were wondering on something. The pairs were different in each sprint. When using this method people felt responsible for looking after each other and help solving any issues their partner might have. Further, the team members had different knowledge of React and the pair work made it easier for those without experience to watch and ask questions, and thus learn quicker. Programming was initially performed like this, in pairs according to the XP practices with one person writing the code, but gradually evolved into each member assigning to themselves a specific sub task while also assisting each other during the sessions. Code reviews were still performed upon merge requests to include quality assurance while maintaining a higher efficiency with a single programmer.

Work-flow during the scheduled work sessions were good, with no member of the team deviating from working on the project. However the team was often too

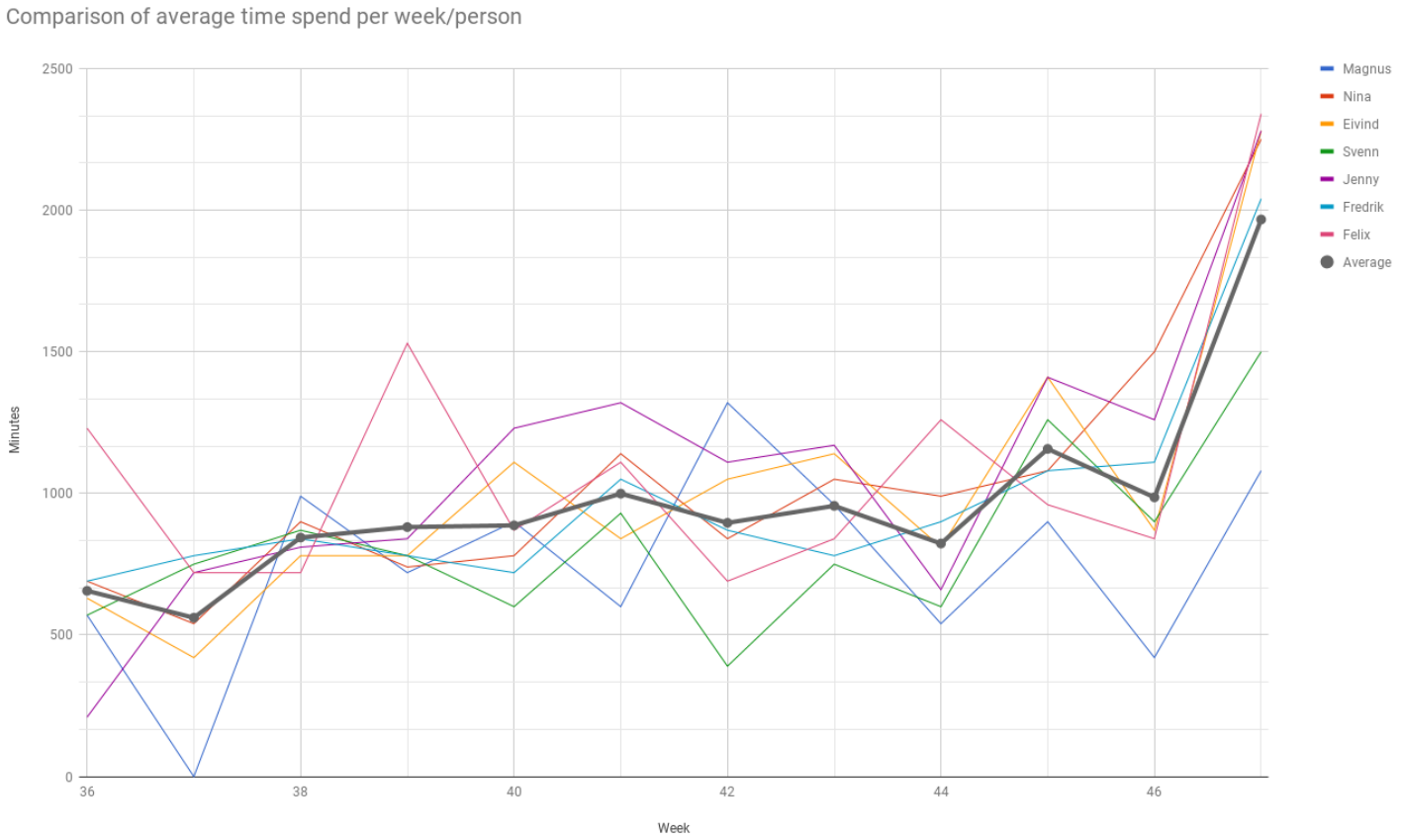


Figure 10.2: Time worked by team members in average per week

focused on their work to take breaks, especially during sessions at the customer offices. Around the third sprint scheduled breaks were planned for every session. Scheduling breaks for the entire team helped to keep a good work-flow without the demoralizing effect of extended periods of writing or programming. This was something that in hindsight should also have been done from the very start.

The communication in English went without any notable the issues. The important discussions were in English and the exchange student could communicate well. The language did not create a barrier internally. One issue was that a lot of the research had to be done in Norwegian. The questionnaire, the interviews and the user tests were done in Norwegian, to make it more comfortable for the subjects as they usually speak Norwegian in their work. The team considered it most important to get as much information as possible by letting them speak their main language. Also, the product owner would speak Norwegian and the other customer would sometimes forget to switch to English. This was usually solved by updating the exchange student afterwards or by reminding the customer to speak English.

10.2 Customer

Communication with the customer was throughout the project perceived as very good. As mentioned one of the first steps was to set up a communication channel in which both the team and the customer was included. Responses were typically very quick, and the customer was happy to help and provide feedback. The customer had assigned three people to act as contact people for the team to assure availability of at least one, but often enough all three were available for meetings and always one that could be reached during a work session.

The customer did not come forward as very strict about the criteria of the application, allowing the team creative freedom and options to explore the feature set. In order to aid the creativity and usability processes the customer also assisted with obtaining interview subjects as well as distributing a questionnaire that in turn led to obtaining users for usability testing.

The customer was also very helpful in renting rooms at their own offices, being very close to the campus, so that the team could sit there and work irrespective of whether or not there were planned meetings with the customer. Renting of rooms at Gløshaugen had mixed results, so the stability of only having a subset of rooms at a predefined location for every meeting and work session was welcome.

A issue that occurred, that the team had not properly preferred for, was that a contact person from the customer side for the second half of the project had a sick leave. Luckily the customer quickly assigned a new person to the task that was qualified to address the same issues as the one he was replacing.

In turn this also led to further discussions of having an API for the application to extract real data from the test system of OpPlan. Unfortunately this was introduced late into the project and the team could not take full advantage of the solution, but the possibility was explored and considered.

As a whole the customer was very welcoming, treating the team similarly to as if they were any other co-worker, and devoting a large amount of their time to answering questions and setting up systems the team required whenever asked.

10.3 Advisor

At the start of the project the team were assigned one advisor to help throughout the course of the project. Communication with the advisor was mainly through a weekly meeting every Friday, lasting roughly half an hour. During these meetings the advisor provided feedback on the report and made sure the team was on the right track by discussing the status report. The advisor was also particularly good with giving guidance with which parts of the project that needed work, whether it was the report or the technical parts.

On some occasions the advisor did not have the opportunity to meet. The team then was presented with the opportunity to arrange a meeting with another advisor, but the team chose to decline. Some of the reasoning behind this was that the advisor made small comments on the report regularly throughout the project. This made it easier for the team to continuously make rapid changes to the report without having to arrange a meeting.

The advisor was also available through email. So if the team needed answer to a question, response was fast and it was easy to get in touch. On some weeks however the advisor was unavailable for the whole week and could not answer any emails, this could make it hard to continue on the project without getting the answers that the team needed. This was not really a big issue for the team, but could easily pose difficulties for other teams in the future.

Overall the team experienced this supervising method as a good solution and don't propose any big changes for next year.

10.4 Project Task

The task assigned was an ideal choice for a student project such as this. Expectations of what the the customer wanted were explicit enough, seeing as how the customer already had a capable software solution for computers, but still allowed a large degree of freedom for the team to consider feature set, design, technology stack and even what mobile os to develop for. This even allowed for creative thinking with regards to what new features the team could

implement that would complement the platform of a smartphone better than a PC. A difficult aspect of this however was understanding the scope of the existing system. This led to a relatively slow start in order to chart what features were present, which would be useful on a phone, how much could be done during the course of the project and how much the team would have to change the format to actually display this on a much smaller screen. In a project context this can be considered beneficial, since it required the team to go through all the necessary steps of going from idea to product. The team went through phases of planning, design, interviews, testing, implementation and documentation, considering all of these steps important for the success of the project. Further work Being that scalpel is a software solution attempting to incorporate a majority of the features of OpPlan, a reasonably large program with a multitude of features, in an easily accessible way for mobile phones, it will essentially never be perfect. Optimizations in terms of presentation as well as performance are tasks that can still have long ways to go before reaching the quality expected from a final release. Presentation wise it is also worth noting that preferences of the order of information as well as graphical versus list based views are highly subjective. Implementations that incorporate learning of patterns of use or remembering the past session of a user are worth considering when improving the application.

Performance wise the main culprit for obstructing fluent behaviour is the implementation of the timelines using D3.js. Even on the highest end Android phones today there is a notable, while not entirely detrimental slowdown especially when scrolling and zooming. On computers the experience is noticeably smoother and the experience might automatically improve over time for phones as well with updates of the D3.js framework as well as improvements to software and hardware of Android phones, but simplifications and optimization to the graphical interface should be considered. When it comes to the frequent updates of information D3.js and React bring with them an innate advantage of only rendering a component over again when it's state has actually changed.

An initial suggestion for a feature, that there also surface interest for during the interviews, was the implementation of notifications. These could be used to for example notify a surgeon in advance when an operation theatre is ready to accept him, as delays when waiting simply waiting for a surgeon to arrive due to miscommunication is a considerable issue. This is a feature that was not implemented due to time constraints, and furthermore would not yield its full potential as a web application as it would require the application to be open at all times (not necessarily with the screen turned on). A notification system would thus have greater potential if the transition to a native mobile application was performed.

Another potential benefit that can be utilized for a native application is in terms of authentication, where biometrics such as fingerprints, irises or facial recognition could be used on modern phones for logging into the system. This could also be introduced for the web application by having an additional

companion app running natively on the device for handling authentication, but would not be as seamless. A transition to the use of biometrics however for logging in to these services is a process that could take time for hospitals, where solutions today mostly depend on the use of authentication cards, username and password. Suggestions for course improvement There are a number of small improvements that could be made to execution of the course that would benefit all parties involved. First of all, the use of the compendium for announcements was not ideal. For example when additional classes were introduced there were no means of being automatically notified that this change occurred, at least not without requiring the students to sign up for such notifications. Announcements should be available in the central hub where students expect to see information from all courses, such as blackboard, to reduce the effort necessary to stay updated.

It would also help to have more information throughout the execution of the project regarding how much progress is expected on report and development at certain intervals, rather than just the expected end result. Not saying that these should be hard deadlines, but it would make it easier for students to measure their progress and for advisor and customer to guide and provide feedback.

Evaluation criteria were not properly conveyed. Towards the end of the project a lot of confusion, and therefore frustration arose when being informed that evaluation was not performed as the students expected, for example stating that the short video was part of the grade, not having specified the target group beforehand, and that the customer would not be part of the evaluation. Upon requesting criteria 3 weeks before the end of the project, no specific details were given. Being a graded project worth 15 study points this absolutely needs to carefully specified.

10.5 Technical Challenges

In the early stages of the project issues relating to network access or restrictions occurred on multiple occasions. For example the first room the team rented had issues connecting to the network, forcing the team to work offline, in addition to the whole network being down one day as well during a work session. While the customer offices provided a stable connection there were some minor issues due to the nature of it being a closed network with several restrictions, as well as being disconnected from the data network.

One issue as a result of the network restrictions at the customer offices was that is was not possible to directly host the application from a computer to a phone for testing. This became especially apparent when the team prepared for the first usability test, with very little time to work out a different solution to perform these tests on a phone. For the places that had Eduroam access this would work directly, but otherwise it was required to use a mobile hotspot for

the phone to access the application.

The fact that the data network was locked off, with the team not able to access it from their own computers posed some problems with regards to implementing the API. Since communication with OpPlan was only possible on the data network this would require the team to host a server for the application on a computer within this network and for making the API calls. A phone connected to this network would then be able to run the test application in the browser.

During the early development phases when organizing the data structure and investigating which features to include there also existed a restriction on when and where the team could access the existing system of OpPlan. The team would require a computer at the customer offices that could open OpPlan and be certified through the use of a PULS authentication card paired with credentials. In conjunction with the test data existing at the time not being fully detailed, this led to the team not being able to explore the existing system quite to the extent desired.

Aside from this no other noteworthy technical challenges occurred that were not considered to be part of any typical software development project.

Bibliography

Atieh Khanjani, Riza Sulaiman (2011). "The Aspects of Choosing Open Source versus Closed Source". In: p. 648.

Beck, Kent (1999). *Extreme programming explained: embrace change*. Addison-Wesley Longman Publishing.

Burgess, Dr. T. F. (2001). "Guide to the Design of Questionnaires". In: URL: http://iss.leeds.ac.uk/downloads/top2.pdf.

Carlsen, H. and G. Zakariassen (2014). *Nordmenn elsker fortsatt Iphone – mens store deler av verden velger andre alternativer*. URL: https://www.nrk.no/norge/nordmenn-elsker-fortsatt-iphone-1.11922649.

Cohn, Mike (2010). *Succeeding with agile: software development using Scrum*. Pearson Education.

Color - Style (n.d.). URL: https://material.io/guidelines/style/color.html#color-color-system.

Cormen, Thomas H (2009). *Introduction to algorithms*. 3rd ed. MIT press.

Dascalescu, Dan (2016). *Why "Progressive Web Apps vs. native" is the wrong question to ask*. URL: https://medium.com/dev-channel/why-progressive-web-apps-vs-native-is-the-wrong-question-to-ask-fb8555addcbb.

Denning, Steve (2015). *Agile: The World's Most Popular Innovation Engine*. URL: https://www.forbes.com/sites/stevedenning/2015/07/23/the-worlds-most-popular-innovation-engine/#2d095ece7c76.

Designing for iPhone X (n.d.). URL: https://developer.apple.com/design/.

Developing for Android vs. iOS: Material vs. Flat Design (2016). URL: https://medium.com/@jrejaud/developing-for-android-vs-ios-material-vs-flat-design-fb341b05b0f0.

Dybå, Tore and Torgeir Dingsøyr (2008). "Empirical studies of agile software development: A systematic review". In: *Information and Software Technology* 50.9-10, pp. 833–859. DOI: 10.1016/j.infsof.2008.01.006.

Fowler, M. (2009). *FeatureBranch*. URL: https://martinfowler.com/bliki/FeatureBranch.html.

Juric, Radmila (2000). *Extreme Programming and its Development Practices*. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=915842.

Martin, Robert C. (2008). *Clean Code: A Handbook of Agile Software Craftsmanship*. 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR.

Material Design (n.d.). URL: https://material.io/guidelines/#.

Material motion - Motion (n.d.). URL: https://material.io/guidelines/motion/material-motion.html#material-motion-what-makes-a-good-transition.

Metrics and keylines (n.d.). URL: https://material.io/guidelines/layout/metrics-keylines.html#metrics-keylines-keylines-spacing.

Moløkken-Østvold, Kjetil, Nils Christian Haugen, and Hans Christian Benestad (2008). "Using planning poker for combining expert estimates in software projects". In: *Journal of Systems and Software* 81.12, pp. 2106–2117.

Navigation (n.d.). URL: https://material.io/guidelines/patterns/navigation.html#navigation-defining-your-navigation.

Norman, Donald (1988). "The design of everyday things". In: *Doubled Currency*.

Preece, J., Y. Rogers, and H. Sharp (2015). *Interaction Design: Beyond Human-computer Interaction*. 4th ed. John Wiley & Sons Ltd.

Smartphone OS Market Share, 2017 Q1 (2017). URL: https://www.idc.com/promo/smartphone-market-share/os.

Sommerville, Ian (2015). *Software Engineering - Global Edition*. Pearson Education.

Svanæs, D. (2001). *Prototyping og brukbarhetstesting*. URL: http://dag.idi.ntnu.no/IT3402_2009/prototyping_brukbarhetstesting.pdf.

Chapter 11

Appendices

A User and Installation Guide

The application uses the power of npm to make the installation and running of the application as simple as possible. The only external dependency that is necessary is Node.js [1].

To get access to the code base, go to the repository [2]. Either clone the repository or download a zip. Then navigate to the root directory of the project in your favorite terminal, and run the command `npm install`. This will install all of the required dependencies. Then there are two options. To start a development server, run the command `npm start`. Then you can view the application at address `localhost:3000` in your browser. The other option is to generate a production build by running the command `npm run build`. This will bundle and minify everything. The built version is but in the /build directory. This may then be deployed to a web server.

Since the application is a single page application, all routing is done on the client side, instead of on the server. Depending on the web server there might be some configuration necessary to hand the routing over to the client. Technical/internal documents

B Questionnaire

This is what the questionnaire looks like, translated into English:

[1] Installation: https://nodejs.org/en/
[2] https://github.com/egrimstad/scalpel

OpPlan Mobile Application
A mobile application will cover some parts of OpPlan, and function as a supplement to the current program. We are looking at the possibilities of making an application that can be used on a mobile unit, for example a smart phone. To come up with a solution that is customized for everyday use at the hospital, we wish that you take some of your time to answer these questions. Thanks in advance for your response.

Question 1.
What occupation do you have at the hospital?

a) Anesthesia doctor

b) Surgeon

c) Nurse: Anesthesia

d) Nurse: Operation

e) Cleaning

f) Other...

Question 2.
What hospital do you belong to?

a) Trondheim

b) Orkdal

c) Røros

d) Volda

e) Ålesund

f) Molde

g) Kristiansand

h) Levanger

i) Namsos

Question 3.
Select your gender.

a) Male

b) Female

Question 4.
Select your age.

a) Below 30 years

b) 30-50 years

c) Above 50 years

Question 5.
How long have you been using OpPlan?

a) Less than 2 years

b) 2-5 years

c) More than 5 years

Question 6.
How would you rate your understanding of technology?

a) 1 (Very bad)

b) 2

c) 3

d) 4

e) 5 (Very good)

Question 7.
What kind of phone do you have?

a) (iOS) iPhone
b) (Android) Samsung, LG, Huawei, Sony, osv.
c) Windows
d) Don't know
e) Other...

Question 8.
How much time do you spend by a computer per day in your job?

a) Less than 0.5 hour
b) 0.5-2 hours
c) 2-5 hours
d) More than 5 hours
e) No time

Question 9.
How much time do you spend using OpPlan during a day?

a) Less than 0.5 hour
b) 0.5-2 hours
c) 2-5 hours
d) More than 5 hours
e) Do not use OpPlan

Question 10.
How do you use OpPlan? (Select several)

a) Long continuous periods
b) Short intervals
c) Passive use/reading (OpPlan in the background)
d) Other..

Question 11.
What are you using OpPlan for? (Select several)

a) Get an overview of tasks

b) Get an overview of and complete tasks for today ('dagen i dag')

c) Time scheduling

d) Lookup of orientation and progress

e) Schedule patients ahead of time (admissions office)

f) Documenting

g) Other..

Question 12.
Do you see the benefit of a mobile application as a supplement to OpPlan?

a) Yes

b) Maybe

c) No

d) I don't know

Question 13.
Do you have any suggestions for what a mobile application for OpPlan should contain?

Question 14.
Is there something you dislike with OpPlan as it is today?

Question 15.
Is there something you dislike with OpPlan as it is today?

Question 16.
If you have other comments or input, write them here.

Question 17.
Do you wish to participate in a usability test? If so, enter name and email.

C Sketches

Early in the project design sketches were made. It was important to have different options to discuss. These sketches can be viewed in the following figures.

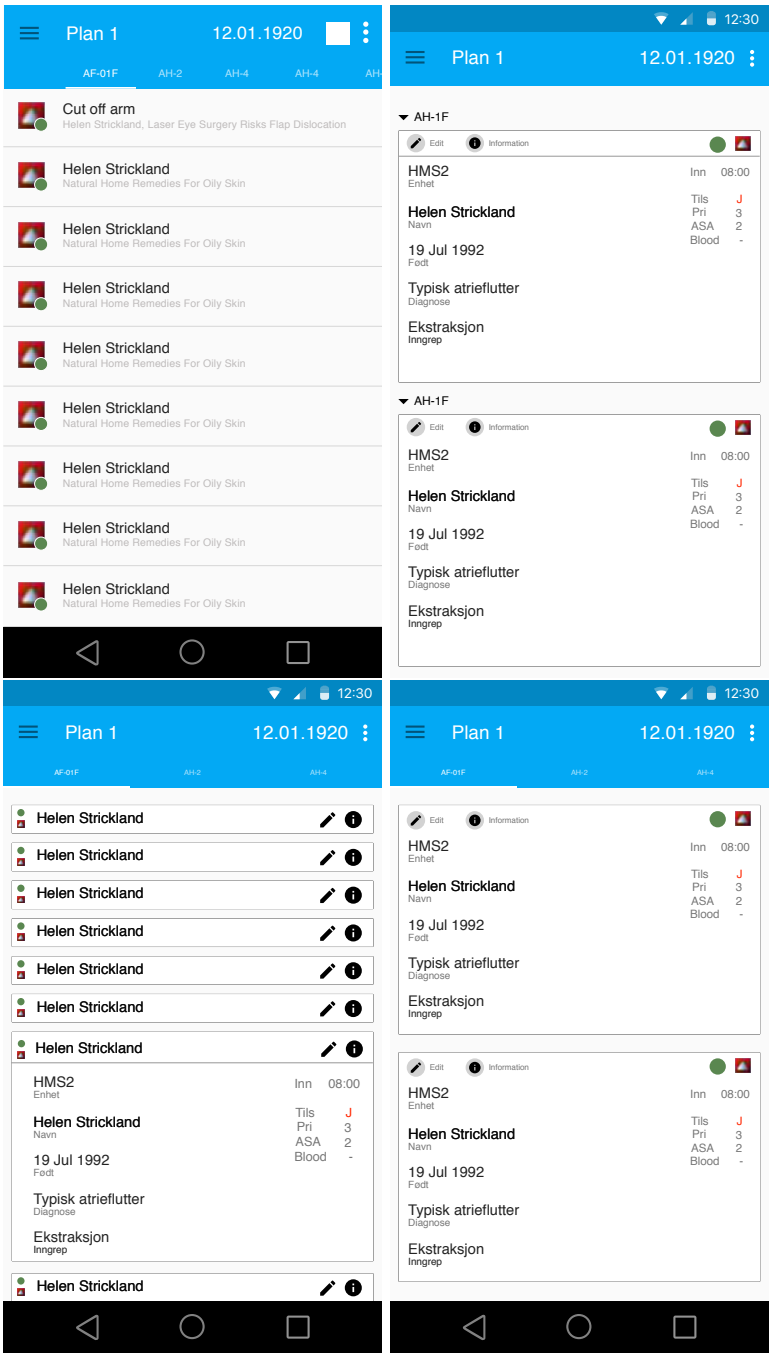


Figure C.1: Sketches with different options for the list view.

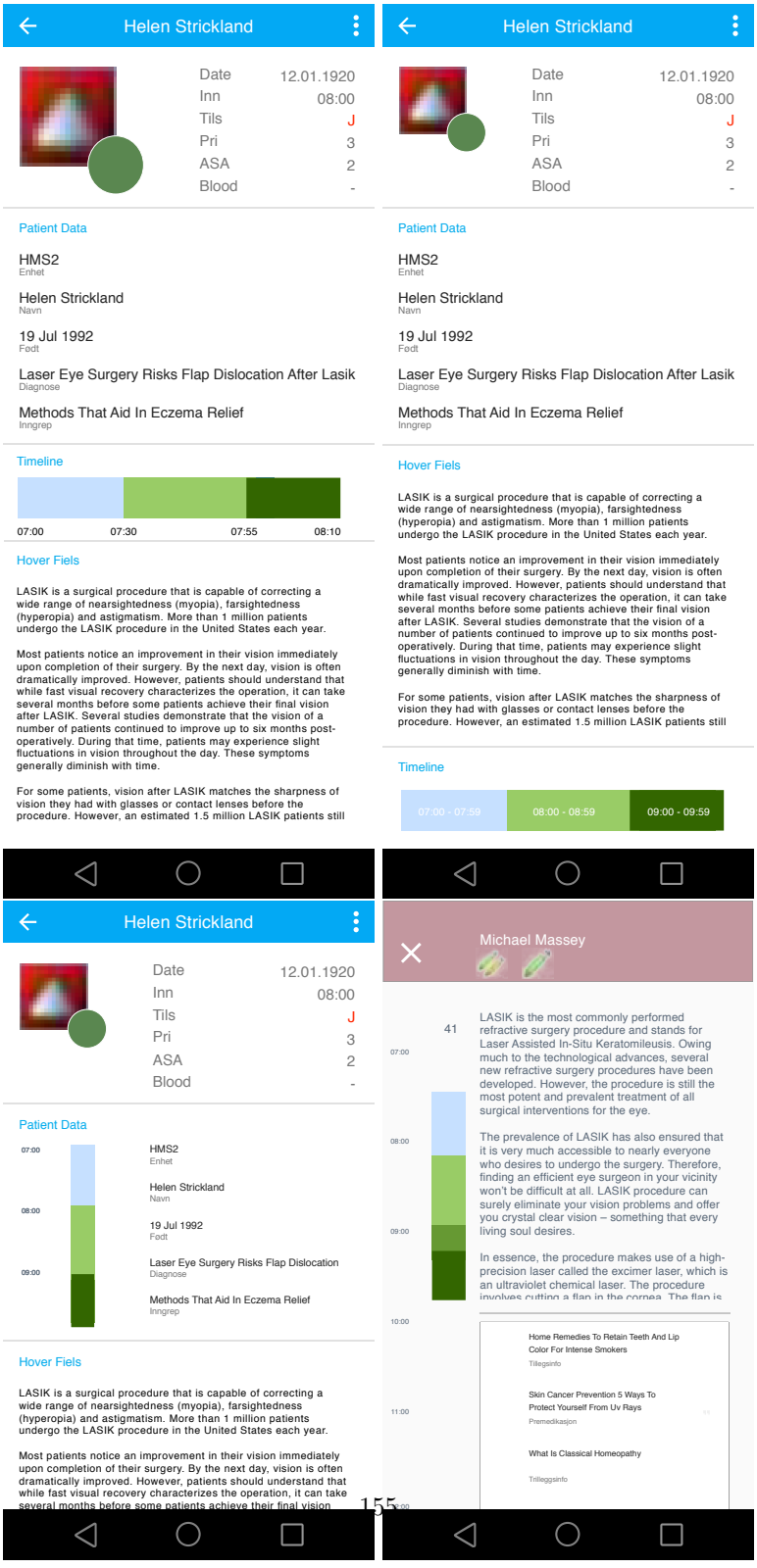


Figure C.2: Sketches with options for the operation details view.

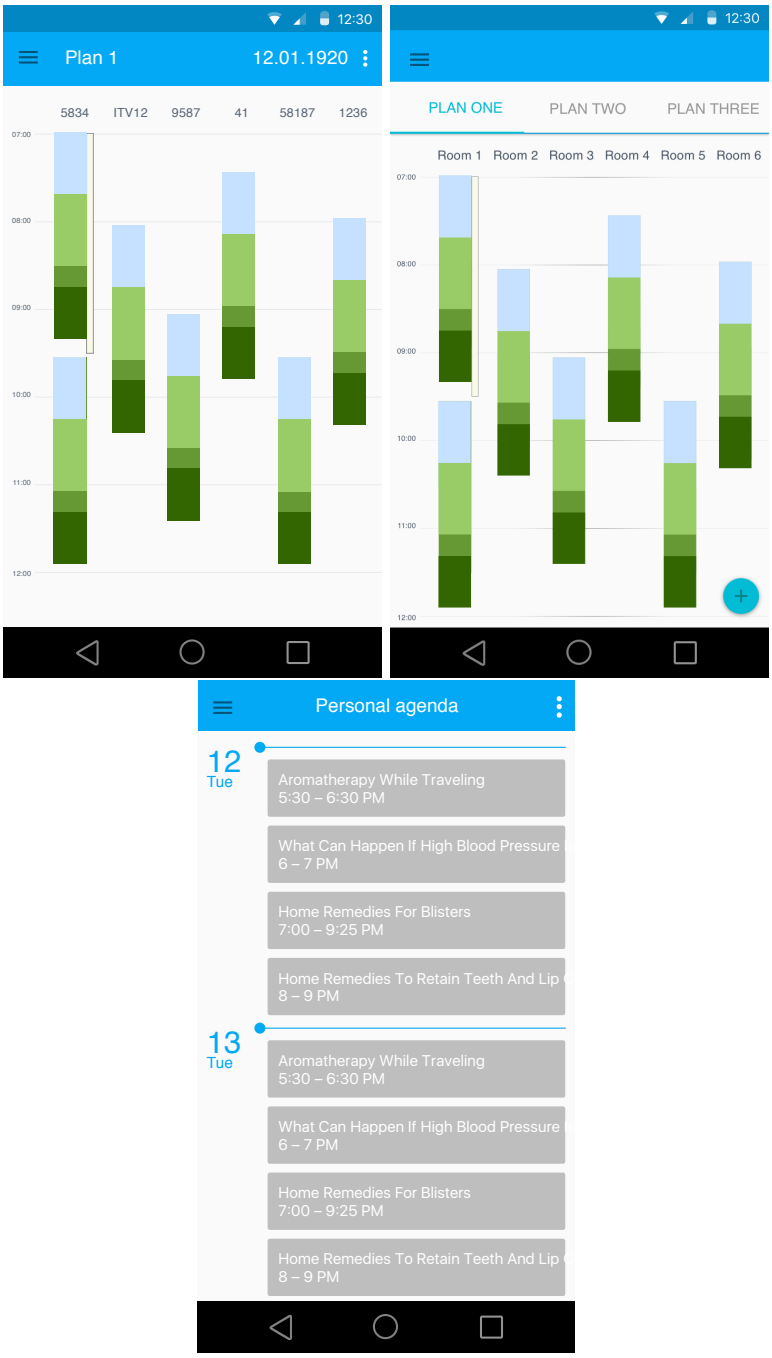


Figure C.3: Sketches for the timeline and the personal plan.

D Clickable Prototype

Screenshots from the final version of the clickable prototype made in InVision. The first figures shows onboarding when the application is first installed. Note that this in not implemented in the final product. The next figures shows the timeline and the details view. Menus for changing date and changing the phase of an operation are shown in the last figures. Some changes were made after the product were tested by users.

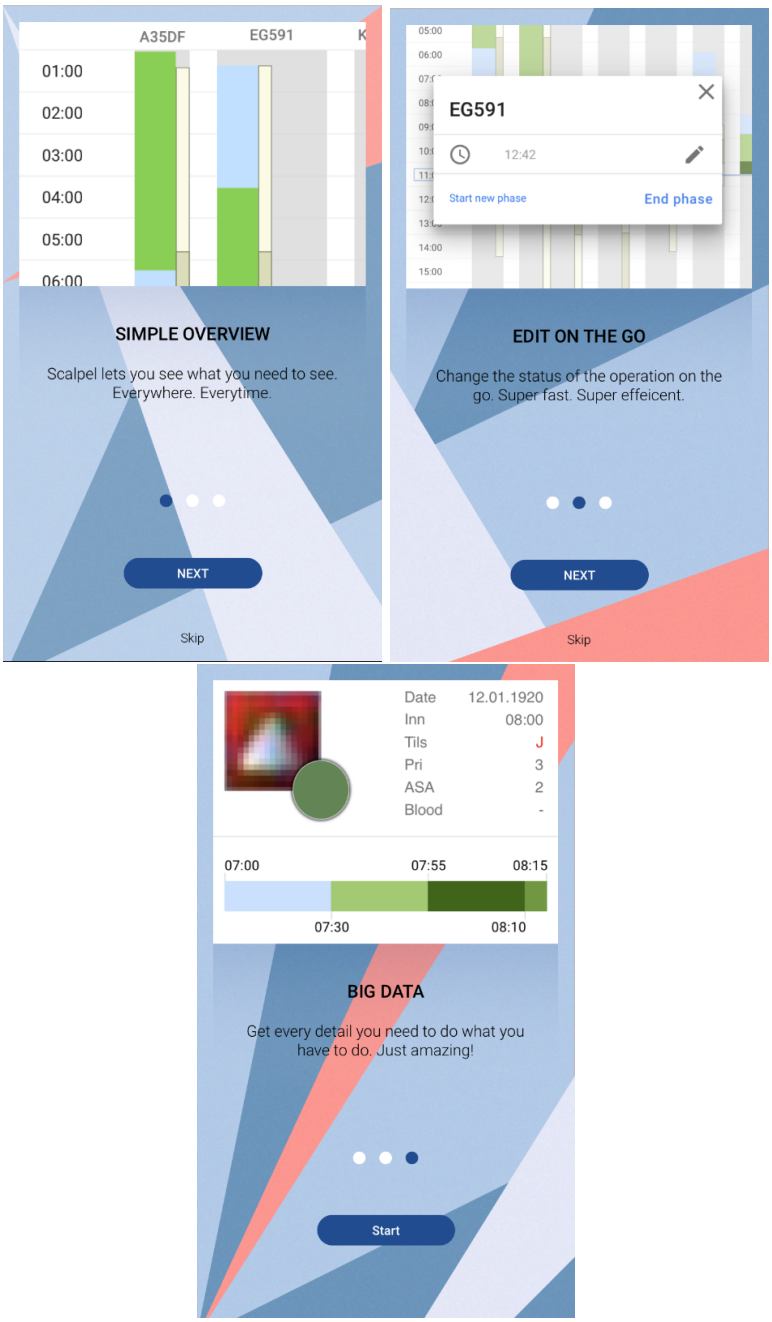


Figure D.4: Onboarding in the clickable prototype

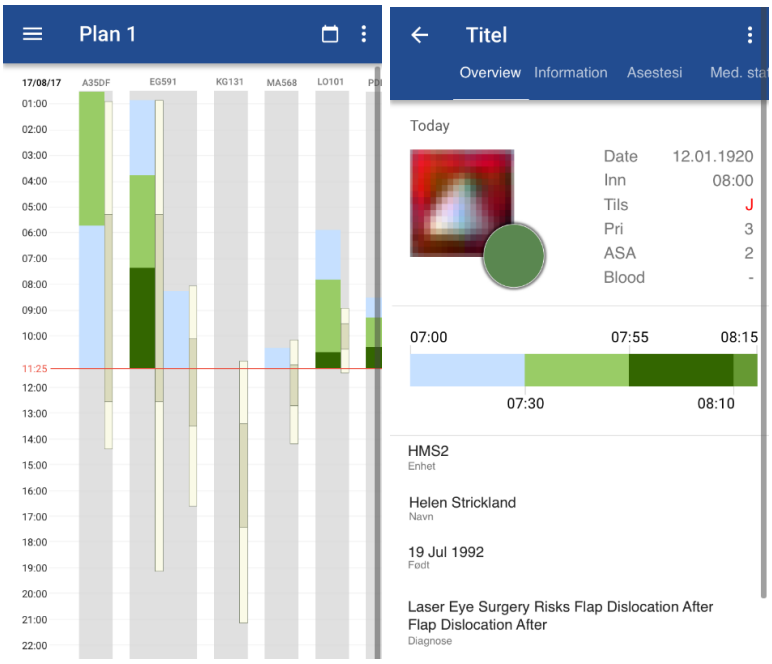


Figure D.5: The timeline and the details view in the clickable prototype

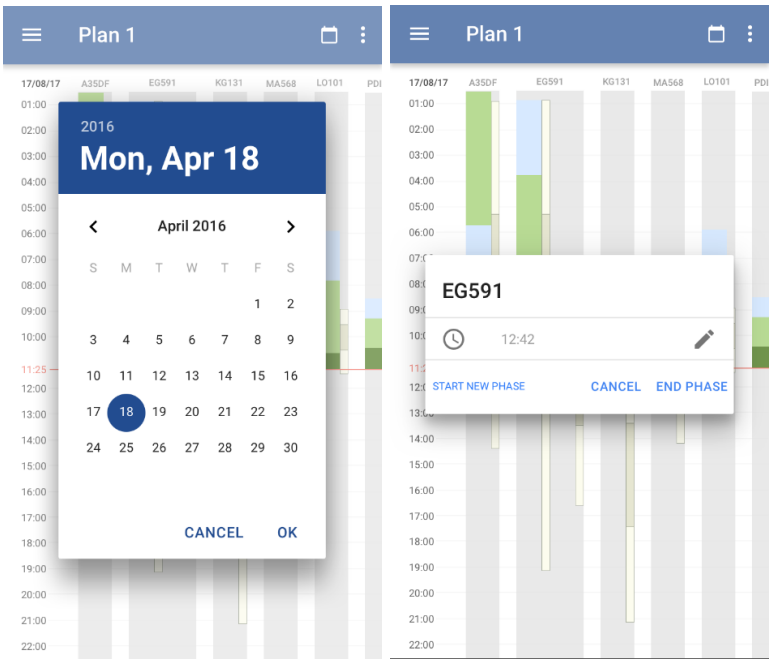


Figure D.6: Menus for changing date and phase in the clickable prototype

E Usability Testing

The tasks that were presented to the test subjects are shown in Table 9.1. A complete list of how the test subjects handled the tasks can be viewed in the tables below.

ID	1
Behavior	Clicks on calendar icon, selects date
Errors	None
Completion	Success
Comments	

ID	2
Behavior	Looks up correctly in the timeline(without zooming) and clicks on operation, but wants to confirm that she selected the correct patient. Tries to look at patient details. Struggles to get back
Errors	Navigates to patient details to try to find start time of operation
Completion	Success
Comments	User wants to make sure that correct operation is clicked in timeline. Some technical issues during this task

ID	3
Behavior	Navigates to patient details from timeline view easily. Looks at anesthesia tab, then operation tab and finally finds blood type in overview
Errors	Navigates away from overview before navigating back
Completion	Success
Comments	Blood type is not the best attribute to look for, usually not used, expects to find it under anesthesia

ID	4
Behavior	Stops to think and looks at the possible menu options. Then selects correctly the side menu and navigates to list view and back to timeline
Errors	None
Completion	Success
Comments	

ID	5
Behavior	Directly selects the menu bar and then changes
Errors	None
Completion	Success
Comments	Probably remembers to have seen this earlier in the test
ID	6
Behavior	Changes date as in task 1, selects operation (same problem as task 2) and stops to think for a second before selecting the date picker. Struggles to use the Google datepicker but completes task successfully
Errors	None
Completion	Success
Comments	Used to iPhone

Table E.6: Usability testing with surgical nurse. Note that this test had one less task.

ID	1
Behavior	Clicked back, clicked datepicker when given a hint. Chose the wrong date
Errors	Clicked first to go back, had issues finding the datepicker and menu button until given a hint. Misclicked first on the date
Completion	Failure (success after one hint)
Comments	Would like to see the current chosen date somewhere

ID	2
Behavior	Clicked the correct phase, had issues seeing the time
Errors	A bit confused about the end phase time. Think this represents the start time of an operation
Completion	Failure
Comments	

ID	3
Behavior	Found the correct operation and patient details without too much issues
Errors	Tried pressing overview initially to go back to the timeline
Completion	Success
Comments	

ID	4
Behavior	Not quite sure where to start here, eventually finds the correct option after given a hint about the top left menu
Errors	Accidentally hits the home button and calendar while looking for the button. Cycles through the options in the detailed view.
Completion	Failure (success after one hint)
Comments	

ID	5
Behavior	Looked into the menu and selected the correct option
Errors	None
Completion	Success
Comments	

ID	6
Behavior	Clicked operation and first clicked details before clicking ‘end knife time’
Errors	Went into operation details and navigated back
Completion	Success
Comments	Did not find it very intuitive, did not think ’end knife time’ was the goal even though he quickly noticed this button

ID	7
Behavior	Quickly navigated to the personnel tab and clicked a number
Errors	None
Completion	Success
Comments	Was already in the details when given the task

Table E.6: Usability testing with surgeon.

ID	1
Behavior	Pushed the calendar icon and chose the correct date
Errors	None
Completion	Success
Comments	Understood it right away

ID	2
Behavior	Pushed on the operation in the timeline view. Found the information in the popup
Errors	None
Completion	Success
Comments	Completed the task fast and with no trouble

ID	3
Behavior	She pushed back from to the timeline. Pushed the operation and pushed the “more details”-button. Used a little time to find out where the information about the blood type were
Errors	Did not understand that she could scroll down to see more information
Completion	Success
Comments	Had a hard time figuring out which theatre she was on when she was in the operation details view

ID	4
Behavior	She tried to push title where it said ’ortopedisk avdeling’ to get out of the operation details. Then she pushed the menu icon. She finally got out of the operation details when pushing the return button on the phone. Then she pushed the menu icon and then the ’list view’
Errors	Tried pushing the title
Completion	Success
Comments	A back button could be implements. The operation details should not show the department as the header.

Table E.6: Usability test with an anesthesia doctor.

ID	5
Behavior	Pushed the menu icon and then selected 'Fødeavdeling'
Errors	None
Completion	Success
Comments	

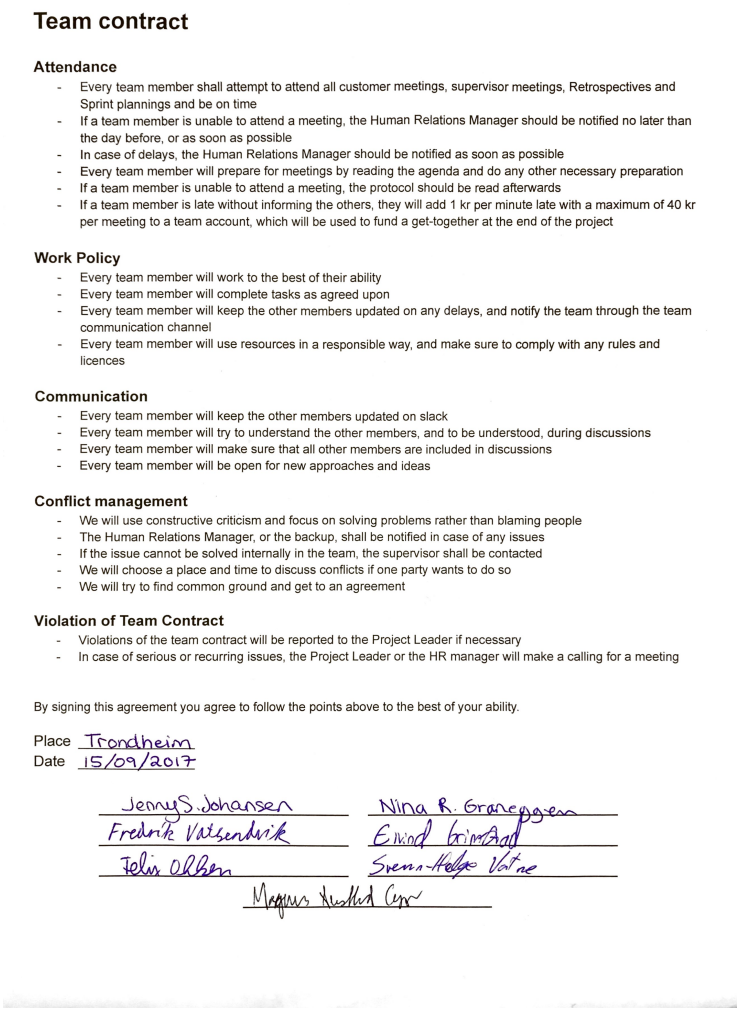
ID	6
Behavior	She pressed the right operation. Went to more details. Pressed the 'Fødeavdelingen' option. She scrolled a lot in the operation details. When she finally found the change phase in the operation detail she tried to push the title
Errors	Went into operation details. Tried to push the title in 'change phases'
Completion	Failure
Comments	She commented that it was not easy to know what was touchable. She mentioned that "change phases" was not the right words to use. She thinks "change time" is more appropriate

ID	7
Behavior	Changed the date to 22th of October. She pressed the right operation. Went into the details view. Scrolled to the bottom of this view and did not touch the app bar. Pressed the personal text on the bottom. Then she saw the appbar and the pushed the personell tab. Completed the task by calling.
Errors	Tried to press a text field
Completion	Success
Comments	She thinks that it is not clear enough that the app bar is possible to press

Table E.6: Usability testing with anesthesia doctor.

F Team Contract

The team contract was signed by all team members at the very start of the project. The team contract can be viewed in Figure F.7.



Team contract

Attendance

- Every team member shall attempt to attend all customer meetings, supervisor meetings, Retrospectives and Sprint plannings and be on time
- If a team member is unable to attend a meeting, the Human Relations Manager should be notified no later than the day before, or as soon as possible
- In case of delays, the Human Relations Manager should be notified as soon as possible
- Every team member will prepare for meetings by reading the agenda and do any other necessary preparation
- If a team member is unable to attend a meeting, the protocol should be read afterwards
- If a team member is late without informing the others, they will add 1 kr per minute late with a maximum of 40 kr per meeting to a team account, which will be used to fund a get-together at the end of the project

Work Policy

- Every team member will work to the best of their ability
- Every team member will complete tasks as agreed upon
- Every team member will keep the other members updated on any delays, and notify the team through the team communication channel
- Every team member will use resources in a responsible way, and make sure to comply with any rules and licences

Communication

- Every team member will keep the other members updated on slack
- Every team member will try to understand the other members, and to be understood, during discussions
- Every team member will make sure that all other members are included in discussions
- Every team member will be open for new approaches and ideas

Conflict management

- We will use constructive criticism and focus on solving problems rather than blaming people
- The Human Relations Manager, or the backup, shall be notified in case of any issues
- If the issue cannot be solved internally in the team, the supervisor shall be contacted
- We will choose a place and time to discuss conflicts if one party wants to do so
- We will try to find common ground and get to an agreement

Violation of Team Contract

- Violations of the team contract will be reported to the Project Leader if necessary
- In case of serious or recurring issues, the Project Leader or the HR manager will make a calling for a meeting

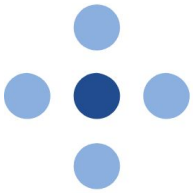
By signing this agreement you agree to follow the points above to the best of your ability.

Place Trondheim
Date 15/09/2017

Figure F.7: Team contract

G Meeting templates

The following pages contain the meeting templates that were used during the project.



Customer meeting agenda

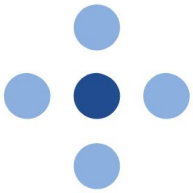
TDT4290 Customer Driven Project - Group 10

When
Where
Who

Agenda

Preparations

Practical
Responsible for writing minutes:



Customer meeting minutes

TDT4290 Customer Driven Project - Group 10

When
Where
Who

Topic

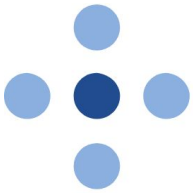
Advisor meeting agenda

TDT4290 Customer Driven Project - Group 10

When
Where IT building, ITV-157
Who Letizia, Eivind, Felix, Fredrik, Jenny, Magnus, Nina, Svenn-Helge

1. Approval of agenda
2. Approval of minutes from last advisor meeting
3. Comments to the minutes from last customer meeting or other meetings
4. Approval of status report
5. Review/approval of attached phase documents
6. Issues

Practical
Resonsible for writing minutes



Advisor meeting minutes

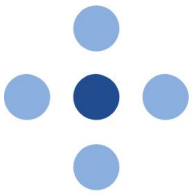
TDT4290 Customer Driven Project - Group 10

When

Where IT building, ITV-157

Who

Topic



Status report

TDT4290 Customer Driven Project - Group 10

Sprint:
Week:

Summary

Work done in this period

Status of the documents that are being created
Meetings
Other activities

Challenges

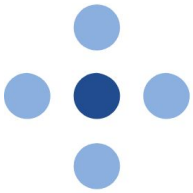
What is interfering with the progress or taking resources? Problems are often risks that have taken effect.

Planning of work for the next period

Meetings
Activities

Other

Review/approval of attached phase documents
Other issues



Student meeting

TDT4290 Customer Driven Project - Group 10

What have you done since the last meeting?

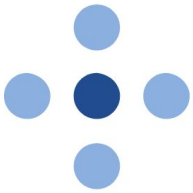
Eivind:
Felix:
Fredrik:
Jenny:
Magnus:
Nina:
Svenn-Helge:

Problems that have come up

Eivind:
Felix:
Fredrik:
Jenny:
Magnus:
Nina:
Svenn-Helge:

What will you do today?

Eivind:
Felix:
Fredrik:
Jenny:
Magnus:
Nina:
Svenn-Helge:



Sprint Retrospective

TDT4290 Customer Driven Project - Group 10

When
Where
Who

What went well during the sprint cycle?

What went wrong during the sprint cycle?

What could we do differently to improve?

H Time Tracking

All team members tracked their working hours during the project. The following pages shows the template for time tracking and a summary of hours spent in total by all team members.

How to write hours	
Meetings:	Any kind of meeting (not work sessions)
Classes:	Going to class.
Organizing:	Planning/organizing something that is related to the people in our group. Meetings are the exception!
Architecture:	Working on the overall structure, ie. product planning and designing.
Learning/researc	Learning something new, looking for solutions (online), general reading about Software Engineering.
Programming:	Actually writing code.
Testing:	Testing code and systems.
Documentation:	Doing things related to documentation or the report.
Other:	Write something in the "Comments" section.
Budget:	24h / week (1440 min / week)

Date	Meetings	Classes	Organizing	Architecture	Learning/Research	Programming	Testing	Documentation	Other	Working minutes	Working hours	Hours in sprint	Comment
8/28/2017										0	0		
8/29/2017										0	0		
8/30/2017										0	0		
8/31/2017										0	0		
9/1/2017										0	0	0	

Figure H.8: Template for hour recording including some example dates

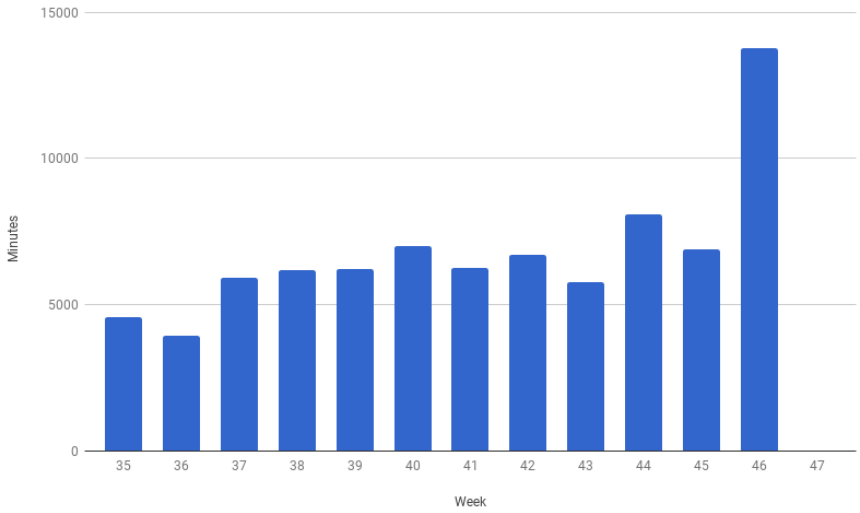


Figure H.9: Total minutes worked by the whole team per week

I Customer Feedback

After the fifth sprint the team got the following feedback from Rune Andreas Grimstad at HEMIT:

"Handling the logistics related to surgeries is a mission critical task for all

hospitals. Hospitals in Central Norway perform this task using an application developed by HEMIT called OpPlan. This is a very large and complex application that requires the user to have access to a Windows computer.

For the last two years, HEMIT has been exploring options for how to build a light-weight version of OpPlan for mobile devices, but unfortunately we haven't had the time to follow up this work. We therefore decided to suggest this as a task for the TDT4290 Customer Driven Project course. The task was fairly open so our expectations were that the students would do a concept study and deliver a design sketch and possibly a clickable prototype. We did not expect the students to have time to develop a functional app, so they have delivered far beyond our initial expectations.

Much of the time spent in the project was used to perform a study of the requirements of our users. The students performed several interviews with personnel working with OpPlan and they distributed a questionnaire to a large group of users. The resulting knowledge will be invaluable for HEMIT going forward with the project and has been a real eye opener for us.

The finished app is a much more polished product than we had initially expected. The app is not finished, it's missing a proper authentication process and some polish, but it is a very nice starting point. We are already looking at options for how to offer the app to the clinical personnel at our hospitals.

Being part of the TDT4290 Customer Driven Project course has been very interesting and I hope we are able to participate with an interesting project next year as well."