Angelika

Using Wearable Devices in Welfare Services

Haaland, Torgeir Hovind, David Jordal, Iver Le, Tan Quach

Sandve, Sigurd Solheim, Martin Vassli, Lars Tore

Group 11
Autumn 2014
TDT4290
Customer Driven Project

NTNU – Trondheim
Norwegian University of
Science and Technology

The team would like to thank the advisor, Jon Atle Gulla, the customer contacts from Trondheim Kommune, Lise Høiberg, Tone Dypaune and Kirsti Fossland Børs, Designhjelpen at NTNU, and all the other people the team contacted during the course of the project for invaluable help, feedback and guidance.

Abstract

Senior citizens often want to live in their own homes as long as possible, which can be achieved with assistance from healthcare professionals. However the current state of communication technology in Trondheim for care of the senior citizens is limited to telephone and other analogue solutions, and there is a growing need for more healthcare professionals. Technology is evolving rapidly in other parts of our society with smart phones and small interconnected embedded computer systems. This gives an incentive for the work presented in this report, a solution that can make the job for healthcare professionals more manageable by utilizing a wearable sensor for recording vital signs.

This report describes the development process of the healthcare support system Angelika, a system that uses an off the shelf fitness tracker to monitor a user's vital signs. This data is then transferred securely to a server where it is analysed in order to find anomalies in the values compared to threshold values set by healthcare professionals. After receiving the data, the results are displayed on a web page accessible for healthcare professionals. The user of the sensor also has access to a an interface that displays their measured values and motivational notes from the healthcare professionals.

The finished system is a prototype of the described system, designed to be stable and functional to a degree that makes it suitable for testing in real environments. It has high focus on modifiability, making it adaptive and possible to change according to future needs. Providing healthcare professionals with vital signs of their users, this system can make their work more manageable by examining users' current situation faster, limiting home visits, and improving communication with users.

Preface

This report is written for Trondheim Kommune (TK) and is a project in the Norwegian University of Science and Technology (NTNU) course Customer Driven Project. The team has worked in cooperation with representatives from TK and NTNU. TK was represented by three people: Lise Høiberg, the project leader in the program for welfare technology at TK; Tone Dypaune, Professional development nurse; and Kirsti Fossland Brørs, project leader at TK. Jon Atle Gulla was appointed by the Department of Computer and Information Science as the team's advisor.

The project is motivated by the growing need for personnel in the healthcare sector in Norway. The senior citizen population of the country is steadily growing, but the capacity to handle the growth is lacking. The government has put forward multiple guidelines in order to help the situation, exploring the applications of information technology is among these. By incorporating technological solutions one might relieve the workload for healthcare professionals and enable them to help more people.

By using information technology and personal devices, users of the healthcare service might be empowered to take more control over and understand their health situation better. This in turn might help healthcare professionals give better assistance to the users of the healthcare system.

.....	.....
Haaland, Torgeir	Hovind, David
.....	.....
Jordal, Iver	Le, Tan Quach
.....	.....
Sandve, Sigurd	Solheim, Martin

.....

Vassli, Lars Tore

Contents

List of Figures

List of Tables

Part I

Planning and Requirements

Chapter 1

Introduction

1.1 General

This chapter will briefly introduce the project, its background and purpose. It also covers the involving parties and stakeholders of the project.

1.2 Project Mandate

The project was defined by TK in the course compendium. Below is the project description, unedited as it appeared in the compendium:

Using Wearable Devices in Welfare Services

Norway is investing in advanced welfare technologies as a tool to improve the quality and scalability of welfare services for seniors. Low maintenance cost for welfare technology is a decisive success factor. Low maintenance cost can be achieved by using standard, off-the-shelf components. In this task we will use off-the-shelf wearable devices to support common welfare services. Example devices are Fitbit and various smart watches.

The project will consist of the following tasks:

- Conduct a study of existing off-the-shelf devices, their capabilities and their APIs and tools for integration with open platforms.

- Create a set of scenarios for future services using these devices. The scenarios will include all stakeholders common in welfare services, such as seniors and municipality personnel. One of these scenarios will be selected for implementation.

- Design and implement one example service. The implementation will be end-to-end and will simulate a real service provided by the municipality.

The project will be run in close cooperation with user groups. The requirements will come from the customer. The development process will be agile and lean, with weekly or bi-weekly iterations where the project group will demonstrate new functionality. Tools such as paper prototyping, and low and high fidelity prototypes will be used. The project will produce open source code [22].

After initial meetings with the customer, it became clear that the main focus of this project was the creation and implementation of one example service. The team was to conduct a preliminary study and then implement an end-to-end prototype. What this service provided or how it functioned was left to the team to decide. TK specified that the service had to target senior citizens, and wanted the team to figure out what new possibilities wearable sensor devices on the market might provide them with, hopefully innovating the way TK uses technology today.

1.3 The Customer

The customer for this project is Trondheim Kommune (the municipality of Trondheim). Their responsibilities include health and welfare services for the whole municipality area. This project is part of TK's project called Welfare Technology [23].

1.4 Involved Parties

There are three main parties involved in this project: the customer, the developers, and the advisor. The customer, which is described in section 1.3, was represented by Lise Høiberg, Tone Dypaune, and Kirsti Fossland Brørs. See table 2.4 for more information.

The development team consisted of six computer science students and one informatics student, all master level students from the Department of Computer and Information Science at the NTNU.See table 2.5 for a list of developers.

The advisor was Professor Dr. Jon Atle Gulla at the Department of Computer and Information Science at NTNU. He provided help and feedback

to the development team during the project. For contact information, see table 2.6.

1.5 Project Background

Norway's population consists of a growing portion of elderly people. This leads to numerous challenges for the healthcare sector. The capacity for giving healthcare to elderly people is strained, both in institutions and for home services. As a response to this, the government has set several national goals aimed at addressing these problems. There is a strong focus on investigating the potentials of using technology in different aspects of the healthcare sector.

This is the reason TK's healthcare sector is investing in new technology.[57] TK has several projects concerning technology and healthcare services currently in process. One of these are a programme examining the use of fall sensors, and how to detect a fall. They are constructing a "smart-house" to show off future technologies that they want to use. There is also a development project working on moving the existing safety alarm system from analogue telephone system to using the internet.

In later years the market for wearable health tracking devices has grown considerably. The cost of the devices has decreased, and they are easy to acquire. Most sensors are accompanied by some kind of software on the web or on smart phones. This allows the users of the sensor to track and monitor their vital signs constantly. This can help them get a better understanding of their health and motivate them to keep a healthy lifestyle. The use of sensors for tracking vital signs is growing in popularity, one can expect to see more and more sensors with the ability to monitor an increasing amount of parameters.

1.6 Project Objective

The objective of this project is to look at the applications for commercial health trackers in the healthcare sector. The possibilities for using this technology preventively in order to enable elderly people to live at home longer will also be explored. At the end of the project a working prototype of the system will be presented.

1.7 Duration

The project ran over the course of 13 weeks from the 28th of August 2014 to the 20th of November 2014. The period consisted of 85 days, where 60 of them were weekdays.

1.8 Stakeholders

Table 1.1 describes the identified stakeholders in the project.

Stakeholders	Description and interests
	Constructive stakeholders
User	The people using wearable devices to be monitored by the system. These are the main stakeholders in the system. The term "user" does not include other users of the system, like healthcare professionals or system administrators. When referring to all users of the system, the term system users will be used. A user should find the system useful, be comfortable using the wearable device and feel more safe. Also, if the user has been granted access to see the data that is recorded, this should be easily accessible and intuitive to use.
Healthcare professionals	Healthcare professionals who can use the system to monitor data from the users, and get alarms if abnormal values are registered. Healthcare professional will be abbreviated to Healthcare Professional (HP) in some tables, in order to save space. They want the system to be efficient and easy to use, as this will make their job easier to do, and will result in them being able to help more people.
Next-of-kin	A close relative of the user. A next-of-kin will not have a direct role in the system, but might have an important role as a support person for the user. Next of kin will therefore have an important role, not in the system per se, but in the environment where the system is implemented. They may access the system through the user's account.

Stakeholders	Description and interests
Developers	The team members developing the system, whose responsibilities includes planning, designing and implementation. They are interested in creating a system that fulfils the requirements defined by the customer, and the other stakeholders are happy with.
Testers	Representatives of healthcare professionals and potential users who perform validation and verification of the system. They will test either parts of, or the entire system, to ensure that the system is functioning correctly, and that the needs of the stakeholder they represent are met.
Authorities	National and legal interests. The authorities make and regulate rules that may apply to the system. Authorities may be national or local government or the Directorate of Health.
	Negative stakeholders
Misusers	People who would misuse information or sabotage the system. Misusers might be hackers or employees without access clearance to the system.

Table 1.1: System stakeholders.

Chapter 2

Planning

2.1 General

In this chapter the planning of the project will be discussed. This chapter also describes the organisation of the team, quality assurance, and risk management.

2.2 Project Plan

2.2.1 Measurement of Project Effects

By using this system, healthcare professionals will easily be able to see which users have got abnormal values from their sensors, and can quickly send help if the user is having a medical emergency.

The user benefits from the system because by having a system that monitors their vital signs, abnormal values will quickly be detected and alert healthcare professionals. This in turn will help users to stay healthy by receiving correct assistance in time. The final goal is that the system enables users to live outside of healthcare institutions longer and limit the amount of home visits from healthcare personnel.

2.2.2 Limitations

2.2.2.1 Limitations by the Customer

The product must be open source, and can not be a mobile application, as this was not desired by the customer for maintenance reasons. The hub must use Health Level 7 (HL7) to communicate with server, and the sensor must

use Bluetooth to communicate with the hub. Integration with the customer's current system is not included in the scope of this project, because this is a time consuming process that exceeds the timeframe of this project. Data produced by the sensor could only be stored by the customer and the user, meaning no third party could process the data produced.

2.2.2.2 Limitations by the Project/Team

Time is a big limitation, the team only has 13 weeks to finish the project. The project has a limited budget, the customer has said that they are willing to spend about 3000 NOK on the project. Inexperience is also a big limitation, the team consists of seven students, where most members have little experience with developing web solutions and running a Scrum project. Only one of the team members has extensive experience with the web technologies that have been chosen for the project. For the inexperienced members this means that a lot of time must be spent learning how to use the chosen technologies.

2.2.3 Tool Selection

2.2.3.1 Git and GitHub

Git [24] is a version control system, and GitHub [25] is a hosting service for git repositories. Git gives full access to all the functionality even when using a free account, and is well documented and supported. The team also had experience using Git and GitHub which is why it was preferred over other source code management systems.

2.2.3.2 Google Drive

Google Drive [26] is a file storage and synchronization service provided by Google. It includes an office suite with the applications Google Docs, Sheets and Slides.

Google Docs is a free, web-based word processor, allowing multiple users to edit a document simultaneously and can easily be shared with other users. The team used Google Docs to collaborate on document drafts.

The file storage functionality in Google Drive was used to share other files (not source code files) between the group members.

2.2.3.3 Google Calendar

Google Calendar [27] is a free time-management web application offered by Google. It allows users to create a calendar that can be shared between multiple accounts, and it synchronizes events between all devices connected to this calendar. The team used this for time-management for meetings and work sessions.

2.2.3.4 Trello

Trello [28] is a free web-based project management application. It uses boards (that represents projects), which contain lists (corresponding to task lists, e.g. "To do", "Doing", "Done"). Within a list there are cards (tasks) that are meant to progress from one list to the next. Different members can be assigned to these cards. The team chose this tool because it is easy to use and some of the team members have used it before. Trello also makes it possible to maintain a digital Scrum board, this is an important aid for the Scrum process.

2.2.3.5 LaTeX

LaTeX is a document preparation system and document mark-up language. It was chosen because it is suitable for longer scientific documents and makes them look more professional. It allows the writers to focus on the content, not the look of the document.

The team will write drafts in Google Docs, and then write the final project report in LaTeX.

2.2.3.6 Facebook

Facebook [29] is an online social networking service. The team used Facebook as a communication tool for sending messages between the team members or writing posts for all team members to see.

2.2.3.7 Doodle

Doodle [30] is an Internet calendar tool for time management, and coordinating meetings. One of the team members would create a new "doodle" that spans one or more days with different time slots. He would then send the link to the other team members, and everyone would select the time slots they are able to attend a meeting. When everyone have answered, the time slot with most votes would be selected.

2.2.4 Schedule of Results

The milestones and sprints are listed in tables 2.1 and 2.2.

Milestones	
28. August	Project start
17. October	Pre-delivery of project report
20. November	Final delivery of project report
20. November	Presentation and project demo

Table 2.1: Project milestones.

Sprints	
Sprint 1	9th of September - 26th of September
Sprint 2	29th of September - 17th of October
Sprint 3	20th of October - 14th of November

Table 2.2: Project sprints.

2.2.5 Concrete Project Work Plan

The Gantt diagram in figure 2.1 below shows the project timeline.

2.3 Project Organization

2.3.1 Project Organization

Figure 2.2 illustrates the project organization. The project roles are described in table 2.3.

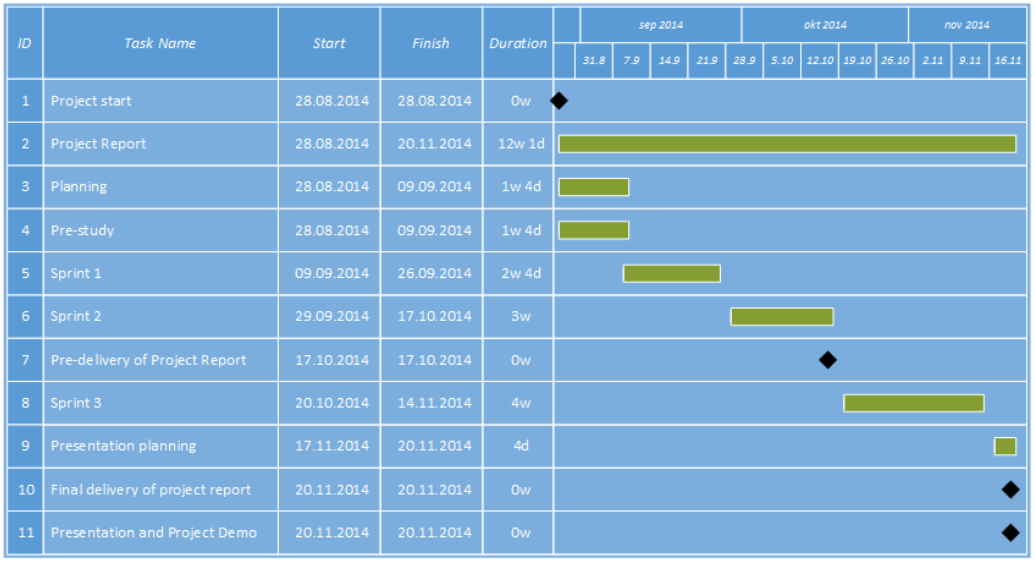

Figure 2.1: Project timeline Gantt diagram.

Role	Responsible	Responsibilities
Project manager	Martin Solheim	Responsible for managing the team and making sure everyone does what they are supposed to.
Customer contact	Martin Solheim	Responsible for communication between the team and the customer, and arranges meetings.
Advisor contact	Martin Solheim	Responsible for communication between the team and the advisor, and arranges meetings.
Lead front-end developer	Iver Jordal	Responsible for managing the front-end development team and for making sure the front-end system is working and following code conventions.
Front-end developer	Martin Solheim, Lars Tore Vassli	Responsible for developing the front-end of the system.
Lead hub developer	Torgeir Haaland	Responsible for managing the hub development team and for making sure the hub and sensor system is working.

Role	Responsible	Responsibilities
Hub developer	David Hovind	Responsible for developing the hub and sensor system.
Back-end programmer	Tan Quach Le, Sigurd Sandve	Responsible for developing the back-end of the system.
Lead tester	Sigurd Sandve	Responsible for testing and making sure that the system is thoroughly tested.
Documentation responsible	David Hovind, Lars Tore Vassli	Responsible for the quality of the documentation and for making sure the documentation is being written.
Secretary	Iver Jordal	Responsible for taking notes during meetings with the advisor, the customer and internal meetings.
Document structuring	Tan Quach Le	Responsible for getting a good structure of the project report.
Document grammar	Iver Jordal	Responsible for eliminating grammar errors in the project report.
User manual responsible	Lars Tore Vassli	Responsible for creating a user manual for the finished product.

Table 2.3: Project roles.

2.3.2 Partners

Name	Role	E-mail
Lise Høiberg	Project leader for the program for welfare technology at TK	lise.hoiberg@trondheim.kommune.no
Tone Dypaune	Professional development nurse	tone.dypaune@trondheim.kommune.no

Name	Role	E-mail
Kirsti Fossland Brørs	Project leader at TK	kirsti-fossland.brors@trondheim.kommune.no

Table 2.4: Project customer representatives.

Name	E-mail
Torgeir Haaland	torgehaa@stud.ntnu.no
David Hovind	davidjh@stud.ntnu.no
Iver Jordal	iverjo@stud.ntnu.no
Tan Quach Le	tanql@stud.ntnu.no
Sigurd Sandve	sigurdsa@stud.ntnu.no
Martin Solheim	martsolh@stud.ntnu.no
Lars Tore Vassli	larstva@stud.ntnu.no

Table 2.5: Team members.

Name	E-mail
Jon Atle Gulla	jag@idi.ntnu.no

Table 2.6: Project advisor.

2.4 Quality Assurance

2.4.1 Routines for producing high quality internally

To ensure that only solid code is produced and that the main branches of the project's git repositories only contains quality code the team has made sure to keep a rigorous scheme of accepting pull requests. Whenever a change was to be made to the system a new branch would be created and the implementation of that change would happen in that branch. When the feature was done the developer made a pull request to make the branch a part of the main branch again. Then another team member had to review the pull request and make sure that the code was of sufficient quality to be merged

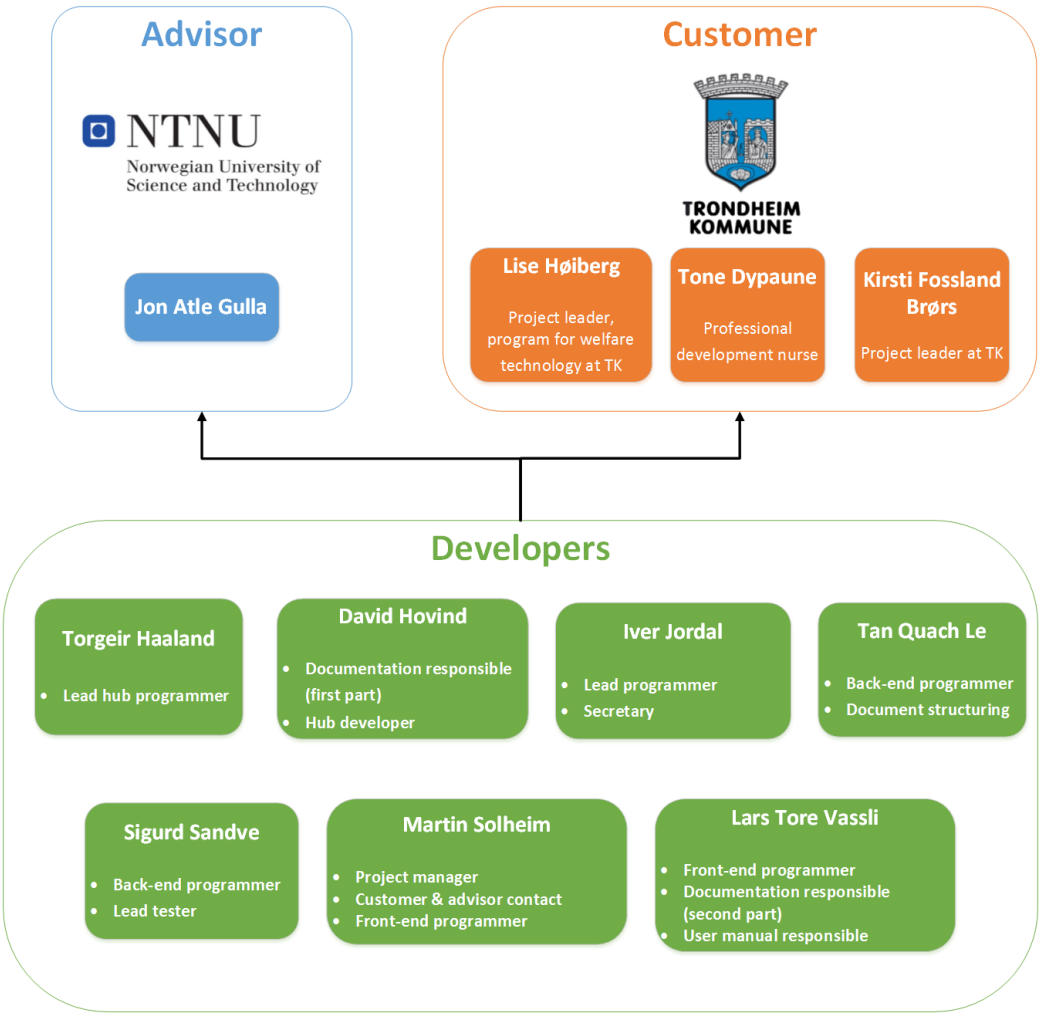

Figure 2.2: Project organization.

with the main branch.

The different development teams internally made sure to work together as often as possible, this let the team members exchange knowledge and let them do quality control of each others code.

2.4.2 Routines for Approval of Phase Documents

The results of each sprint would be presented to the customer so that they could give feedback on the work done so far. The customer had the opportunity to provide feedback on aspects such as requirements, misunderstandings, and suggest solutions to problems. The advisor would also be presented the progress and results from the sprints, so that he could give the team feedback on how the project was doing.

2.4.3 Procedures for Customer Meetings

Customer meetings were planned in advance over e-mail. The agenda for the meeting was written by the team before the meeting and sent to the customer one day in advance. All meetings were lead by the team leader according to the agenda.

Customer meetings have been the primary method for uncovering requirements and for getting feedback for the user driven development process of the web application.

2.4.4 Procedures for Advisor Meetings

Advisor meetings were set up as a regular weekly occurrence, each Tuesday at 10:15 in room 122, IT-bygget, unless otherwise stated. One day before each meeting the team worked out an agenda for the meeting and gathered documents for review by the advisor. These documents and the agenda were then sent by e-mail to the advisor.

2.4.5 Document Templates and Standards

At the start of the project, the team developed standards for the agenda document, the weekly status report, time sheet, and for the report structure. See Appendix D for figures showing the templates.

2.4.6 Coding Routines and Standards

2.4.6.1 Front-end

- JavaScript variables were written in camel case
- Soft line length limit: 120 characters
- Indenting: Two spaces
- JavaScript Code Quality Tool: JSLint

When setting up the project, the team structured the files in a logical way, rather than having everything in one folder, as shown in table 2.7

Location	Description
.	In the root folder, there is one HyperText Markup Language (HTML) file for each Single-page Application (SPA) - (login, dashboard and user-dashboard) and app icons
|-css	Style sheets that the team has written
|-img	background, logo, animated gifs
|-js	General JavaScript files, like static.js and common-config.js
|---controllers	Angular controller modules
|---directives	Angular directive modules
|---filters	Angular filter modules
|---services	Angular service modules
|---lib	A directory for libs that are included. Contains a subdirectory for each lib.
|-snd	Sounds, like the alarm notification sound
|-templates	HTML templates that contains Angular-specific elements
|---modals	All modal templates go in this subdirectory

Table 2.7: Folder structure

2.4.6.2 Back-end and Hub

Coding in the back-end and the hub was done following the Python Enhancement Proposal 8 (PEP8) standard, which is the de-facto code style for Python [45]. This standard covers code layout, naming convention, and commenting. There was one thing the team modified in PEP8; instead of using the standard 80 character line width the team chose to use a 100.

2.4.7 Version Control Procedures

For assuring code compatibility, co-operability and safety, Git version control system was utilized. All of the code was located on three different git repositories, one for front-end, one for back-end and one for the hub development. All team members were responsible for committing their work often and making sure not to break the current build. By doing it this way all team members could stay up to date with the latest version of the software.

2.4.8 Internal Reports

The team used Trello (see section 2.2.3.4 for more information) to keep track of tasks and their status, and assign them to team members. This is a way of documenting the progress internally.

The team also used Google Drive (see section 2.2.3.2) to write minutes from internal meetings and effort registration.

2.5 Risk Management

In this section the risks of the project will be presented. The risks were rated for consequence and probability. Consequence levels are explained as followed:

High (H) Will affect the progression of the project greatly.

Medium (M) Will affect the progression of the project moderately.

Low (L) Will affect the progression of the project slightly.

The rating for probability of the risk occurring, is defined by:

High (H) Probability of risk occurring is greater than 0.8.

Medium (M) Probability of risk occurring is between 0.3 and 0.8.

Low (L) Probability of risk occurring is less 0.3.

A mitigation plan was developed for risks with a consequence level of M or higher. The risk assessments can be found in table 2.8.

Group dynamics	
Risk ID	R1
Risk Factor	Conflict between team members.
Probability	M
Consequence	M: Bad morale, which could lead to a poor project or incomplete project
Mitigation plan	If the conflict is related to the project it can be used as a starting point for discussion within the team where we try to come to a solution. If the conflict is off-topic it should be resolved quickly, involving the advisor if necessary.
Risk ID	R2
Risk Factor	Team members fail to complete assigned task to an acceptable standard.
Probability	L
Consequence	M: Bad grade, or incomplete project due to parts not functioning.
Mitigation plan	The person responsible should try to improve the work, if unable (for example due to illness or lack of understanding of the assigned work) the project lead will be responsible for the completion of the task.
Risk ID	R3
Risk Factor	Other commitments or coursework interfere with the project.
Probability	M
Consequence	H: Incomplete project, or a lot of catching up needed at the end of the project.

Mitigation plan	Team members will be responsible for catching up with assigned work. It will be possible to allocate tasks to team members in a manner that minimises conflicts with other commitments.

Risk ID	R4
Risk Factor	Failure by team members to attend meeting.
Probability	M
Consequence	L

Risk ID	R5
Risk Factor	Illness/Absence
Probability	L
Consequence	L/M/H: Increased workload for rest of team, not enough time to complete project.
Mitigation plan	Consequence of this risk depends on amount of hours lost due to illness/absence. We can mitigate impact if we work in pairs, so that we always have two people with knowledge of an aspect of the project.

Technology choices	
Risk ID	R6
Risk Factor	The Angel sensor does not arrive in time.
Probability	H
Consequence	H: We will not be able to complete the project as expected.
Mitigation plan	A replacement sensor will be acquired in order to proceed with development without the Angel sensor.

Risk ID	R7
Risk Factor	When the Angel sensor arrives, the implementation proves to be incompatible with the sensor.

Probability	H
Consequence	M: We will not be able to complete the project as expected.
Mitigation plan	The Bluetooth communication between sensor and hub has to be done in a highly modifiable manner. Thus enabling a swift adaptation to communication with other Bluetooth LE devices. Furthermore, steps taken for R6 applies to this risk as well.

Risk ID	R8
Risk Factor	Failure by team mate to learn the chosen technologies to a high enough level.
Probability	L
Consequence	M: The implementation process might take more time.
Mitigation plan	By making sure that all team mates work through some tutorials and by having work meetings, team mates should be able to learn the chosen technologies.

Communication	
Risk ID	R9
Risk Factor	Failure to communicate efficiently with the customer.
Probability	H
Consequence	H: Project will be of little value to the customer.
Mitigation plan	We have to constantly follow up the customer, making sure that we always understand them correctly and that they understand us. This is achieved by frequent meetings in person and close contact via email.

Risk ID	R10
Risk Factor	Failure to communicate efficiently with the advisor
Probability	L
Consequence	H: Project will not be completed in a satisfactory way.
Mitigation plan	By arranging for frequent advisor meetings the team should be able to maintain close contact with the advisor.

Risk ID	R11
Risk Factor	Failure to communicate efficiently within the team
Probability	M
Consequence	H: Waste time, not completing on time.
Mitigation plan	By arranging for frequent meetings, keeping contact on Facebook and using tools like Trello and GitHub, the team should be able to communicate efficiently.

Table 2.8: Project risks.

Chapter 3

Preliminary Study

3.1 General

This chapter presents the preliminary study for this project. It discusses the different options of sensors, hubs and technologies considered in this project. Further, this chapter explains the reasoning behind the choices made and the licensing of the final software.

3.2 Similar Solutions

3.2.1 Overview

The customer wanted a tailored interface where healthcare professionals could monitor the vital signs of multiple users, using wearable sensors that transmit data to a server. Complete solutions like this are not open or freely available, but parts of the solution do exist on the market. Most systems are local and closed for third parties to make changes. In this regard it was more feasible to look at individual sensors (see section 3.4). However there exist some comprehensive solutions on the market already doing some of what the customer wanted, the most comprehensive being the e-health sensor platform [2] and iHealth [4].

One does not need to buy into a extensive ecosystem of sensors in order to have access to some basic health tracking features. Everyone who owns a modern smart phone has access to features like activity monitoring and heart rate. By using the camera and flash on the phone some applications can calculate a persons heart rate. Some phones even include a heart rate sensor [3]. Phones also have a huge capability for presenting and analysing data. Apple's newest iteration of iOS8, includes an application called Health [1]. This

application uses Apples HealthKit to gather information from applications that monitor vital signs and share that information with other applications that might be interested in this information. Other mobile operating system providers are also starting to include this kind of functionality. Of course the problem remains, this technology is not open to the extent that is required in order to be used by the municipality.

3.2.2 e-Health Sensor Platform

The e-health sensor platform is a platform with ten different sensors, including body temperature, heart rate, blood pressure and blood-oxygen saturation. The platform uses a Raspberry Pi, or Arduino to capture all the data from the sensors and can transmit the data to the cloud where the data can be processed, or to a computer or mobile device to see the data in real time. The main drawback with this platform is that it has a lot of different sensors that are connected via cables, reducing portability.

3.2.3 iHealth

The iHealth product family is an extensive collection of sensors that can monitor heart rate, blood sugar, blood oxygen, blood pressure and weight. The most notable aspect with this solution is the software that pairs with the different sensors. The data recorded from the different devices can be analysed both on mobile platforms and on computers. The drawback with this solution is that it is proprietary and the data is only available through iHealth's own applications. This means that one gets locked into the ecosystem.

3.3 Relevant Sensors

Relevant, off-the-shelf solutions that could be used for the purpose of this project were sought after by the customer. The team studied a whole range of sensors, with primary focus on wristbands, where the main criteria included health measurements, battery life, the price, and whether it had an open API or not. Table 3.1 describes the different sensors that were studied, and their differences.

Name	Activity	Heart rate	Blood oxygen	Battery life	Wireless	Open API	Cost
Angel	Yes	Yes	Yes	One week	Yes	Yes	NOK 1040
Basis Band	Yes	Yes	No	Four days	No	No	NOK 970
Jawbone Up24	Yes	No	No	One week	Yes	Yes	NOK 1100
Garmin Vivofit	Yes	No	No	One year	Yes	No	NOK 1100
Fitbit Flex	Yes	No	No	Five days	Yes	No	NOK 650
Nike+ Fuelband	Yes	No	No	-	Yes	No	NOK 1760
Fitbit one	Yes	No	No	One week	Yes	No	NOK 730
Fitbit zip	Yes	No	No	Six months	No	No	NOK 470
Jawbone Up	Yes	No	No	One week	Yes	No	NOK 600
Nike+ fuelband SE	Yes	No	No	Four days	Yes	No	NOK 1500
HxM BT	Yes	Yes	No	24 hours	Yes	Yes	NOK 490
iHealth Wireless Activity and Sleep Tracker	Yes	No	No	-	Yes	No	NOK 390
iHealth Wireless Pulse Oximeter	No	Yes	Yes	-	Yes	No	NOK 450
Withings O_2 Pulse	Yes	Yes	Yes	Two weeks	Yes	Yes	NOK 1000

Table 3.1: Comparison of sensors.

The prices in table 3.1 are mainly taken from Norwegian web stores. Except for some of the products, whose prices are taken from American web stores, priced in USD. The prices in USD were converted to NOK using the current exchange rate, NOK 6.47 to USD 1, and rounded. Tax is not included in the American prices, and shipping is not included in any price.

The main problem with most of these devices is that they do not have an open API, meaning that custom software could not be developed for them. Some of the sensors however did have this feature and were prime candidates to be used in this project. The following sections go into more detail about these sensors.

3.3.1 Angel Sensor

The Angel sensor is a flexible wristband with four sensors for monitoring vital signs. It measures heart rate, blood-oxygen saturation, activity and body temperature. It has an open API that gives access to reading the values directly from the sensor, which means that it is very easy to create custom applications for it, and the developer has complete control of the data [5]. The sensor's biggest drawback is that it is a product of a crowd sourcing project and not yet in production. The company behind the sensor has been able to ship some prototype sensors, and the first batch of finished products is set to be shipped 20th of October, 2014.

3.3.2 Jawbone UP24

Jawbone up 24 is an activity monitor wristband. It is primarily made to make people healthier, by monitoring activity, diet and sleep. Jawbone focuses primarily on applications on a mobile device and most of the strengths of the Jawbone up lies in its software [6]. It has an open API so that it is possible to develop applications and access the information of the sensor. The drawback of this solution is that Jawbone saves the data on a third party server and the data must be accessed through it. It is not possible to get the readings directly from the sensor itself. This means that a mobile device has to be paired to the Jawbone up in order for it to send data to their servers, to be retrieved by a custom application later [7].

3.3.3 Withings O_2 Pulse

The Withings O_2 Pulse is an activity monitor that measures activity, heart rate and blood-oxygen saturation. It focuses on versatility, where it is possible to use it as a wristband, a belt clip, or just be worn in the pocket [8].

Much like the jawbone, it features an API that is open, but the data must first go through their servers before it can be accessed by third parties [9].

3.4 Hub for Receiving Sensor Data

The hub is the device that receives data from the sensor and sends it forward to a centralized server such that healthcare personnel can access the data on multiple platforms. The choice of hub is important for the development process and has to be carefully considered. The choice of hub was narrowed down to two possible devices; a mobile device and a mini computer. The solution to this question is somewhat tied to the choice of sensor, as some sensors require an standard application run on a mobile phone to access the data.

3.4.1 Mobile Device as Hub

A mobile device is a smart phone or tablet that can receive the data from the sensor. Here are some of the pros and cons of using a mobile device as a hub.

Advantages:

- Only one device is needed, it can function as both a hub and to present information. The mobile device could have an application that could show the health information of the sensor, as well as sending the data forward to a server.
- If the user has their own mobile device, this could be used as a hub and no costs for an extra device is necessary.
- By using a mobile device the hub is mobile and can send data from anywhere.

Drawbacks:

- There are multiple possible platforms. To cover most of the mobile devices on the market today there would have to be an application for both iOS and Android. This would mean double the work in the development of the hub, and more work in maintenance of the applications.
- Mobile devices are closed platforms, which means that the application which will be developed would only run on that platform.

- The battery life of mobile devices is usually poor and unreliable, and if the hub loses battery life there is no connection between the sensor and the server.

- There are users who do not own a mobile device suitable to be a hub. In this case a new device must be provided to the user, adding extra cost.

3.4.2 Mini Computer as Hub

A mini computer is a small single-board computer, which is a complete computer built on a one circuit board, the most famous mini computer is the Raspberry Pi. It's a credit-card sized computer which has had great success in the mini computer market [10]. Here are some of the pros and cons of using a Raspberry Pi as a hub.

Advantages:

- The Raspberry Pi is an open platform. The code written for this computer can essentially be used on any computer.

- There is no battery for the Raspberry Pi so it does not need to be charged regularly to function.

Drawbacks:

- The cost of the Raspberry Pi is around NOK 550 using cabled network, if the hub needs wireless networking the price adds up to about NOK 650.

- The Raspberry Pi has to have a stationary position in the home, whereas a mobile platform does not.

3.5 Software development-methodology

The software development methodology is important for how a project's development process is going to preform. Two technologies will be discussed in this section, the Scrum model and the Waterfall model. The strengths and weaknesses of these methods will be discussed and compared, which will form a base for the choice made in this project.

3.5.1 Scrum

Scrum is an agile software development model. When using the Scrum method one has to define the product backlogs, and the sprints. A sprint is a specific period of time, usually between one to four weeks, where the team works on the chosen tasks. When a sprint is finished, new tasks for the next sprint is chosen. Scrum's strength is the ability to adapt to new changes. This makes it easier for the customer to be more involved, and it is easier for the development team when the customer changes or adds requirements. However, this also means that it can be harder to meet the deadline, if the customer keeps changing or adding new requirements.

3.5.2 Waterfall

The waterfall model is a sequential software development model and is often considered to be the classical approach to software development. The development process flows like a waterfall through different phases of the project's life cycle, hence the name waterfall. The phases of the waterfall model are requirements, design, implementation, verification and maintenance. The main intention is to move from one phase to the next in a purely sequential manner, such that one phase cannot start until the previous phase is completed, reviewed and approved. [11]

3.6 Programming Languages

3.6.1 Python

Python is an object-oriented programming language with dynamic typing. Its high level built-in data structures and dynamic typing makes programs small and relatively fast to develop compared to more strict languages with more structure and syntax. Python supports modules and packages which encourages modularity and reuse of code. [17]

3.6.2 JavaScript

JavaScript is the programming language of the web. It is the language used to program the behaviour of a web page. For example, with JavaScript, it is possible to fetch data from an API asynchronously, create draggable components and alter the document content when a component is clicked. All modern browsers understand this language.

3.6.3 Java

Java is an object-oriented programming language. Its main advantage is that applications created with Java are cross-platform compatible, meaning that it does not need to be recompiled to run on a different platform ("Write once, run everywhere").

3.6.4 Ruby

Ruby is a dynamic, open source programming language with a focus on simplicity and productivity.

3.7 Technologies

3.7.1 Hub and sensor technologies

Continua Health Alliance is a non-profit organization whose purpose is to agree on international standards within welfare technology. Bluetooth, ZigBee or Universal Serial Bus (USB) is strongly recommended by Continua to be used as communication between a sensor and hub. The message standard HL7 is recommended as a communication protocol towards central servers. The recommendations from Continua together with the availability will form a base for the choice of technologies for the hub and sensor.

3.7.1.1 Bluetooth

Bluetooth is a wireless technology standard for exchanging data over short distances. The Bluetooth technology is built in to billions of devices, from mobile phones to medical equipment. [18]

3.7.1.2 ZigBee

ZigBee is a standards-based wireless technology focusing on low-cost and low-power control networks. It is the preferred short-distance technology defined by Continua and is easily extendible. There is no need for configuration when adding a device to the network. [19]

3.7.1.3 USB

USB is a widespread industry-standard external bus used to connect computers and devices. Unlike the other alternatives, ZigBee and Bluetooth, USB is not a wireless technology. [20]

3.7.1.4 HL7

HL7, or Health Level 7, is an organization whose primary work is related to the exchange of healthcare information. Version 2 is the most widespread of the message standards and comprise a set of standards which define the content to be sent, the message syntax to be used and the lower level protocol which is how the exchange of the message should be. Although recommended by Continua, the standard is not used in Norway at this time.[21]

3.7.2 Front-end Technologies

3.7.2.1 Proposed Technologies

Technology	Uniform Resource Identifier (URL)
jQuery mobile	`http://demos.jquerymobile.com/1.4.3/`
AngularJS	`https://angularjs.org/`
Angular UI Bootstrap	`http://angular-ui.github.io/bootstrap/`
Bootstrap	`http://getbootstrap.com`
Ionic Framework	`http://ionicframework.com/`
Intel's App Framework	`http://app-framework-software.intel.com/`
Lungo	`http://lungo.tapquo.com/`
Gumby Framework	`http://gumbyframework.com/`
Angular Strap	`http://mgcrea.github.io/angular-strap/`
Golden Layout	`https://golden-layout.com/`
DevExtreme	`http://js.devexpress.com/`
Telerik	`http://www.telerik.com/kendo-ui`
Ratchet	`http://goratchet.com/`
EmberJS	`http://emberjs.com/`
Foundation	`http://foundation.zurb.com/`
Frameless Grid	`http://framelessgrid.com/`

Flat UI	`http://designmodo.github.io/Flat_UI/#`
Mobile Angular UI	`http://mobileangularui.com`
HTML	`http://www.w3.org/standards/techs/html#w3c_all`
Cascading Style Sheets (CSS)	`http://www.w3.org/standards/techs/css#w3c_all`

Table 3.2: Proposed front-end technologies.

3.7.2.2 Single-Page Application

A SPA is a web application that fits on a single web page with the goal of providing a more fluid user experience akin to a desktop application. Although the SPA provides the perception and navigability of separate logical pages in the application, it's essentially just one page containing logic for the whole system. Rather than reloading a new page for each action, client side technology is used to manipulate the Document Object Model (DOM)ṫhe client side fetches data from the API using Cross-Origin Resource Sharing (CORS) when needed. This kind of dynamic communication with the web server happens behind the scenes.

The main benefits of a SPA are the following:

- It reduces round trip delays of loading complete web pages, so we get a faster user interface. This enhances the User Experience (UX).
- It is easier to implement complex state logic, because small view state changes do not map well to URL.
- It is good for responsive web design.

3.7.2.3 HTML

HTML is the standard mark-up language used to create web pages. Since the front-end is developed as a modern SPA, the front-end developers are required to write HTML5, which is the latest widely adopted revision of the HTML standard. HTML5 is specified by the HTML Working Group of the World Wide Web Consortium.

3.7.2.4 CSS

CSS, or Cascading Style Sheets, are used to define the style and appearance of the HTML pages.

3.7.2.5 AngularJS

"AngularJS is what HTML would have been, had it been designed for building web-applications"
— angularjs.org

Since HTML was not designed for dynamic views, one needs a framework such as AngularJS to make it easy to develop and maintain a SPA. AngularJS is a JavaScript framework that fixes the shortcomings of HTML when it comes to developing a SPA. AngularJS provides an imperative way for manipulating the DOM, so that it becomes easy to create dynamic views. It also contains a set of tools that you typically need in a SPA. AngularJS is fully extensible and works well with other libraries, such as Highcharts.

3.7.2.6 Highcharts

Highcharts is a JavaScript library used to create interactive and dynamic charts for web applications.

3.7.2.7 Bootstrap

"Bootstrap, a sleek, intuitive, and powerful mobile first front-end framework for faster and easier web development."
— getbootstrap.com

Bootstrap is the most popular front-end framework for developing responsive web sites. CSS media queries are used to scale the web site so that it fits perfectly on a wide array of devices. It is well tested and works in all modern web browsers. Bootstrap is also free and open source.

3.7.2.8 Mobile Angular UI

Mobile Angular UI is a Framework that utilizes AngularJS and Bootstrap in order to make responsive web apps, with especially high focus on mobile devices.

3.7.2.9 Flat UI

Flat UI is a user interface kit based on Bootstrap. There is both a free and a paid version. The free version contains a very limited amount of elements.

3.7.2.10 Frameless Grid

Frameless Grid is a resource to assist the design process of making a web page.

3.7.2.11 Foundation

Foundation is a responsive front-end framework. The framework has a strong focus on user experience.

3.7.2.12 EmberJS

EmberJS is a framework, similarly to Angular, made for creating web applications. It is designed to simplify the development process and make the job of making a web application faster and more efficient.

3.7.2.13 Ratchet

Ratchet is a framework focused on making web applications for mobile devices.

3.7.2.14 Telerik

Telerik has a User Interface (UI) framework called Kendo UI. Kendo UI is intended to make it easy to create responsive web sites and apps using HTML5 and JavaScript (JS). The problem with Kendo UI is that there is no free version of it.

3.7.2.15 DevExtreme

DevExtreme is a cross-platform HTML5/JS tool aimed at developing for web and mobile devices. DevExtreme has no free version, only a free trial.

3.7.2.16 Golden Layout

Golden Layout is a tool for organizing windows and tabs for JavaScript applications. Does not have full functionality with touch devices, but its features is only needed for managing windows on a big screen or between multiple

screens. Golden Layout is licensed under the Creative Commons [32] license and is therefore free to use for non commercial projects.

3.7.2.17 AngularStrap

AngularStrap is a set of native directives that integrates Bootstrap with AngularJS.

3.7.2.18 Gumby Framework

The Gumby Framework

3.7.2.19 Lungo

Lungo is a JavaScript library for developing HTML5 hybrid mobile apps.

3.7.2.20 Intel App Framework

Intel's App Framework is a JavaScript library for developing HTML5 hybrid mobile apps.

3.7.2.21 Ionic Framework

Ionic Framework is a CSS framework and JavaScript library for developing HTML5 hybrid mobile apps.

3.7.2.22 AngularUI UI Bootstrap

Angular UI Bootstrap is a set of native directives that integrates Bootstrap with AngularJS written by the AngularUI Team. UI Bootstrap has more powerful functionality than AnguarStrap.

3.7.2.23 jQuery mobile

jQuery mobile is a JavaScript library for creating web applications for mobile devices, optimized for a wide variety of smartphones and tablets with touchscreen interfaces.

3.7.3 Back-end Technologies

3.7.3.1 Django

Django is a open source web application framework. It supports four database back-ends: PostgreSQL, MySQL, SQLite and Oracle. Django makes it easier to create database-driven websites.

3.7.3.2 Django Representational State Transfer (REST) Framework

Django REST Framework is a tool to make it easier to build Web APIs.

3.7.3.3 Cron Job

Cron is a software utility that handles time based job scheduling. The team set up a scheduler to handle deletion of old motivation texts, by using SetCronJob that calls the correct API endpoint[49].

3.8 IDE

In this section the Integrated Development Environment (IDE)s considered in development of the system are described.

3.8.1 PyCharm

PyCharm [37] is an IDE used for development using Python. It has good support for text editing, syntax highlighting, auto indentation, code navigation and completion, and automatic error checking. Some features are behind a pay wall but PyCharm also has a well functioning free community edition as well as free student accounts. PyCharm has support for many frameworks, including Django.

3.8.2 WebStorm

WebStorm [44] is an IDE for front-end development. Like PyCharm, you have to pay for an account in order to use WebStorm. Fortunately JetBrains, the company that makes WebStorm and PyCharm offers a free student account with access to, among others, WebStorm. The IDE offers an extensive set of features for developing web applications using technologies like Angular.js and Bootstrap.

3.9 Conclusion and Evaluation

This section provides an explanation, and a justification for the choices made based on the preliminary study.

3.9.1 Choice of Sensor

The study shows that existing wearable devices are usually locked to one system, and does not have an open API. Based on this study, the Angel sensor was chosen. The reason for this decision is that this wristband fulfils most of the requirements presented by the customer, see section 1.2. The price is reasonable compared to the alternatives - even devices with only movement sensor comes at about the same price. It is completely open, meaning that it does not require the data to go through a mobile phone or external server (as is the case with all the other sensors we have considered), and grants access to raw data if necessary. But it can also be used with an API to access data easier. The Angel sensor does not need to send the data to a third party to provide access to it, which is the case with other sensors, this is vital when it comes to privacy.

3.9.2 Choice of Hub

The mini computer, Raspberry Pi, was chosen. This is because it is an open platform and only one version of the hub software need to be developed, in contrast to a mobile app. Another reason to choose the Raspberry Pi is that the customer does not want a mobile app to maintain. The Raspberry Pi was set up with the Debian [53] based operating system Raspbian [54], which is free and optimized for the Raspberry Pi.

3.9.3 Choice of Software Development Methodology

Scrum was chosen as the development methodology for the project. The choice was made because of the flexibility that the method makes possible. The project has a short time span and requires a considerable amount of development, and these are conditions that Scrum is well suited for. Scrum also allows for an agile approach to new demands, making it less troublesome to accommodate new requirements. The methodology also makes the relation with the customer easier to manage. In this project communication with the customer is essential to develop a good solution, involving the customer in the project is made easy by using Scrum.

3.9.4 Choice of Programming Languages

The languages that have been chosen to implement the Angelika system is JavaScript for the front-end and Python for the back-end, hub, and communication between the hub and the sensor.

The choice to use JavaScript in the implementation of the front-end was motivated by the fact that the front-end web page will be a SPA. This means that the page will require a degree of flexibility and responsiveness that is most easily achieved by using JavaScript. In addition, all the group members were familiar with JavaScript, so this meant that we would not need any extra time to learn a new programming syntax.

The reasons for choosing Python is discussed in section 3.9.7.

3.9.5 Choice of Hub and Sensor Technologies

Bluetooth was chosen as communication method between the sensor and hub. This was largely influenced by the choice of sensor. Since very few off-the-shelf devices support ZigBee, it is either Bluetooth or USB that must be used if we wanted to stay within the recommendations of Continua (See section 3.7.1 on page 32). Bluetooth does have an advantage over USB due to it being a wireless protocol. In addition to this the Angel sensor supports Bluetooth 4.0, which is more power-friendly than previous versions. To meet the recommendations of Continua, HL7 was chosen as the message standard to use to exchange data between the hub and the central server.

3.9.6 Choice of Front-end Technologies

When using JavaScript to construct a web page or a web application, which is the case for this project, one should use a framework to make development less complicated. A good JavaScript framework unlocks the power of JavaScript and makes it easily accessible for the developer. The different libraries have different applications. The library that has been chosen for the construction of the Angelica web application is AngularJS, see section 3.7.2.5. This is a library developed by Google in order to make development of SPAs easy.

Reasoning behind the choice of the Angular library:

- Angular works well with a multitude of APIs. This includes REST framework, which have been chosen for our back-end.
- An important requirement from the customer is that the solution should be open. Angular lives up to this requirement, being licensed under the

MIT license [31].

- Angular has got a large user community, which makes it easy to find helpful information on the Internet, and there are many online learning resources [33, 34] on the web that are free and easily accessible.
- Having this amount of learning resources will enable the team to keep the development schedule, while at the same time learn a new technology.
- Testing JavaScript code can be quite complex, but the Angular framework makes testing easier.

The web page that the healthcare professionals will be using to view information about the users is going to be quite data intensive. There will be a great amount of data to display on the page. This requires an intelligent and intuitive layout of information. In order to facilitate this a library called Golden Layout [35] will be used. This library helps in the structuring of the page and makes displaying multiple data views less complicated.

3.9.7 Choice of Back-end Technologies

Django was chosen because it makes it easy to build database driven websites, and supports python which was a familiar programming language to the developers. Django REST framework provides an easy to use api to post and fetch data from the database. This functionality was very important to our system because the sensor produces a lot of data that needs to be stored, and the front-end fetches this data to present for the user. All this is made easy by using REST framework.

3.9.8 Choice of IDE

PyCharm and WebStorm was chosen as IDEs. Since python was chosen as the development language, the team needed a good IDE for development. PyCharm was chosen for this because of its quick code navigation, code completion, easy refactoring, unit testing and its debugger. Furthermore it also has good support for the Django framework. It is made by JetBrains which focuses on intelligent IDEs that makes coding easier and quicker. WebStorm is also made by JetBrains and was chosen for the front-end web development. Since they come from the same company, it is build on the same core, which makes the user interface and shortcuts very similar. This makes the development process smoother and makes it easier to switch between developing in

Python, HTML and JS. WebStorm was also chosen because it offers extensive support for frameworks like Angular.js and Bootstrap and the fact that it was free for students.

3.10 Software Licenses

3.10.1 Open Source

One important requirement from the customer was that the product of the project should be open source. Open source software is software that can be freely used, changed, and shared (in modified or unmodified form) by anyone. Open source software is made by many people, and distributed under licenses that comply with the Open Source Definition [22]. There are many kinds of open source licenses, two of these are mentioned in this section.

3.10.2 MIT License

The MIT license [31] is a free software license that grants the person who obtains a copy of the software the permission to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the software. The MIT license is very simple and uncomplicated.

3.10.3 Creative Commons

The Creative Commons license [32] is a free licence for creative work. The Creative Commons organization describes their mission as:

"*Creative Commons helps you share your knowledge and creativity with the world.*

Creative Commons develops, supports, and stewards legal and technical infrastructure that maximizes digital creativity, sharing, and innovation."

In order to use the Creative Commons licence one only has to choose one of the six versions of the licence and mark the work with the license. The six variations are:

Attribution This lets people use and share ones work as they want, as long as they attribute the work to the creator.

Attribution-ShareAlike This is the same as Attribution, only that people who wish to share the work has to use the same license.

Attribution-NonCommercial This is the same as Attribution, only that the work cannot be used for commercial purposes.

Attribution NoDerivs This lets people pass the work along and use it as they want, as long as they do not make any changes to the work and attribute the work to the creator.

Attribution-NonCommercial-ShareAlike This is the same as Attribution NoDerivs, only non commercial.

Attribution-NonCommercial-NoDerivs This is the most stern licence. It lets people download the work and share it, as long as they credit the creator, but nothing else.

3.10.4 Choice of license

The team wanted the software to be as open as possible, but the software produced is restricted to the most restrictive licence of the technologies chosen. Considering this, the license chosen was the Creative Commons Attribution-NonCommercial, because this is what Golden Layout uses, which is the most restrictive license used.

Chapter 4

Requirements

4.1 General

This chapter describes the requirements for the system, including initial and final requirements, user stories, and the complete product backlog.

4.2 Initial Requirements

This section shows the initial requirements, as agreed with the customer in the initial phase of the project in table 4.1. Based on these requirements the development team created a list of requirements to form a basis for the complete system. For a list of the final requirements, see table 4.2. How the requirements were changed during the development can be seen in section 4.4. All the requirements have a priority and a complexity, this is explained in section 4.3.3 and 4.3.4 respectively.

ID	Description
CR1	The system must include a hub, a sensor, a server, an interface for the healthcare personnel, and an interface for the users
CR2	The front-end interface must be a web application, both for the healthcare personnel and the users
CR3	The communication between hub and sensor should use Bluetooth, USB, ZigBee or Near Field Communication (NFC)
CR4	The communication between hub and server should use HL7
CR5	The system should show normal vales for the users

ID	Description
CR6	The system should alert healthcare professionals of abnormal values measured by a sensor
CR7	Healthcare professionals should be able to restrict what kind of health information the users can see
CR8	Healthcare professionals should be able to view multiple users at the same time
CR9	It should be possible to register a note or tag to an alarm, and the notes should be searchable
CR10	It should be possible to search for a user by name, date of birth or National Identification Number (NID)
CR11	The information tied to users should include: NID, phone number, sensor measurements, restriction of measurements, motivational messages, information messages for the user.
CR12	There should be authentication of the healthcare personnel

Table 4.1: The initial requirements from the customer

4.3 Final Requirements

4.3.1 Functional Requirements

Table 4.2 shows the final functional requirements for the system.

ID	User Story ID	Description	Pri.	Cmp.
FR-S1	US02	The sensor must monitor vital signs	H	L
FR-S2	US02	The sensor must communicate with a hub device using Bluetooth	H	M
FR-S3	US03	The sensor should cache data if communication with hub fails	L	M
FR-S4	US02	The sensor should send data to hub automatically	M	M

ID	User Story ID	Description	Pri.	Cmp.
FR-H1	US02	Hub should send data to server when new data is received by sensor	H	H
FR-H2	US17	Hub should communicate with a server using HL7	H	H
FR-H3	US03	Hub should be able to cache data for a certain period of time	L	M
FR-H4	US32	Hub should be able to start and resume work after a power outage	L	M
FR-B1	US02	The monitored values must be stored in a database	H	H
FR-B2	US11	The system must alert healthcare professionals when an abnormal value is registered	M	H
FR-B3	US04	The server must store information about the user	H	L
FR-B4	US28	The front-end interface should not display multiple consecutive alarms	M	L
FR-F1	US01	Actors need to be able to log in to the system through a web interface	H	M
FR-F2	US10	Web page should display historical data for monitored values to the user	M	L
FR-F3	US01	Access to the front-end interface needs to be password protected	M	L
FR-F4	US04	Front-end must display a list of registered users	M	L
FR-F5	US05	Front-end must allow searching for users by name, date of birth or NID	M	L
FR-F6	US06	Front-end needs to allow healthcare professionals configure threshold values for users	M	L
FR-F7	US07	Front-end needs to allow healthcare professionals configure what information is visible for the user	L	L
FR-F8	US08	Front-end needs to display data as graphs	L	H

ID	User Story ID	Description	Pri.	Cmp.
FR-F9	US13	Front-end needs to display data as a table	M	M
FR-F10	US09	Front-end needs to allow healthcare professionals change the time span of a graph	L	L
FR-F11	US12	Front-end needs to present a list of alarms	H	M
FR-F12	US14	Front-end needs to have a tab-based interface	L	M
FR-F13	US15	Front-end needs to have a customized, mobile-friendly view for when a user (not healthcare professional) is logged in	L	M
FR-F14	US16	Front-end needs to allow changing of global system settings	L	L
FR-F15	US17	Front-end needs to use Hyper Text Transfer Protocol (HTTP) Secure (HTTPS)	L	M
FR-F16	US18	The front-end interface must display a list of next of kin for the users, with relations.	M	L
FR-F17	US19	The front-end interface for users should make it possible to give voice messages to the users.	L	H
FR-F18	US20	The system should store previous motivational/informative messages for users, and the front-end interface for the users should display them	M	L
FR-F19	US21	The front-end interface for users should have the ability to notify the healthcare professionals that the user would like to be called.	M	L
FR-F20	US22	The system should store previous threshold values, and display them on the front-end interface.	H	M
FR-F21	US23	The front-end interface should allow healthcare professionals to choose what kind of information they want to see from each patient.	H	L
FR-F22	US24	The front-end interface should have a form for adding new users to the system, with standard values preset in the form.	H	M

ID	User Story ID	Description	Pri.	Cmp.
FR-F23	US25	The front-end interface should highlight abnormal values in the list view	H	M
FR-F24	US26	The front-end interface should play a sound when new alarms are received	M	M
FR-F25	US27	Graphs in the front-end interface should display handled alarms differently than unhandled alarms	M	L
FR-F26	US29	Graphs in the front-end interface should update automatically with regular intervals to get live data	M	M
FR-F27	US30	The front-end interface should display a message instead of no values if there are no data	M	M
FR-F28	US31	Phone numbers in the front-end interface should be clickable	L	L

Table 4.2: Functional requirements

4.3.2 Non-Functional Requirements

- Modifiability: The system should be built in such a way that it makes the system easy to extend or alter, for example by making it easy to use another sensor or add additional functionality.
- Usability: The user interface should be quick to learn and easy to use.
- Availability: The system should provide a reliable service. Healthcare professionals should be able to rely on the system to display current and correct data from all sensors.

4.3.2.1 Modifiability

The modifiability tactics are described in section 6.5.1.

ID	Source	Stimilus	Environment	Artifact	Response	Response measure
M1	HP	Add sensor	Build time	System	Added sensor	Added sensor and related implementation within seven days
M2	Developer	Add model	Design time	Code	Add model	1 hour to make the change
M3	Developer	Modify view	Design time	Code	Modified model	2 hour to make the change

Table 4.3: Modifiability scenarios.

4.3.2.2 Usability

The usability tactics are described in section section 6.5.2.

ID	Source	Stimilus	Environment	Artifact	Response	Response measure
U1	End user	Cancel handling an alarm.	Runtime	System	Operation cancelled	Operation cancelled within 0.5 sec
U2	End user	Input text to search for user	Runtime	System	System sorts users based on compliance with search text	The system should instantly present users that complies with the search text, whether it matches birth data, NID, age, surname or first name
U3	End user	When editing user, click save when the form is invalid	Runtime	System	Display warning, do not save data	When the save button is clicked a warning should appear instantly, telling the system user that there is something wrong with the data written in the form

Table 4.4: Usability scenarios.

4.3.2.3 Availability

The availability tactics are described in section section 6.5.3.

ID	Source	Stimulus	Environment	Artefact	Response	Response measure
A1	Hub	Not transmitting data	Normal operation	Communication channel between hub and server	Include the missing data in the next transmission	Detect the fault, and add data to the next package within next transmission

Table 4.5: Availability scenarios.

4.3.3 Prioritization

The requirements are prioritized in three categories: a) High, b) Medium or c) Low.

High (H) Core functionality of the utility that must be implemented.

Medium (M) Requirements that will improve the value of the utility.

Low (L) Requirements that will not add much value to the utility.

4.3.4 Complexity

The complexity of each requirements has been estimated, and placed in the following categories:

High (H) Functionality that seems difficult and non-trivial to create.

Medium (M) Functionality that seems time consuming but straight forward.

Low (L) Requirements that are trivial to implement.

4.4 Requirements Evolution

4.4.1 Sprint 1

No new requirements were added during this sprint, but two new tasks to the functional requirement FR-B3 were added. The system should store the address of the users as well as next of kin. This resulted in the following new tasks:

FR-B3-H Store address

FR-B3-I Store next of kin

4.4.2 Sprint 2

In this sprint the implementation was well under way and the team could start to focus on details within the system. The requirements that were uncovered during this sprint was:

- It was established that the system is not a proof of concept, but rather a prototype that should be ready for testing by the customer upon delivery.

- The specifics of how next of kin should be presented was established:
  - Next of kin should be displayed in a prioritized list, with the main contact person on the top.
  - The information about next of kin should include the relation the person has to the user.
- In the users interface there should be a button that when pushed notifies the healthcare professionals that the user wishes to be contacted.
- It should be possible to add voice messages in addition to written messages and motivational texts. This is to ensure that the service is accessible also for users with reduced eyesight.
- Old motivational and information messages should be stored and possible for the user to see.
- The system should store previous threshold values in a history

This led to the following changes in requirements:

FR-F16 The front-end interface must display a list of next of kin for the users, with relations.

FR-F17 The front-end interface for users should make it possible to give voice messages to the users.

FR-F18 The system should store previous motivational/informative messages for users, and the front-end interface for the users should display them.

FR-F19 The front-end interface for users should have the ability to notify the healthcare professionals that the user would like to be called.

FR-F20 The system should store previous threshold values, and display them on the front-end interface.

4.4.3 Sprint 3

The customer provided the group with the following information in sprint 3:

- HL7 is no longer a requirement. After doing some research [58] and reading about HL7, both the customer and the group have realized that this standard is immature, lacks national guidelines and may therefore

not be a good choice given the scope of this project. The main motivation behind the requirement was to make the system open and easily extendable by using a message standard that soon may be applied to the Norwegian healthcare sector[50].

- High security is no longer a high priority in the project.
- Healthcare professionals should be able to choose what kind of information they want to see from each user. This is in order to not have to deal with irrelevant information.
- The language used in the service should only be Norwegian, no alternative language is required.
- The license for the project should be open and it should be possible to use in other research.
- In the future the system should become a part of an existing system made by Siemens.
- There should be a form for adding new users to the system
- Abnormal values should be highlighted in the list view
- A handled alarm should be displayed differently in graphs than an unhandled alarm
- A sound should be played when a new alarm is received in the system
- Healthcare professionals should have a setting for disabling sound effects
- A graph should not display multiple alarm icons next to each other if there are multiple measurements in a row with abnormal values
- Graphs should update automatically with regular intervals to get live data
- A telephone call request should be shown as an alarm
- New threshold values should be validated
- A message should be shown instead of a graph if there is no data
- A message should be shown instead of a list of sensor data if there is no sensor data

- A message should be shown instead of a list of alarms there are no alarms
- The current threshold values should be shown in list view
- There should be some standard values for the 'new user' form
- The alarms tab should have a number on it showing the number of unhandled alarms
- There should be an option to only show unhandled alarms in the alarms list
- Phone numbers should be clickable

Based on this, the following new requirements and tasks were created:

FR-F6-C The front-end interface should validate a new threshold value

FR-F11-C The alarms tab should show the number of unhandled alarms

FR-F11-D There should be an option to only show unhandled alarms in the alarms list

FR-B4 The front-end interface should not display multiple consecutive alarms

FR-F21 The front-end interface should allow healthcare professionals to choose what kind of information they want to see from each user.

FR-F22 The front-end interface should have a form for adding new users to the system, with standard values preset in the from.

FR-F23 The front-end interface should highlight abnormal values in the list view

FR-F24 The front-end interface should play a sound when new alarms are received

FR-F25 Graphs in the front-end interface should display handled alarms differently than unhandled alarms

FR-F26 Graphs in the front-end interface should update automatically with regular intervals to get live data

FR-F27 The front-end interface should display a message instead of no values if there are no data

FR-F28 Phone numbers in the front-end interface should be clickable

Also, these requirements and tasks were skipped:

FR-S1: The sensor must monitor vital signs The backup sensor does not require any work for it to start recording the vital signs needed.

FR-S2-A: Make sensor send data to hub This is no longer an issue, as the backup sensor is locked and the team cannot control how it communicates. It sends data to the third party server, which the system needs to access to get the data.

FR-S3: The sensor should cache data if communication with hub fails The backup sensor caches data automatically.

FR-H2: Hub should communicate with server using HL7 The customer does no longer require the use of HL7 in the scope of this project.

4.5 Requirement Description

This section gives a brief description of the requirements for the system. Each functional requirement has a user story as its "parent" (see table 4.2 for an overview over the connection between functional requirements and user stories), and each functional requirement is divided into one or more tasks (see table 4.7). The functional requirements are divided into four main categories: FR-S is sensor, FR-H is hub, FR-B is back-end, and FR-F is front-end related requirements.

FR-S1 The wearable sensor used in the system must monitor vital signs that are useful for health care professionals to monitor.

FR-S2 The sensor must use Bluetooth to communicate with a hub device. See section 3.9.5 for the justification of the choice of transfer technology.

FR-S3 If the sensor is unable to send data to the hub, it should cache data so that data is not lost.

FR-S4 The sensor should send data to the hub automatically, so that the user does not need to interact with the sensor in order to send data to the hub.

FR-H1 When the hub receives new data from the sensor, it should send it to the server as soon as possible.

FR-H2 The hub should use the HL7 guideline to send data to the server.

FR-H3 If the hub is unable to send data to the sensor, it should cache data so that data is not lost.

FR-H4 The hub should be able to restart and continue work after a power outage, so that the user does not have to interact with the hub to get it working.

FR-B1 The data received from the hub must be stored in a database.

FR-B2 When a recorded value that is outside the defined threshold values for a patient, the system must display an alarm to the healthcare professionals.

FR-B3 The information about the users must be stored in the database

FR-B4 Consecutive abnormal measurements of the same type should not create more than one alarm, because it is desirable to only have one alarm per incident.

FR-F1 The system must be web based, where actors can access the system from anywhere in the world, assuming they have an Internet connection.

FR-F2 The system needs to store and display all data recorded by a sensor, so that healthcare professionals can see historical vital sign measurements to better understand a user's current health situation.

FR-F3 An actor needs to have a username and password to access the system interface, to ensure system security and protect sensitive, personal information.

FR-F4 The front-end interface must offer a list with all registered users of the system, so that healthcare professionals can find users and access their health information.

FR-F5 The front-end interface must offer a search functionality in the list view (FR-F4) where name, date of birth or social security number is searchable, so that healthcare professionals quickly can find the users they are looking for.

FR-F6 The healthcare professionals must be able to edit the threshold values, or threshold values, for a user, as these might change over time.

FR-F7 The healthcare professionals must be able to change what information is visible for the user in the user's dashboard. For example, a healthcare professional can grant access to heart rate data if the user requests it, and can revoke the access if it is no longer desirable for the user to have access.

FR-F8 The front-end interface must display data as graphs, as this is often considered an easier way to get an overview over recent development of vital signs.

FR-F9 The front-end interface must display data as a table, as it is sometimes useful to see data represented as numbers instead of a graph.

FR-F10 Healthcare professionals need to be able to change the time span of a graph, so that they can see how the user's heath situation has developed over both a long and short time span.

FR-F11 The front-end interface needs to present a list of unhandled alarms, so that healthcare professionals easily can see what alarms needs to be looked at.

FR-F12 The front-end interface needs to be a tab-based interface, because healthcare professionals often need to have multiple users open at the same time, and they need to quickly switch between users. A tab-based interface offers this.

FR-F13 The front-end interface for users (not healthcare professionals) must be mobile-friendly, because users must be able to access the system by a mobile device, such as an iPad. It should display graphs for the vital sign measurements.

FR-F14 The front-end interface must allow actors to edit the global settings for the system (settings that are not linked to specific users).

FR-F15 The front-end needs to communicate with back-end using HTTPS, to ensure secure transfer of data.

FR-F16 The front-end interface must display a list of next of kin for the users, with relations, so that healthcare professionals know who to contact in case a medical emergency should occur.

FR-F17 The front-end interface for users should make it possible to give voice messages to the users, as this might be preferable to text messages for some users.

FR-F18 The system should store previous motivational/informative messages for users, and the front-end interface for the users should display them, as it might be interesting for users to see not only the newest motivational text

FR-F19 The front-end interface for users should have the ability to notify the healthcare professionals that the user would like to be called, as this gives users an easy way to signal that they have some kind of issue they want to take up with healthcare professionals.

FR-F20 The system should store previous threshold values, and display them on the front-end interface, as this is useful for seeing how a user's health has developed over time, and give correct threshold values to old alarms.

FR-F21 The front-end interface should allow healthcare professionals to choose what kind of information they want to see from each patient, to prevent irrelevant information to be shown if only some of the measurement types are relevant.

FR-F22 The front-end interface should have a form for adding new users to the system, with standard values preset in the form, so that healthcare professionals easily and quickly can add new users.

FR-F23 The front-end interface should highlight abnormal values in the list view, so that healthcare professionals easily can see abnormal values in this view.

FR-F24 The front-end interface should play a sound when new alarms are received, so that healthcare professionals can be notified about new alarms even if they are not looking at the screen.

FR-F25 Graphs in the front-end interface should display handled alarms differently than unhandled alarms, so that healthcare professionals can easily see if an alarm is handled or not. There should also be an option for muting the sound, because healthcare professionals might find the sound irritating if they do not need it, and should be able to turn it off.

FR-F26 Graphs in the front-end interface should update automatically with regular intervals to get live data, so that healthcare professionals do not need to refresh the page manually.

FR-F27 The front-end interface should display a message instead of no values if there are no data. This is so that the healthcare professional will know that there are no data, rather than think that there is something wrong with the system.

FR-F28 Phone numbers in the front-end interface should be clickable, so that the healthcare professionals can easily call the user when using a tablet or phone.

4.6 User Stories

Table 4.6 shows the user stories.

ID	Description
US01	As a user of the system, I want to log in to the system to be able to use its features.
US02	As a user, I want the data recorded by my wearable sensor to be sent to the system, so that healthcare professionals can be alerted if there is an unusual value.
US03	As a user, I want data to be stored even when it is not possible to send data to the system, so that no data is lost.
US04	As a healthcare professional, I want to see a list of all users in the system, so that I can get an overview of the registered users.
US05	As a healthcare professional, I want to search for a user by name or social security number, so that I can easily find the user I am looking for.
US06	As a healthcare professional, I want to change the threshold values for a user, because these are different from user to user, and they can change over time.
US07	As a healthcare professional, I want to select what information is available for the user to see, because by default, no information is available for the user to see, and he/she might want to see it.
US08	As a healthcare professional, I want to see the data recorded from a users sensor presented in graphs, so that I can easily see how the current values compares to earlier values.

ID	Description
US09	As a healthcare professional, I want to change the time span for the graphs, so that I can set how far back in time I compare the current values to.
US10	As a healthcare professional, I want to see historical data for a user, so that I can see how his/her values have developed over time.
US11	As a healthcare professional, I want to get an alarm when a registered value is outside the specified threshold values, and be given the opportunity to mark this alarm as handled, because a value outside the threshold values could mean a health risk for the user.
US12	As a healthcare professional, I want to see a sorted list of all unhandled alarms, because this gives me an overview which users might need assistance.
US13	As a healthcare professional, I want to see the data recorded from a users sensor presented with numbers, so that I can see the current and past values.
US14	As a healthcare professional, I want to be able to switch between users quickly using a tab-based interface, because this makes the workflow faster and easier.
US15	As a user, I want a web interface that shows the information I have access to, because I want to see how my vital signs have developed over time.
US16	As a healthcare professional, I want to change settings for the system, because I want to customize how the system works.
US17	As a user, I want the information to be securely transferred, so that my privacy is protected.
US18	As a healthcare professional, I want to see a list of next of kin to a user, so that I can alert them if the user has a medical emergency.
US19	As a healthcare professional, I want to make an audio recording of a message to a user, because this will both save me time and the user might find it easier to listen to a message rather than to read it.
US20	As a user, I want to see multiple motivational and/or informative messages, because previous messages might be important.

ID	Description
US21	As a user, I want to have a button for requesting healthcare professionals to call me, because I might need assistance and would like to easily notify healthcare professionals that I want to be contacted.
US22	As a healthcare professional, I want to see previous threshold values for a user, so that I can see how his/her health has developed over time.
US23	As a healthcare professional, I want to be able to choose what kind of information I see for each user, because some types of vital signs might be irrelevant for some users.
US24	As a healthcare professional, I want to be able to use a form to add new users to the system, and this form should have some default values, so that it is quick and easy for me to add a new user to the system.
US25	As a healthcare professional, I want cells with abnormal values in list view to be highlighted, so that I can easily see the abnormal values.
US26	As a healthcare professional, I want to hear a sound when a new alarm is registered, so that I am alerted even if I am not looking at the screen.
US27	As a healthcare professional, I want to see different icons in graphs for handled alarms and unhandled alarms, so that I can easily see if an alarm has been handled or not.
US28	As a healthcare professional, I want to see only one alarm even if there are multiple consecutive abnormal measurements, because I only want to see one alarm for each incident.
US29	As a healthcare professional, I want graphs to update automatically when new data is available, so that I don't need to refresh the page manually.
US30	As a healthcare professional, I want to see a message if there are no data in the system, so that I do not see an empty table.
US31	As a healthcare professional, I want phone numbers to be clickable, so that I can quickly call a number if I am using a device that allows phone calls.
US32	As a user, I want the hub to restart and continue working seamlessly even after a power outage.

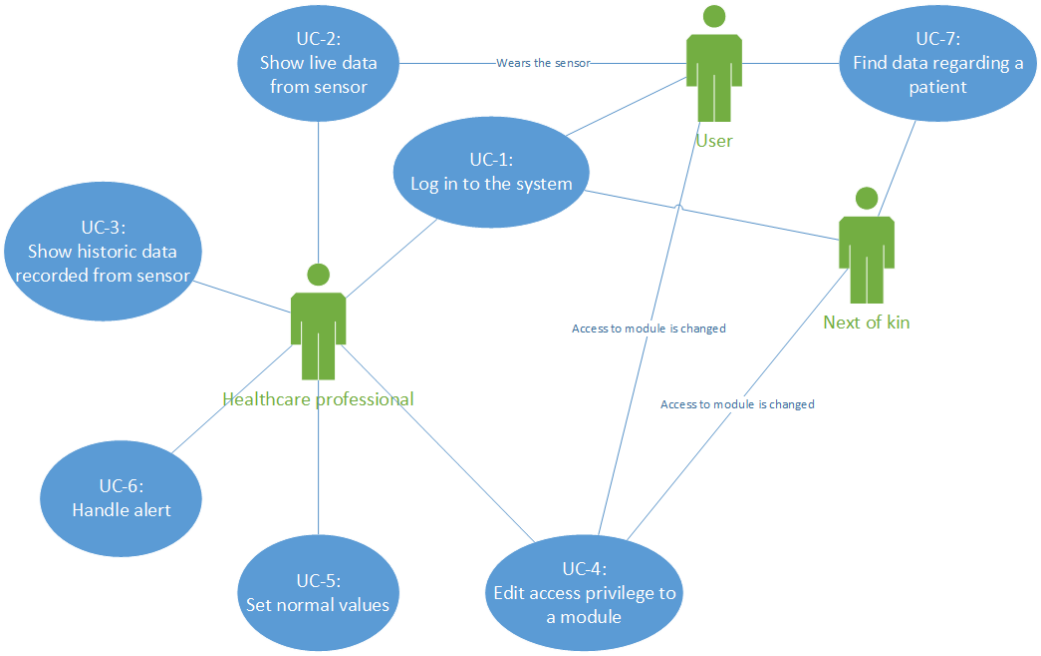

Figure 4.1: Use case diagram

ID	Description

Table 4.6: User stories

4.7 Use Cases

Creating use cases was also considered, but after discussing this with the advisor, the team were told that it would not be necessary to create both user stories and use cases. However, the team spent some time creating use cases before this was decided, and a use case diagram created in the initial phase of the project can be seen in figure 4.1.

4.8 Product Backlog

Table 4.7 shows the complete product backlog containing all the functional requirements and their tasks. It provides a short description for each task, the sprint it was completed in, and the estimated and actual time spent on the

task. The column for actual time spent does not include fixes, improvements or refactoring done in later sprints, testing or documentation.

			Hours	
Req.	Description	Sprint	Est.	Act.
FR-S1	*The sensor must monitor vital signs*		**8**	**N/A**
FR-S1-A	*Make sensor monitor blood oxygen*	N/A	2	N/A
FR-S1-B	*Make sensor monitor heart rate*	N/A	2	N/A
FR-S1-C	*Make sensor monitor heart rate*	N/A	2	N/A
FR-S1-D	*Make sensor monitor heart rate*	N/A	2	N/A
FR-S2	The sensor must communicate with a hub device using Bluetooth		**10**	**20**
FR-S2-A	*Make sensor send data to hub*	N/A	5	N/A
FR-S2-B	Make hub receive data from sensor	SP2	5	20
FR-S3	*The sensor should cache data if communication with hub fails*		**5**	**N/A**
FR-S3-A	*Cache data up to 7 days in sensor if failed to establish connection to hub*	N/A	5	N/A
FR-S4	The sensor should send data to hub automatically		**24**	**30**
FR-S4-A	Make sensor send data to hub when new data is recorded	SP3	24	30
FR-H1	Hub should send data to server when new data is received by sensor		**28**	**24**
FR-H1-A	Make hub send data to server when new data is received from sensor	SP3	28	24

			Hours	
Req.	Description	Sprint	Est.	Act.
FR-H2	*Hub should communicate with a server using HL7*		**40**	**N/A**
FR-H2-A	*Use HL7*	N/A	40	N/A
FR-H3	Hub should be able to cache data for a certain period of time		**15**	**20**
FR-H3-A	Make sure data is cached in hub if a connection cannot be established with the server	SP2	15	20
FR-H4	Hub should be able to start and resume work after a power outage		**10**	**2**
FR-H4-A	Make sure hub starts and continues work automatically after a power outage	SP3	10	2
FR-B1	The monitored values must be stored in a database		**24**	**24**
FR-B1-A	Store blood oxygen data in database	SP1	2	1
FR-B1-B	Store heart rate data in database	SP1	2	1
FR-B1-C	Store activity data in database	SP1	2	1
FR-B1-D	Store temperature data in database	SP1	2	1
FR-B1-E	Store data from the hub in database	SP3	16	19
FR-B2	The system must alert healthcare professionals when an abnormal value is registered		**35**	**26**
FR-B2-A	Generating alarm on back-end	SP2	10	8
FR-B2-B	Receiving alarm on front-end	SP2	5	3
FR-B2-C	Front-end for handling of alarm	SP2	10	7
FR-B2-D	Back-end for handling of alarm	SP2	10	8

Req.	Description	Sprint	Hours Est.	Hours Act.
FR-B3	The server must store information about the user		**9**	**4.5**
FR-B3-A	Store name	SP1	1	0.5
FR-B3-B	Store NID	SP1	1	0.5
FR-B3-C	Store phone number	SP1	1	0.5
FR-B3-D	Store restrictions of user	SP1	1	0.5
FR-B3-E	Store motivational messages	SP1	1	0.5
FR-B3-F	Store information messages	SP1	1	0.5
FR-B3-G	Store threshold values	SP1	1	0.5
FR-B3-H	Store address	SP1	1	0.5
FR-B3-I	Store next of kin	SP1	1	0.5
FR-B4	Consecutive abnormal measurements of the same type should not generate more than one alarm		**4**	**5**
FR-B4-A	Create only one alarm per incident	SP3	4	5
FR-F1	Actors need to be able to log in to the system through a web interface		**16**	**17**
FR-F1-A	Create basic web interface	SP1	8	10
FR-F1-B	Log-in screen	SP1	2	3
FR-F1-C	Log out option	SP1	2	1
FR-F1-D	Database with users table	SP1	4	3
FR-F2	Web page should display historical data for monitored values to the user		**8**	**12**
FR-F2-A	Code for getting all data about a user from database to front-end	SP3	8	12

			Hours	
Req.	Description	Sprint	Est.	Act.
FR-F3	Access to the front-end interface needs to be password protected		**6**	**9**
FR-F3-A	Password protect front-end	SP1	6	9
FR-F4	Front-end must display a list of registered users		**8**	**10**
FR-F4-A	Front-end for list of users	SP2	3	2
FR-F4-B	Back-end for providing list of users	SP2	3	5
FR-F4-C	Database with user table	SP2	2	3
FR-F5	Front-end must allow searching for users by name, date of birth or social security number		**2**	**1**
FR-F5-A	Front-end search functionality	SP2	2	1
FR-F6	Front-end needs to allow healthcare professionals to configure threshold values for users		**9**	**5**
FR-F6-A	Front-end for normal value settings	SP2	3	2
FR-F6-B	Back-end for normal value settings	SP2	3	2
FR-F6-C	The front-end interface should validate a new threshold value	SP2	3	1
FR-F7	Front-end needs to allow healthcare professionals to configure what information is visible for the user		**6**	**5**
FR-F7-A	Front-end for user access settings	SP2	3	2
FR-F7-B	Back-end for user access settings	SP2	3	3

			Hours	
Req.	Description	Sprint	Est.	Act.
FR-F8	Front-end needs to display data as graphs		**20**	**24**
FR-F8-A	Front-end for activity graph	SP2	5	6
FR-F8-B	Front-end for blood oxygen graph	SP2	5	6
FR-F8-C	Front-end for heart rate graph	SP2	5	6
FR-F8-D	Front-end for temperature graph	SP2	5	6
FR-F9	Front-end needs to display data as a table		**10**	**8**
FR-F9-A	Front-end for list view	SP3	10	8
FR-F10	Front-end needs to allow healthcare professionals to change the time span of a graph		**4**	**6**
FR-F10-A	Front-end for selecting graph time span	SP2	4	6
FR-F11	Front-end needs to present a list of unhandled alarms		**10**	**9**
FR-F11-A	Front-end for alarm list	SP2	2	2
FR-F11-B	Back-end for alarm list	SP2	3	3
FR-F11-C	The alarm tab should show the number of unhandled alarms	SP3	3	2
FR-F11-D	There should be an option to only show unhandled alarms in the alarms list	SP3	2	2
FR-F12	Front-end needs to have a tab-based interface		**15**	**24**
FR-F12-A	Front-end for tab-based interface	SP2	15	24
FR-F13	Front-end needs to have a customized, mobile-friendly view for when a user is logged in		**10**	**9**

			Hours	
Req.	Description	Sprint	Est.	Act.
FR-F13-A	Customized front-end view for users	SP3	5	6
FR-F13-B	Make view mobile-friendly	SP3	5	3
FR-F14	Front-end needs to allow changing of global system settings		**6**	**8**
FR-F14-A	Front-end for global system settings	SP3	3	4
FR-F14-B	Back-end for global system settings	SP3	3	4
FR-F15	Front-end needs to use HTTPS		**16**	**24**
FR-F15-A	Use HTTPS	SP3	16	24
FR-F16	The front-end interface must display a list of next of kin for the users, with relations		**6**	**6**
FR-F16-A	Front-end for next of kin	SP2	3	4
FR-F16-B	Back-end for next of kin	SP2	3	2
FR-F17	The front-end interface for users should make it possible to give voice messages to the users		**11**	**6**
FR-F17-A	Front-end implementation of voice message for healthcare professionals	SP3	5	2
FR-F17-B	Front-end implementation of voice message for users	SP3	3	2
FR-F17-C	Back-end storing of voice messages	SP3	3	2
FR-F18	The system should store previous motivational/informative messages for users, and the front-end interface for the users should display them		**10**	**8**

			Hours	
Req.	Description	Sprint	Est.	Act.
FR-F18-A	Front-end for motivational/informative messages	SP2	5	3
FR-F18-B	Back-end for motivational/informative messages	SP2	5	5
FR-F19	The front-end interface for users should have the ability to notify the healthcare professionals that the user would like to be called		**5**	**4**
FR-F19-A	Add 'call me' button on user dashboard	SP2	1	1
FR-F19-B	Handle call request on back-end	SP3	2	1
FR-F19-C	Display call request in alarm list	SP3	2	2
FR-F20	The system should store previous threshold values, and display them on the front-end interface		**11**	**12**
FR-F20-A	Display previous threshold values on the graphs	SP2	6	7
FR-F20-B	Back-end should store old threshold values	SP2	5	5
FR-F21	The front-end interface should allow healthcare professionals to choose what kind of information they want to see from each patient		**5**	**5**
FR-F21-A	Settings for changing the visible information	SP3	5	5
FR-F22	The front-end interface should have a form for adding new users to the system, with standard values preset in the from.		**5**	**4**
FR-F22-A	Front-end form for adding new users	SP3	5	4

			Hours	
Req.	Description	Sprint	Est.	Act.
FR-F23	The front-end interface should highlight abnormal values in the list view		**1**	**1**
FR-F23-A	Front-end for highlighting of abnormal values	SP3	1	1
FR-F24	The front-end interface should play a sound when new alarms are received		**5**	**7**
FR-F24-A	Implementation of notification sound for alarms	SP3	4	6
FR-F24-B	Setting for muting of notification sound	SP3	1	1
FR-F25	Graphs in the front-end interface should display handled alarms differently than unhandled alarms		**2**	**1**
FR-F25-A	Give handled alarms a different icon than unhandled alarms	SP3	2	1
FR-F26	Graphs in the front-end interface should update automatically with regular intervals to get live data		**3**	**4**
FR-F26-A	Update graphs automatically	SP3	3	4
FR-F27	The front-end interface should display a message instead of no values if there are no data		**5**	**3**
FR-F27-A	Show a message instead of an empty graph if there are no graph data	SP3	1	1
FR-F27-B	Show a message instead of an empty list if there are no list data	SP3	1	0.5
FR-F27-C	Show a message instead of an empty alarm list if there are no alarms	SP3	1	0.5

			Hours	
Req.	Description	Sprint	Est.	Act.
FR-F27-D	Show a message instead of an empty motivational texts list if there are no motivational texts	SP3	1	0.5
FR-F27-E	Show a message instead of an empty informative texts list if there are no informative texts	SP3	1	0.5
FR-F28	Phone numbers in the front-end interface should be clickable		**1**	**1**
FR-F28-A	Make phone numbers in front-end clickable	SP3	1	1

Table 4.7: Product backlog.

Chapter 5

Test Plan

5.1 General

This chapter describes the test plan for the project. Methods and requirements will briefly be discussed, before looking at general routines and sprint test plans.

5.2 Methods for Testing

5.2.1 White Box Testing

White box testing is when you know what your code looks like and you use this to test a certain structure of a certain code. Django's unit tests use a Python standard library module:unittest. Testing is done by first writing a test that will test a certain function, then run the test by writing "./manage.py test" in the terminal/command line.

5.2.2 Black Box Testing

Black box testing is when you test your applications functionality without having any knowledge of the code. You know what the output should be and you check if your application gives the right output. The code itself is irrelevant.

5.3 Non-Functional Requirements

Testing of the non-functional requirements found in section 4.3.2 was planned for sprint 3. These requirements could not be completed by simply evaluating the source code. User Acceptance Testing (UAT) was planned, using the user stories as a test plan. A full scale test like this is important to make sure that the requirements are fulfilled, and that the system meets the expectations of the customer. It also provides a mean to determine how ”done” the system is.

5.4 Templates for Testing

Test Case ID:
Test Case Name:
Tester:
Description:

Pre-conditions:

Step	Action	Data	Exp. Res.	Act. Res.	Pass/fail	Comment
test	test	test	test	test	test	test
test	test	test	test	test	test	test
test	test	test	test	test	test	test

Table 5.1: Template for testing

5.5 Test Criteria

For most of the test cases, they will be considered a success if the output matches the expected result. Naturally, a test will be considered a failure if the output varies from the expected result. Some tests do not have an explicit output, and will have to be considered individually for success through varied

methods. This applies to some of the non-functional testing that are to be done with external resources.

5.6 Testing Responsibilities

Each person is responsible for writing unit testing that passes before committing new code. The test leader is responsible for the quality of the test plan and tests. When making changes to the code, the developer has to make sure all tests still pass before committing.

5.6.1 Testing Routines

Testing was to be done continuously during a sprint. This to ensure that new code did not break old code when submitted. Tests for back-end was written in the included test suite [38]

5.7 Sprint Testing

5.7.1 Sprint 1

In sprint 1, the team expected to have rudimentary testing of code.

5.7.2 Sprint 2

In sprint 2, testing should be more extensive, and tests should be run before pushing new code to git-hub.

5.7.3 Sprint 3

A UAT is planned for this sprint. This will be a test of the whole system, and an important step to verify that the system cover all the requirements. A test plan will have to be produced using the user stories. Furthermore a stress test of the system was planned for this sprint, to see if the system could handle large amounts of data.

Chapter 6

Architectural Description

6.1 General

This chapter describes the general architecture of the system. It contains architectural drivers, tactics, patterns and views.

6.2 Architectural Drivers

6.2.1 Motivation

The customer wanted the possibility to use different sensors, and also add new measurements. The system had do be easy to use and it had to be reliable. This set the focus for the quality attributes modifiability, usability, and availability. With high modifiability it should be easier to change sensors. The customer should also be able to easily implement secure transmissions after the project. Focusing on usability meant that the system should be easy do use for both the healthcare professionals and users. With high availability the hub should preferably always be connected to the server, but if it loses connection it should retransmit the data when it reconnects.

6.2.2 Expected Life Time

The development team will not continue working on this project after this course. However it should not be difficult for the customer to maintain it in the future. The customer wanted a functioning product that they could implement in their services as a prototype, and maybe even continue development for extended use in the future. However, it is possible for the customer

to abandon this project if they do not feel the need for this product, or if they find something better.

6.2.3 Experience of the Team

The team members are all experienced in programming and are familiar with system development, albeit mostly theoretical. However, the technologies used (Django and Angular), were new to most of the members. Working together in such a large group over such a long time was also a new experience. Overall, the project presented many new challenges, and provided opportunities to develop new skills.

6.2.4 Business Requirements

The customer cannot maintain a mobile application, but they can maintain a web application. This means that if the team was to create or use an existing mobile application for this project, it would have to be maintained by someone else than the customer.

- The system should run on different platforms, and be as open as possible.
- The system should be ready and functioning before the deadline, since it is not possible for the team to work on it after the project.
- The product is not a proof of concept, but should be a working prototype, ready for field testing.

6.3 Architectural Patterns

6.3.1 Model-view-controller

MVC is an architectural pattern where the implementation is divided into three main components, the model, view, and controller. Data is stored in the model and displayed in the view, while the controller is used to manipulate the model through the view. Each entity in figure 7.1 has its own model. In Django, the view is actually what is called the controller in MVC, it controls what should be displayed in the view, while the serializer is the Django equivalent of the MVC view.

6.3.2 Layered Pattern

The Layered Pattern is an architectural pattern in which the entire system is partitioned into smaller systems called layers. This is done in such a manner that portions of the system can be developed and evolved individually. This pattern was used to partition the software into three parts: Hub, back-end, and front-end. This was done in order to make it possible to change the technologies without rewriting the whole system. For example, the hub could be replaced with a mobile device in the future without having to change the back-end or the front-end. This layered structure also makes it easy to change the sensor, and this is one of the main reasons the system was engineered this way.

6.4 Architectural Views

6.4.1 4+1 View Model

This project uses the 4+1 View Model. This model suggests four different views: Logical, process, Development and physical view. In the case of the Angelika system, only three of the views has been used. The Angelika web application is too complex for a process view. The reason for that will be discussed below.

6.4.2 Process View

The view in figure: 6.1 is a very simple presentation of the process view. It would not be constructive to draw more of this view because of the amount of states possible when logged in as a healthcare professional. The system has too many states to draw a process view diagram that is both correct and readable.

6.4.3 Logical View

The purpose of the logical view is to describe the functionality provided to end users. It also shows the classes and the relationships between them.

Figure 6.2 shows a class diagram for the front-end of the system, and figure 6.3 shows the logical view for the hub. For an ER diagram of the back-end, see figure 7.1.

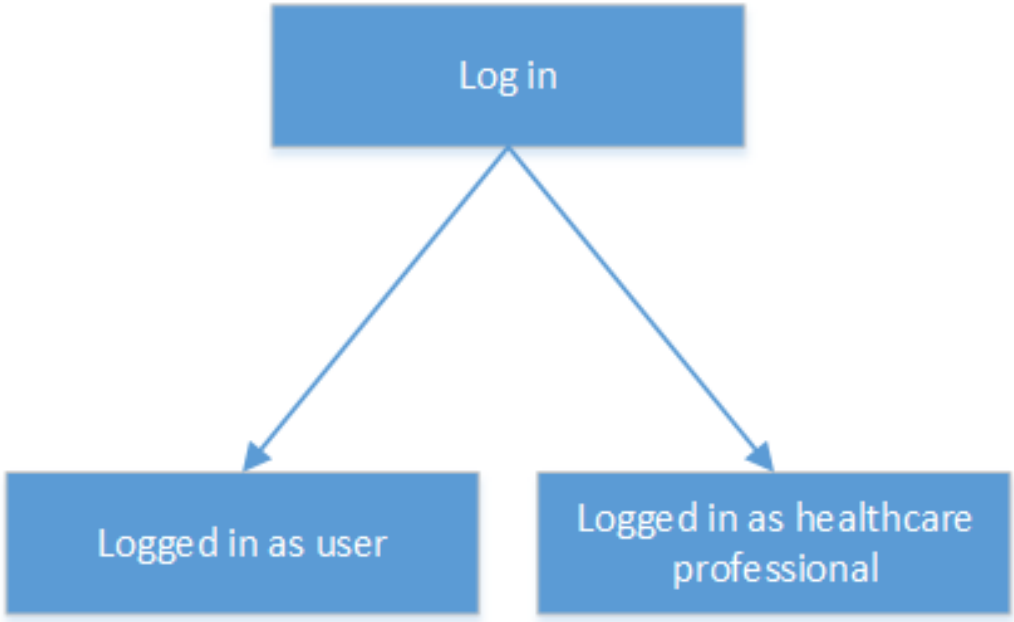

Figure 6.1: Process view

6.4.4 Development View

The development view is used to build up a code structure and its dependencies. It describes the architecture that supports the development process and eases the development. Having a standard also ensures integrity.

Figure 6.4 shows the development view for the layered pattern. The front-end contains the implementation of the Graphic User Interface (GUI), the part that interacts with the user. The back-end controllers contains the logic and data, and handles the data fetched by the front-end. The hub gathers new data and sends it to the back-end.

Figure 6.5 shows the MVC development view. This view is quite typical, except for the fact that it has two types of views. One containing the serializers which is the view from the back-end perspective and one containing templates which is from the front-end perspective. The model contains all the data. The data is displayed in the view and the controller uses the input from the view to manipulate the data in the model.

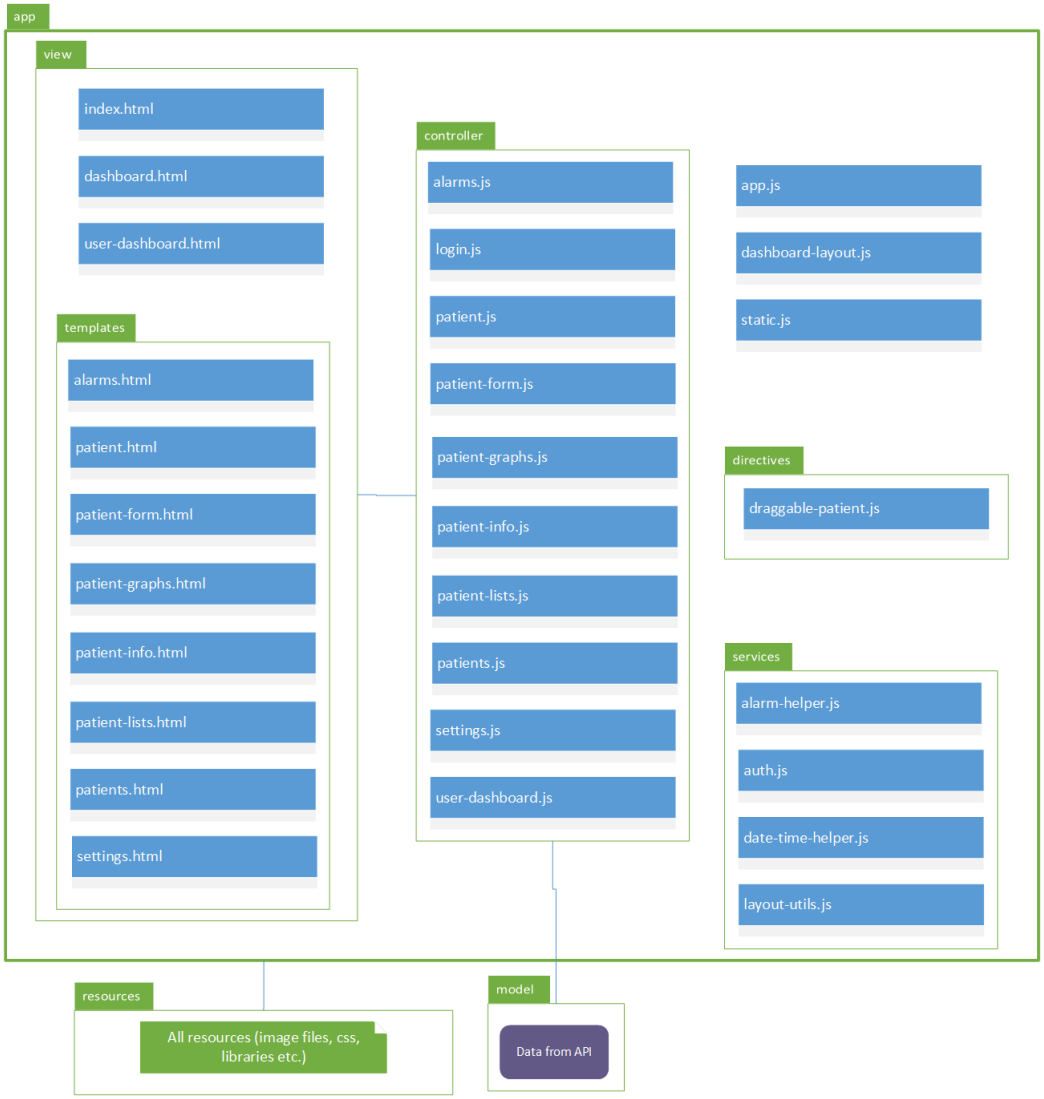

Figure 6.2: Class diagram, front-end

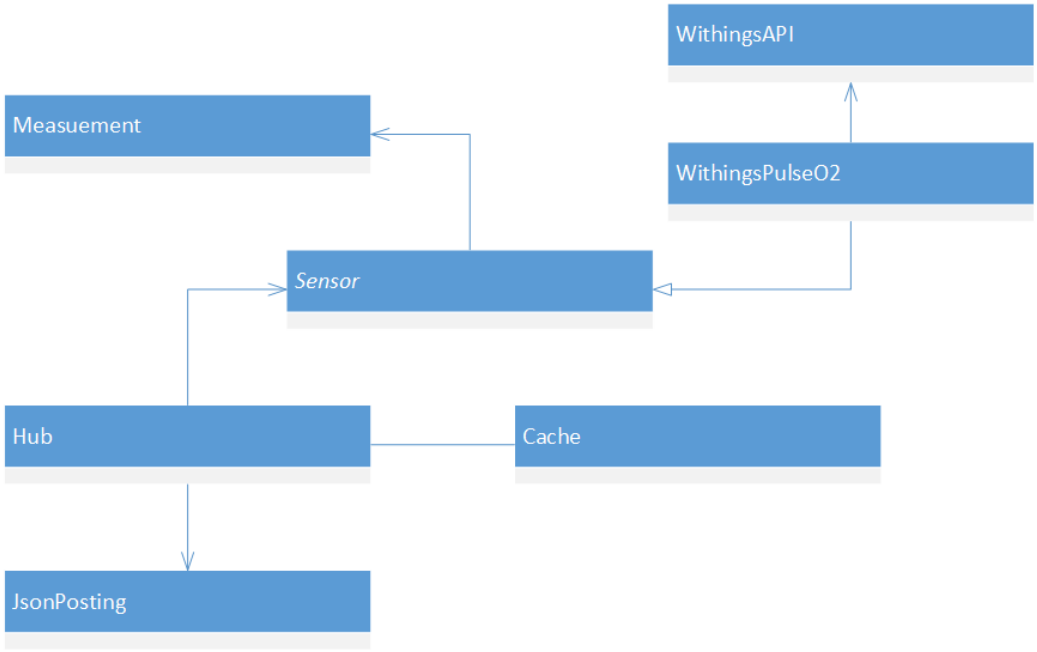

Figure 6.3: Class diagram, hub

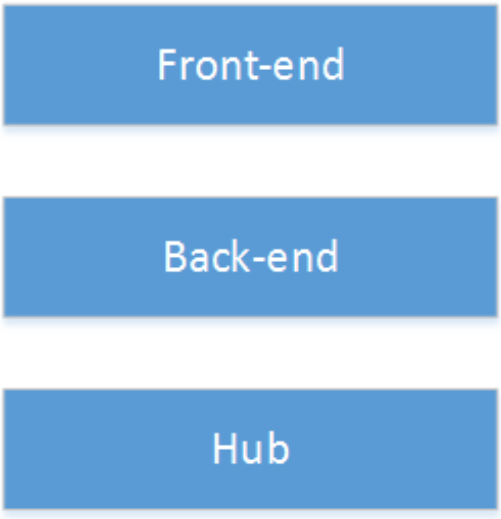

Figure 6.4: Layered pattern

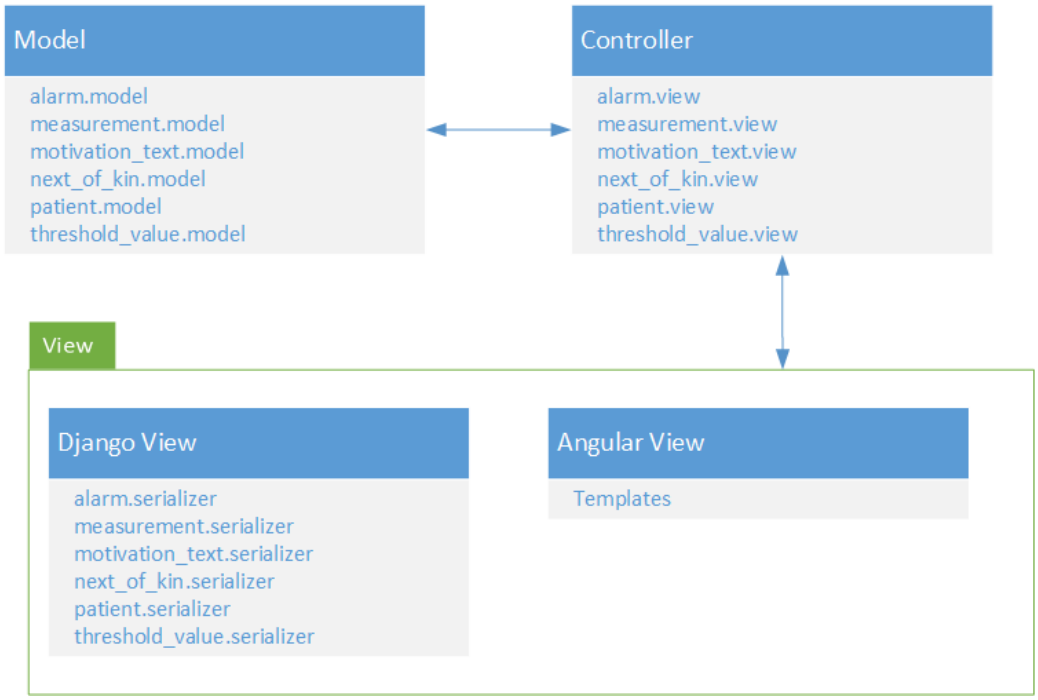

Figure 6.5: MVC

6.4.5 Physical View

The physcial view presents the system from a system engineers point of view. It is concerned with topology of the components on a physical level.

Initially the team created a physical view, shown in figure 6.6. The view describes an architecture where a sensor communicates with a hub, using a USB connected Bluetooth dongle. The hub then send data to a server that consists of a database and an API. The API handles requests from healthcare professionals and users. These two user groups have different user interfaces, the users interface is very simple and has little functionality.

This is the desired architecture by the customer. The Angel sensor does not need any user interaction (other than charging), and the data is not stored on any external server owned by a third party. However, as the project progressed, and the team got indications that the Angel sensor might not arrive in time, an updated diagram with the backup sensor was created. This is shown in figure 6.7.

The main difference between the original architecture from figure 6.6 and the updated architecture described in figure 6.7, is that the data from the backup sensor needs to go through a third party server before it can reach the

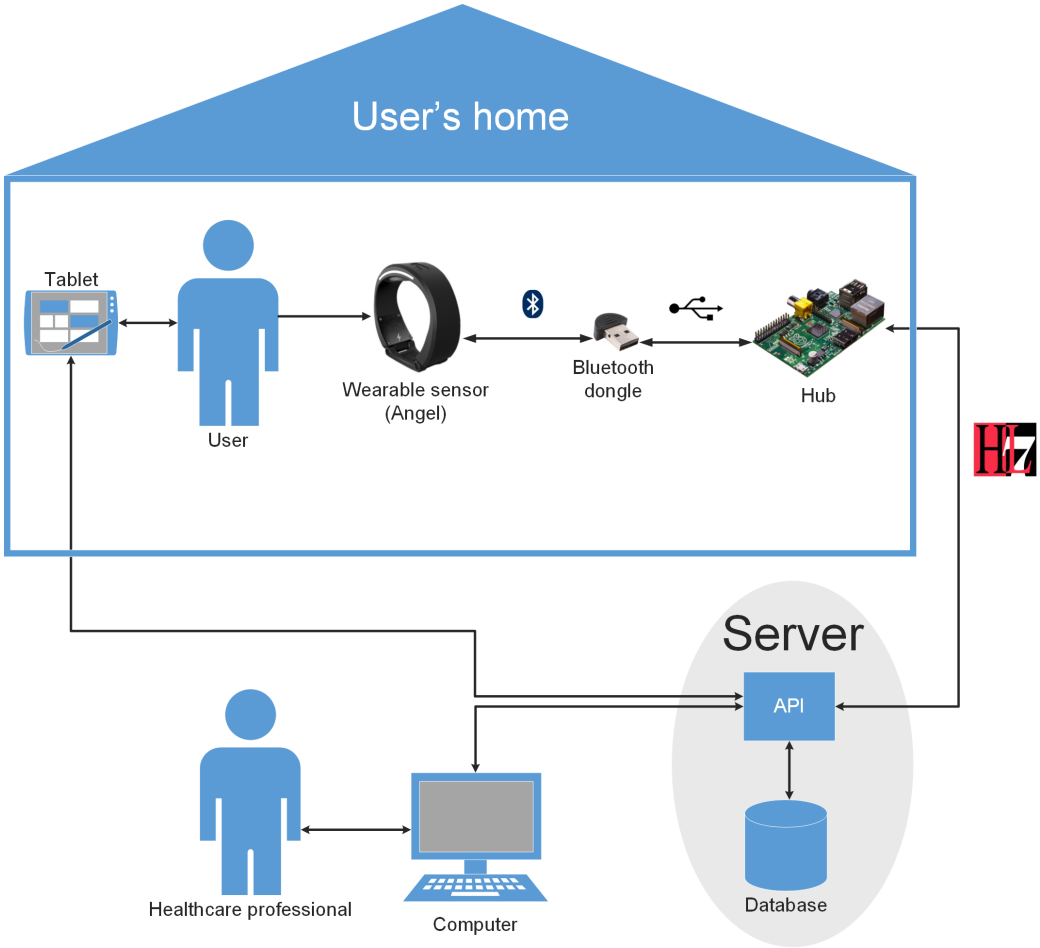

Figure 6.6: Initial physical view diagram

hub. This means that the data, which contains sensitive information about vital signs, can be accessed by the third party provider.

The updated architecture is not an ideal solution, but the team has not been able to find a wearable sensor currently on the market that would allow data to be sent directly to the hub.

6.5 Architectural Tactics

Architectural tactics are decisions about the overall design.

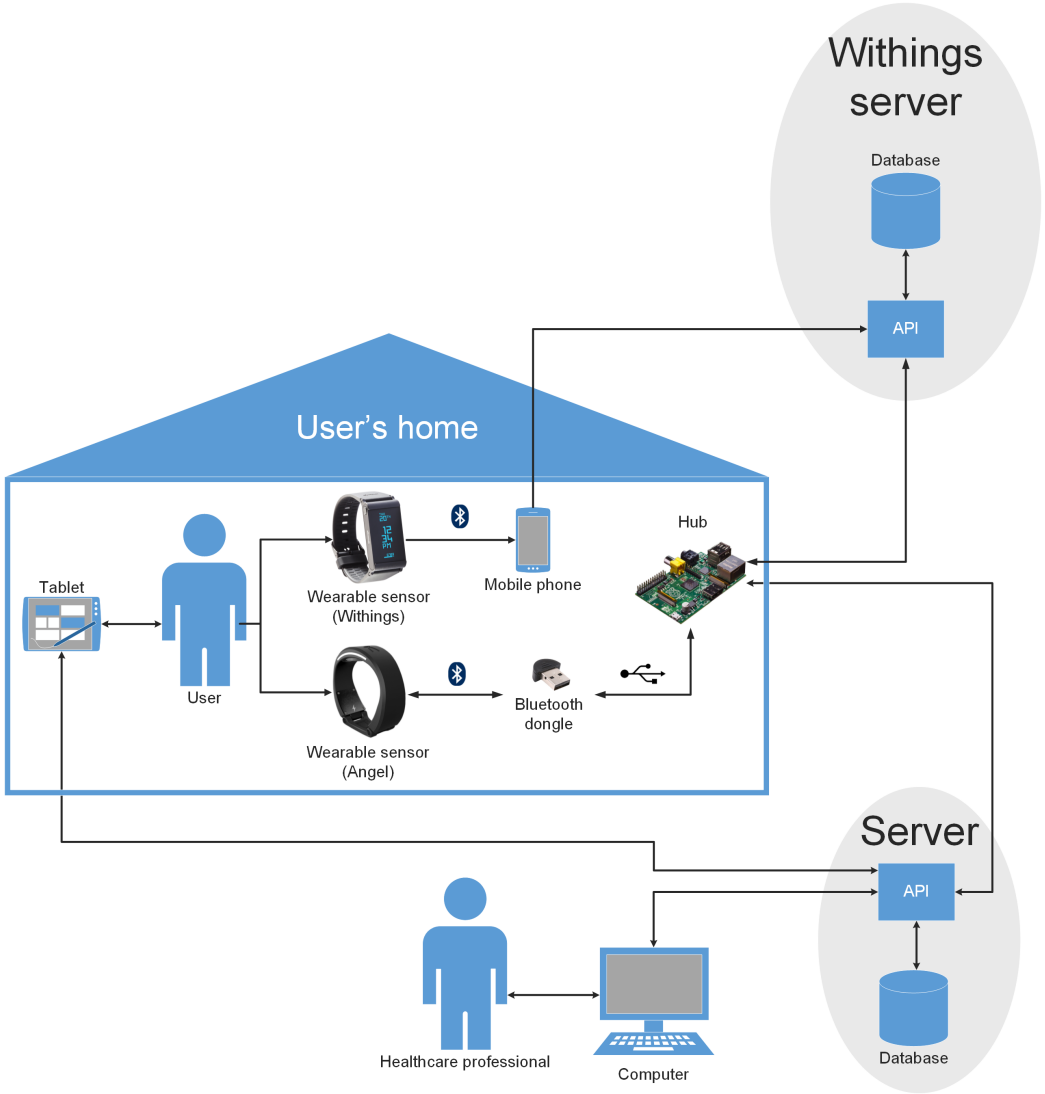

Figure 6.7: Updated physical view diagram

6.5.1 Modifiability Tactics

Tactics that were chosen to improve modifiability [14] is described in table 6.1.

Req.	Tactic	How	Why
M1	Resource files	By maintaining a configuration file with information regarding what sensor is connected to the hub, what URL the hub should post to and intervals for how often data should be posted, there is no need to recompile the code if the hub needs to be reconfigured.	Defer binding
M2 and M3	Increase semantic coherence.	Place responsibilities that do not serve the same purpose in different modules. Do not group responsibilities concerning models in modules with other responsibilities.	Increase cohesion. Lower the chance to affect different responsibilities when changing a module.
	Restrict dependencies.	Layered pattern. Have three seperated layers: the hub, back-end, front-end.	Reduce coupling. Lower the chance of affecting multiple modules when making a change

Table 6.1: Modifiability tactics.

6.5.2 Usability Tactics

Tactics that were chosen to improve usability [15] is described in table 6.2.

Req.	Tactic	How	Why
U1	Support User Initiative	Cancel: Listen for the cancel request, terminate the command that has been cancelled	Support the user in either correcting errors or being more efficient

U2	Support System Initiative	Maintain Task Model: Determine the context of interaction, so that the system can have an idea of what the user is trying to accomplish, in order to assist the user.	Make interaction with the system more efficient and easy for the user
U3	Support System Initiative	Maintain System Model: Maintain an explicit model of the state of the form in order to be able to know what kind of values to expect in the form	In order to make sure that the user does not make any big mistakes while filling out the patient form

Table 6.2: Usability tactics.

6.5.3 Availability Tactics

Tactics that were chosen to improve availability [13] is described in table 6.3.

Req.	Tactic	How	Why
A1	Detect Faults	Self-test: When the connection between sensor and hub fails, the hub gets an error	In order to detect faults
	Recover From Faults	Retry: Saves the data that has not yet been sent and sends it when a connection is established	In order to recover from a fault

Table 6.3: Availability tactics.

6.6 Design Patterns

6.6.1 Factory Method Pattern

The Factory Method Pattern is a design pattern that makes it possible to instantiate objects without specifying the class. Factory Method makes the

system more easily customizable by making it easy to create new classes and integrate them into the system.[41]

The hub uses this pattern for creating instances of different sensors. This is to make it easy to support new sensors in the future. To add a new sensor one would simply need to create a new class for the new sensor and add that class to the factory.

6.7 Architectural Rationale

Choosing MVC, Layered pattern, and Factory Method pattern increases the modifiability of the system. Because of the nature of the system, which consist of a hub, database and a view, the team found MVC as a fitting architecture for the system. Implementing the layered pattern enables modification of the different parts of the system without interfering with all other parts. The team can also work on different layers at the same time without having to take any precautions in order to avoid breaking any other part of the system.

Chapter 7

System Design

7.1 General

This chapter describes the overall system design of the three main parts: the back-end, the front-end and the hub. It also describes tools used to develop the design over time.

7.2 Back-end

7.2.1 Database Design

The database design is shown in figure 7.1. The figure shows the different Django models and their relationships. The DjangoUser [36] entity is a standard model that comes with Django. It handles user accounts, groups, permissions and cookie-based user sessions. Each box is an entity that has attributes in the database, and the lines between them is how they are connected, usually by a foreign key.

While developing the database the back-end team made an ER diagram to get a better understanding of the database. This was also a tool used to come up with solutions of how the data flow could work in the finished system. The ER diagram went through several iterations as the database developed.

7.2.2 Application Programming Interface

API is a three letter acronym for Application Programming Interface, which denotes an interface of a software so that specific parts of it can be activated, run, from another software. This lets the web page access the data from the

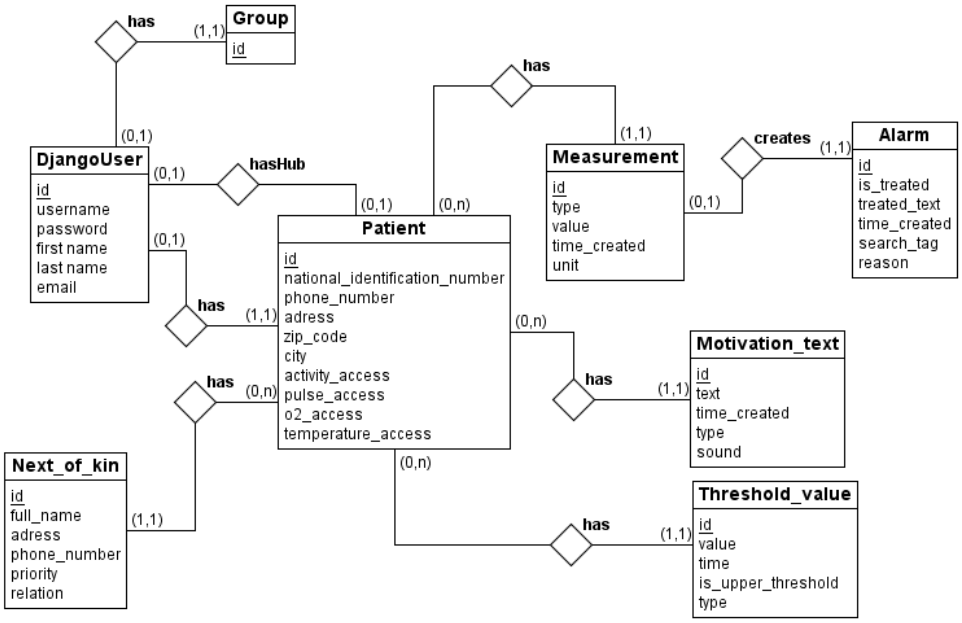

Figure 7.1: Final ER diagram for database

database. Django REST Framework creates the API endpoints to POST and GET JavaScript Object Notation (JSON) data.

7.3 Front-end

7.3.1 User-Centred Design

In the development of the systems UI the team chose to use the principles of ISO 9241-210 for user-centred design [40]. Cooperative design, or participatory design, will also be applied in order to involve the system users in the design process. Healthcare professionals is the most important user group. They do not have time for inefficient tools and might be reluctant to accept a system that does not conform to their expectations and demands. By involving the healthcare professionals in the design process, the UI will better reflect their needs. Another reason for focusing on the UI design process is that the system is heavily reliant on displaying information and making information accessible in an efficient and user-friendly manner. This makes the user interface development process key to the success of the entire system.

7.3.2 The Design Process

The design process started with the creation of a simple paper prototype. This prototype was then scanned into the prototyping tool POP [42]. The tool helped us turn our simple drawings into a wireframe that we presented to the customer in order to gather feedback. The customer had some suggestions for changes and improvements for the prototype. The user requested a bigger user interface, because the primary device for viewing it will be a PC. They also wanted tabbed interface for user data pages.

The second iteration of the prototyping process was to make a wireframe prototype using the tool Pidoco [43]. This prototype included the changes that had been requested by the customer after viewing the first prototype. The team made a video presenting the new prototype and this video was shared with the customer. The customer distributed the video within their organization and gathered feedback. Preferably some representatives from the team should have observed and interviewed the people who saw the film, but circumstances did not permit this. The feedback collected by the customer was presented to the team on a customer meeting.

The team continued to run iterations to gather more feedback. After receiving feedback on a video the changes were made and another video recorded. From this point the implementation functioned as the prototype.

By using dummy-data the front-end could be used to demonstrate suggested solutions. The front-end was presented at every customer meeting in order to gather as much feedback as possible.

7.3.3 Prototypes

The two major prototypes that were made during the prototyping process was the POP and Pidoco prototypes. A selection of screen shots from the POP prototype is shown in figure 7.2, 7.3, and 7.4, the full prototype can be accessed on the web[1]. Screen shots from the Pidoco prototype can be seen in figure 7.6, 7.5, and 7.7, the full prototype can be accessed on the web[2]

7.3.4 Designhjelpen

Designhjelpen [46] is a group of skilled and dedicated students from Industrial design at NTNU. Their workshops can include brainstorming, user testing and marketing of a product from a design viewpoint. It is possible to approach them with a request to facilitate a workshop. As the demand for workshops is high, they only pick the most interesting products. The team approached Designhjelpen early in the first sprint.

7.4 Hub

7.4.1 Hub and Angel Sensor

Using the Angel sensor makes it necessary for a hub at the location of the user to pull data from the sensor over Bluetooth as seen on figure 6.6. The hub runs a script which periodically pulls data from the Angel sensor and forwards the data to a central server.

7.4.2 Hub and Backup Sensor

When using the backup sensor, the architecture is slightly altered. As we can see from figure 6.7, the Withings sensor needs a mobile device running a Withings app. This app makes it possible to pull data from the backup sensor and automatically forwards it to the Withings server. The only way to obtain and use the data is to interact with the Withings API. The function

[1] `https://popapp.in/w/projects/5410435500988ce66306fbb4/mockups`

[2] `https://pidoco.com/rabbit/api/prototypes/124705/pages/page0001.xhtml?mode=sketchedArial&api_key=t4VPAflORKja4YhccyRM2DYPEM6m74pNP1e2MWaq`

Figure 7.2: Prototype of the log in screen in POP

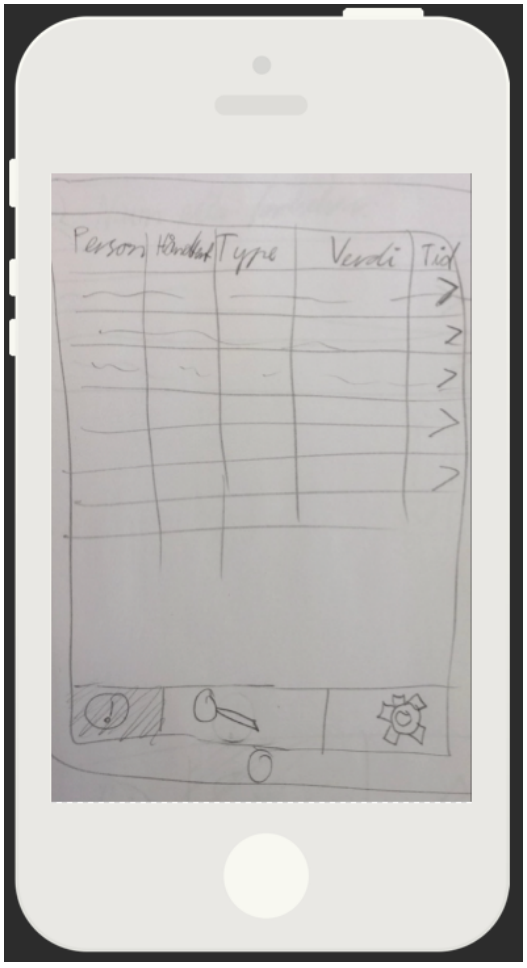

Figure 7.3: Prototype of the alarm screen in POP

Figure 7.4: Prototype of the user page screen in POP

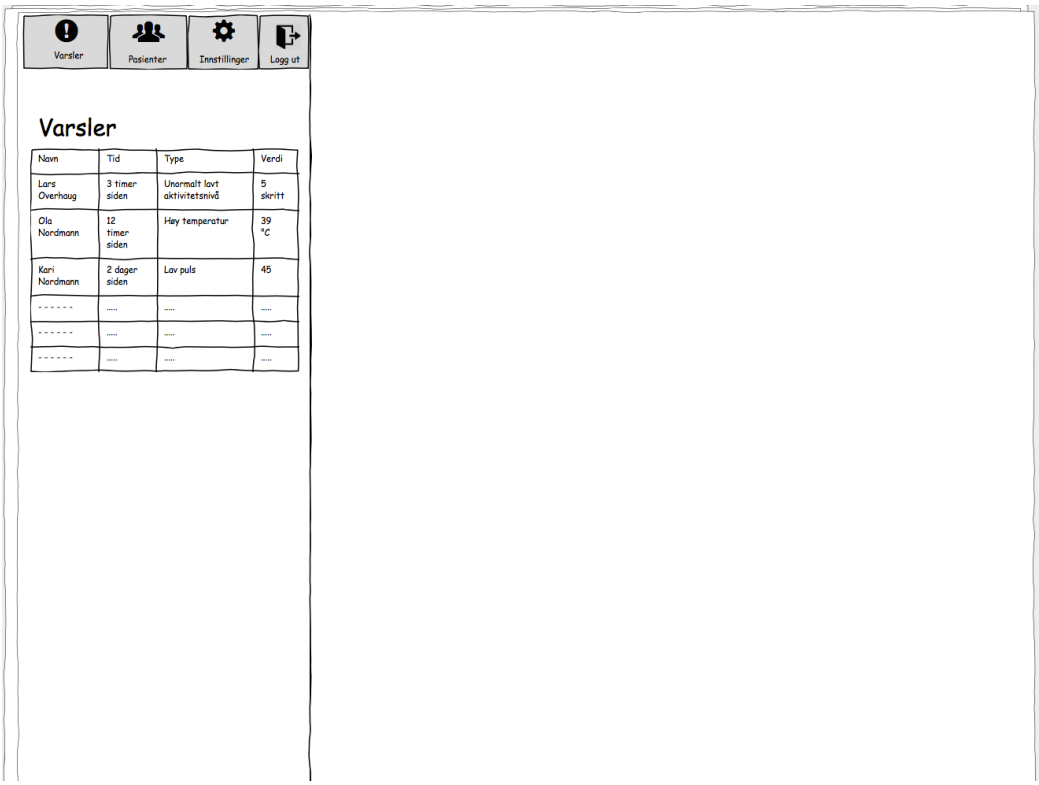

Figure 7.5: Prototype of the alarm screen in Pidoco

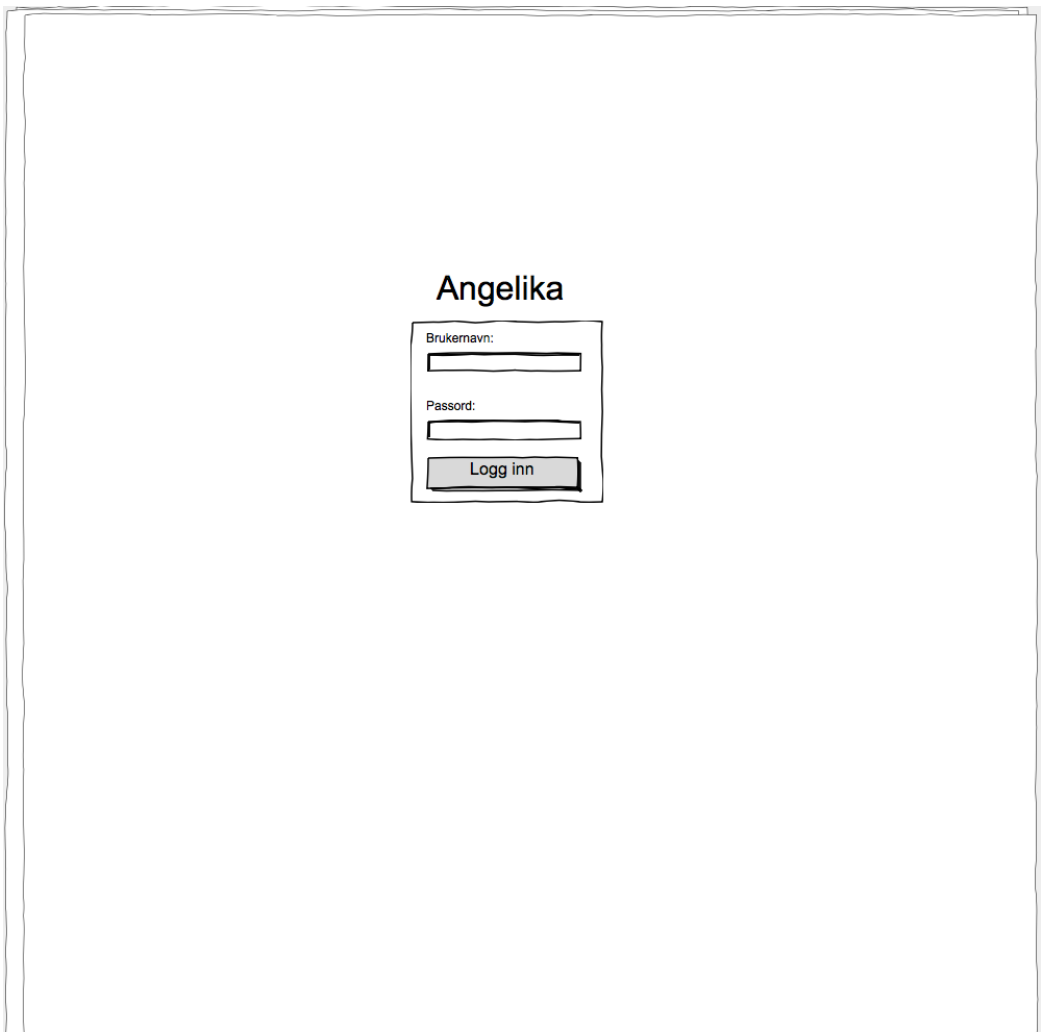

Figure 7.6: Prototype of the log in screen in Pidoco

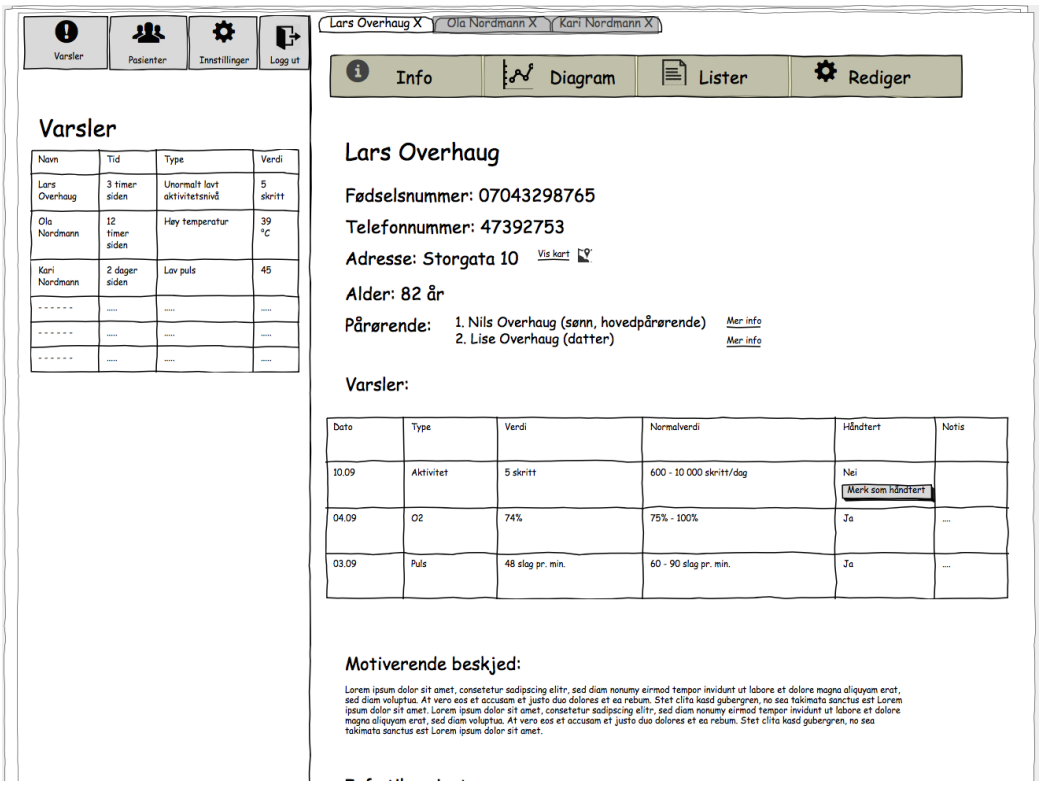

Figure 7.7: Prototype of the user page screen in Pidoco

of the hub is to pull data from the Withings API and post it to the central server.

As the hub might seem slightly redundant in this scenario, an approach to designing the system with the backup sensor is to remove the hub from the architecture and let the central server interact with the Withings API itself. This will probably be a better solution if the idea is to use the Withings sensor when the system is operational. However the Whitings sensor does not fulfil the open source requirements and also stores the data with a third party, it does not make sense to re-tailor the system for this sensor. The focus has been directed towards making it easy to switch to the Angel sensor at a later point in time. By including the hub in the system and writing very general scripts for pulling and posting data, the work needed to make a transition to a different sensor will be reduced significantly.

Part II

Sprints

Chapter 8

Sprint 1

8.1 General

This chapter describes the planning, execution, result and evaluation of the first sprint.

8.2 Sprint Planning

8.2.1 Duration

Sprint 1 began 9th of September and ended 26th of September, lasting 18 days.

8.2.2 Sprint Goal

The goal of the first sprint was to create the main foundation of the system: a database, a front-end solution, and an API to communicate between front-end and the database. The healthcare professional should be able to log in to the system with a username and password.

The rest of this period was used to create prototypes that were presented to the customer in order to collect feedback. Work was also done on selecting and learning programming languages, frameworks, tools and other technologies, and writing documentation.

8.2.3 Backlog

		Hours	
User story	Req. and description	Est.	Act.
US01	FR-F1: Actors need to be able to log in to the system through a web interface	**16**	**17**
	FR-F1-A: Create basic web interface	8	10
	FR-F1-B: Log-in screen	2	3
	FR-F1-C: Log out option	2	1
	FR-F1-D: Database with users table	4	3
US01	FR-F3: Access to the front-end interface needs to be password protected	**6**	**9**
	FR-F3-A: Password protect front-end	6	9
US04	FR-B3: The server must store information about the user	**9**	**4.5**
	FR-B3-A: Store name	1	0.5
	FR-B3-B: Store NID	1	0.5
	FR-B3-C: Store phone number	1	0.5
	FR-B3-D: Store restrictions of user	1	0.5
	FR-B3-E: Store motivational messages	1	0.5
	FR-B3-F: Store information messages	1	0.5
	FR-B3-G: Store thresh values	1	0.5
	FR-B3-H: Store address	1	0.5
	FR-B3-I: Store next of kin	1	0.5
US02	FR-B1: The monitored values must be stored in a database	**8**	**4**
	FR-B1-A: Store blood oxygen data in database	2	1
	FR-B1-B: Store heart rate data in database	2	1

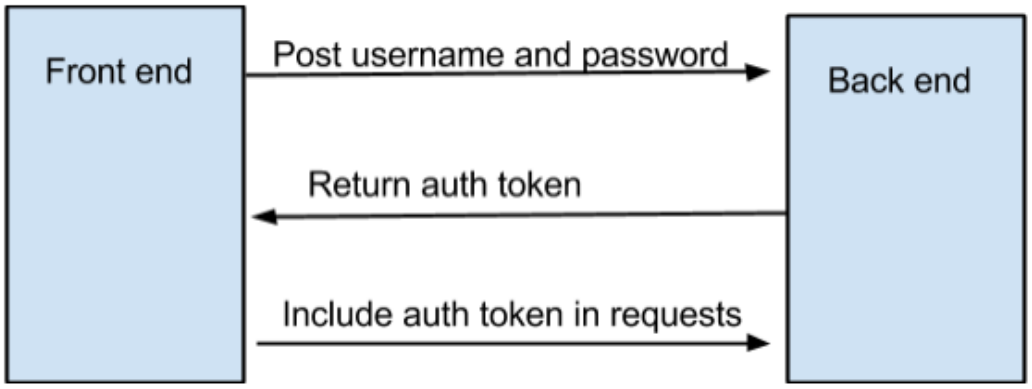

Figure 8.1: Login procedure over the API

		Hours	
User story	Req. and description	Est.	Act.
	FR-B1-C: Store activity data in database	2	1
	FR-B1-D: Store temperature data in database	2	1
	Total	**39**	**34.5**

Table 8.1: Sprint backlog, sprint 1

8.3 System Design

8.3.1 System Overview

The front-end development was done through a user-centred design process using multiple prototype iterations, as can be seen in section 7.3.1. After identifying some main requirements the team also started implementing the solution.

Figure 8.1 shows a diagram illustrating the login procedure over the API.

Figure 8.2 presents the structure of the database at the end of sprint 1.

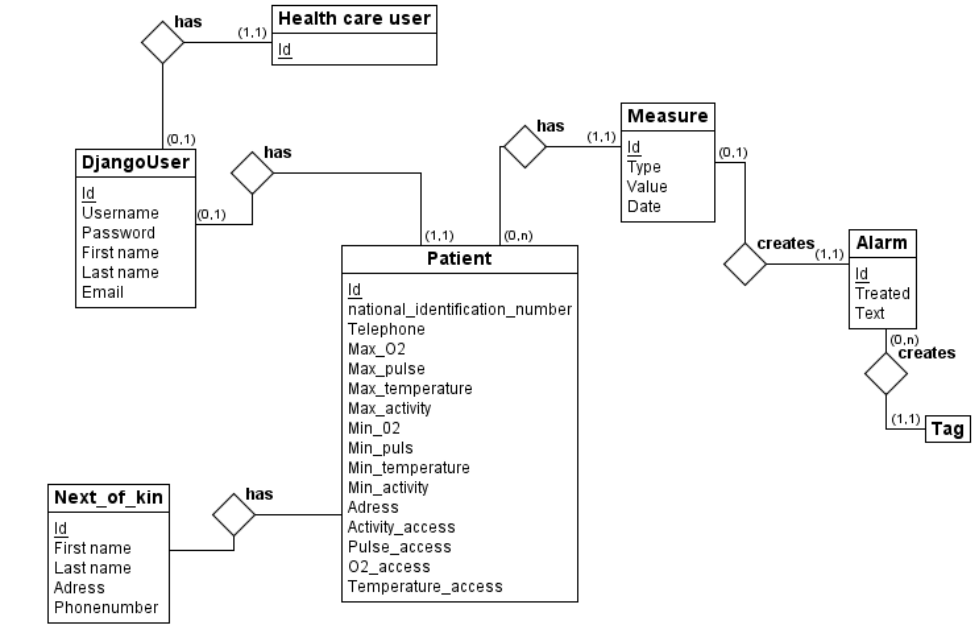

Figure 8.2: ER diagram at the end of Sprint 1

8.4 Implementation

8.4.1 Log In

Token-based authentication.

- Whenever an auth token is not included in the HTTP request to a view that has restricted access, the API returns HTTP 401 Unauthorized.
- Whenever the client attempts to log in with an invalid combination of username and password, the API returns HTTP 400 Bad Request, since an auth token could not be obtained from the given combination.
- Whenever the client attempts to log in with a valid combination, the API returns an auth token. When the client has obtained the token, it navigates from the log in page to the dashboard page.
- The client can include the obtained auth token in HTTP requests to the API to authorize itself so that it can get access.

8.4.2 Log Out

The client clears the auth token from memory, resets HTTP defaults, and navigates to the login page. No request is sent to the API during this process.

8.4.3 Store information about the user

The team created a model called "patient", which has a 1-1 relation to the Django user model. Django already has fields for first name, last name, username and password. This is a way to extend the built-in user model. Even if patient queries often require a join database operation, it is best practice according to the Django community. When a model is created in Django, a database migration is automatically created. This makes it easy for the team members to have the same database structure during the development of the system.

8.4.4 Menu

A simple bootstrap button group was included in the header. This would serve as a menu that could be used to switch between multiple pages.

8.4.5 Script to Send Email Containing the IP of the Hub

To be able to control the Hub using Secure Shell (SSH), its Internet Protocol (IP) must be known. A script that emails the IP of the Hub at startup was created. This will eliminate the need to use a screen and keyboard whenever interaction with the Hub is necessary.

8.5 Customer Feedback

During sprint 1, the team and the customer were still in the process of defining the scope and requirements of the project, so a lot of the customer feedback in this period was related to this.

One of the major decisions that was taken in sprint 1, was to use a Raspberry Pi instead of a mobile device as a hub (see sections 3.4 and 3.9.2). The team presented their opinion to the customer, and the customer agreed that a Raspberry Pi would be a better solution.

There was also a lot of focus on prototyping in this sprint. The team produced mockups of the user interface that were presented to the customer.

The customer was mostly satisfied with the team's suggestions, but also gave constructive feedback on how certain parts of the user interface should be created.

8.6 Testing

Since the team was still doing a lot of research, very little code was produced. As the team were still getting to know new technology, no tests were produced in this sprint as previously planned.

8.7 Sprint Evaluation

8.7.1 Review

The implementation went slowly in the first sprint, mainly because of lack of requirements. However, the team had an idea of what kind of system they wanted and therefore much time was spent on designing the interface. Although there was a lack of requirements, there were some requirements to work with. In the end of the sprint, the team managed to implement a log in function and began working on the API for the patient model. A group meeting to decide what kind of technology should be used was also held.

Figure 8.3 shows the burndown chart for the sprint.

8.7.2 Positive Experiences

- Good communication with customer.
- The team members are cooperating well with each other.

8.7.3 Negative Experiences

- Since many project details were partly undefined at the beginning of the sprint, it was difficult to start producing a product when it was unclear exactly what was to be produced. This meant that the team's productivity was limited by research and meetings with the customers to receive more answers before development could start.
- It was challenging to achieve the required 25 person hours per week. This was due to both lacking insight in what needed to be done and team members focusing on other courses in this period.

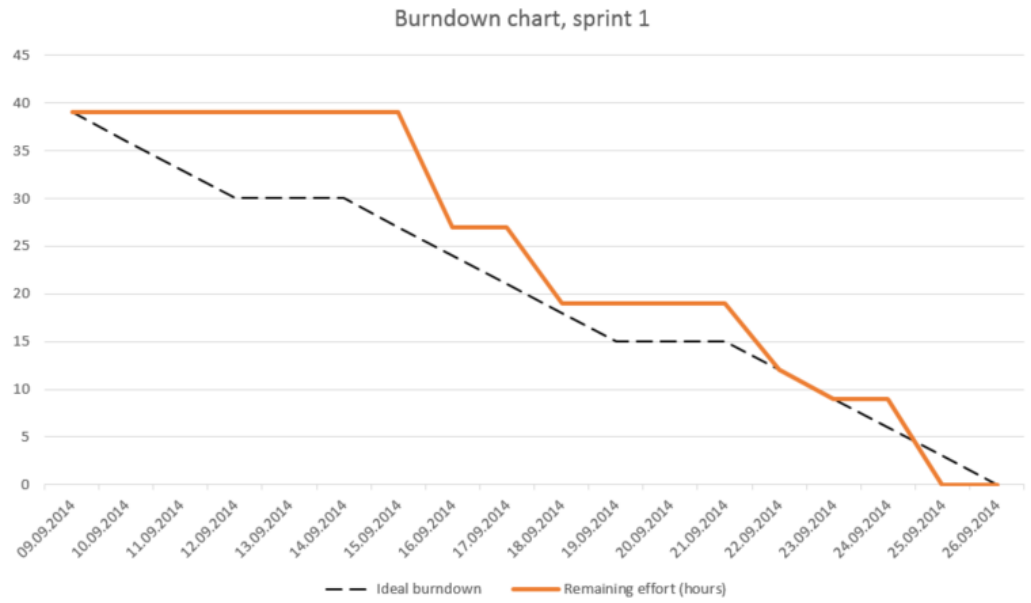

Figure 8.3: Burndown chart, sprint 1

- Little work was put into documentation, which resulted in a lot of sprint 1 lacking documentation, that would have to be completed in other sprints.

- Planned testing was dropped, due to lack of progress.

8.7.4 Planned Actions

Documentation and testing would need to be improved in the next sprint. The team planned to make templates for documentation and testing, and also put up a skeleton for the report.

8.7.5 Barriers

The major barrier of this sprint was that the team did not have access to any wearable sensor. This made it harder for the team members assigned to working on the sensor part of the system to find out how to get communication between the sensor and the hub to work.

In addition, at the start of this sprint the team did not have all the requirements for the system ready, as the customer wanted the team to investigate what possibilities a wearable sensor based system could offer. This

meant that the design and implementation phases had to start later than if the customer had presented the team with the requirements from the start.

Chapter 9

Sprint 2

9.1 General

This chapter describes the planning, execution, result and evaluation of the second sprint.

9.2 Sprint Planning

9.2.1 Duration

Sprint 2 began 29th September and ended 17th October, lasting 19 days.

9.2.2 Sprint Goal

The goal of the second sprint was to implement the core functionality. It was also needed to create additional prototypes in order to gather more information and feedback from the customer. The prototypes also needed to be more high level, showing more detail than earlier. On the hardware side, the hub should receive data from a sensor and the team should find and use a backup-sensor. This other sensor should be used until the Angel device arrive in order to start implementing sensor functionality for the system.

9.2.3 Backlog

Table 9.1 shows the backlog for sprint 2. Requirements/tasks that were postponed to the next sprint are written in italics, and the ones that have been completed in the first sprint have a strike through.

		Hours	
User story	Req. and description	Est.	Act.
US02	FR-S2: The sensor must communicate with a hub device using Bluetooth	**5**	**20**
	FR-S2-B: Make hub receive data from sensor	5	20
	FR-H3: Hub should be able to cache data for a certain period of time	**15**	**20**
	FR-H3-A: Make sure data is cached in hub if a connection cannot be established with the server	15	20
US11	FR-B2: The system must alert healthcare professionals when an abnormal value is registered	**35**	**26**
	FR-B2-A: Generating alarm on back-end	10	8
	FR-B2-B: Receiving alarm on front-end	5	3
	FR-B2-C: Front-end for handling of alarm	10	7
	FR-B2-D: Back-end for handling of alarm	10	8
US04	FR-F4: Front-end must display a list of registered users	**8**	**10**
	FR-F4-A: Front-end for list view	3	2
	FR-F4-B: Back-end for list view	3	5
	FR-F4-C: Database with user table	2	3
US05	FR-F5: Front-end must allow searching for users by name, date of birth or social security number	**2**	**1**
	FR-F5-A: Front-end search functionality	2	1

		Hours	
User story	Req. and description	Est.	Act.
US06	FR-F6: Front-end needs to allow healthcare professionals to configure threshold values for users	**9**	**5**
	FR-F6-A: Front-end for threshold value settings	3	2
	FR-F6-B: Back-end for threshold value settings	3	2
	FR-F6-C: The front-end interface should validate a new threshold value	3	1
US07	FR-F7: Front-end needs to allow healthcare professionals to configure what information is visible for the user	**6**	**5**
	FR-F7-A: Front-end for user access settings	3	2
	FR-F7-B: Back-end for user access settings	3	3
US08	FR-F8: Front-end needs to display data as graphs	**20**	**24**
	FR-F8-A: Front-end for activity graph	5	6
	FR-F8-B: Front-end for blood oxygen graph	5	6
	FR-F8-C: Front-end for heart rate graph	5	6
	FR-F8-D: Front-end for temperature graph	5	6
US09	FR-F10: Front-end needs to allow healthcare professionals to change the time span of a graph	**4**	**6**
	FR-F10-A: Front-end for selecting graph time span	4	6

		Hours	
User story	Req. and description	Est.	Act.
US12	FR-F11: Front-end needs to present a list of unhandled alarms	**5**	**5**
	FR-F11-A: Front-end for alarm list	2	2
	FR-F11-B: Back-end for alarm list	3	3
US14	FR-F12: Front-end needs to have a tab-based interface	**15**	**20**
	FR-F12-A: Front-end for tab-based interface	15	20
US18	FR-F16: The front-end interface must display a list of next of kin for the users, with relations	**6**	**6**
	FR-F16-A: Front-end for next of kin	3	4
	FR-F16-B: Back-end for next of kin	3	2
US20	FR-F18: The system should store previous motivational/informative messages for users, and the front-end interface for the users should display them	**10**	**8**
	FR-F18-A: Front-end for motivational/informative messages	5	3
	FR-F18-B: Back-end for motivational/informative messages	5	5
US21	FR-F19: The front-end interface for users should have the ability to notify the healthcare professionals that the user would like to be called	**1**	**1**
	FR-F19-A: Add 'call me' button on user dashboard	1	1

User story	Req. and description	Hours Est.	Hours Act.
US22	FR-F20: The system should store previous threshold values, and display them on the front-end interface	**11**	**12**
	FR-F20-A: Display previous threshold values on the graphs	6	7
	FR-F20-B: Back-end should store old normal values	5	5
	Total	**152**	**169**

Table 9.1: Sprint backlog, sprint 2.

9.3 System Design

9.3.1 Hub

The requirement FR-S2 states that the sensor must communicate with the hub using Bluetooth. In anticipation of the Angel sensor, a backup sensor was acquired in the form of a Withings Pulse O_2. The Withings sensor uses Bluetooth, but it can not send the data directly to the hub. The data must first be sent to the Withings server. It was still unclear at the time whether the Angel sensor would arrive in time, so the focus was to make the Withings sensor communicate with the hub.

9.3.2 Back-end

Figure 9.1 presents the structure of the database at the end of sprint 2

9.4 Implementation

9.4.1 Receive Data from Sensor

Withings uses an authentication system called OAuth, which had to be used to receive the data from their servers. A library called python-withings was

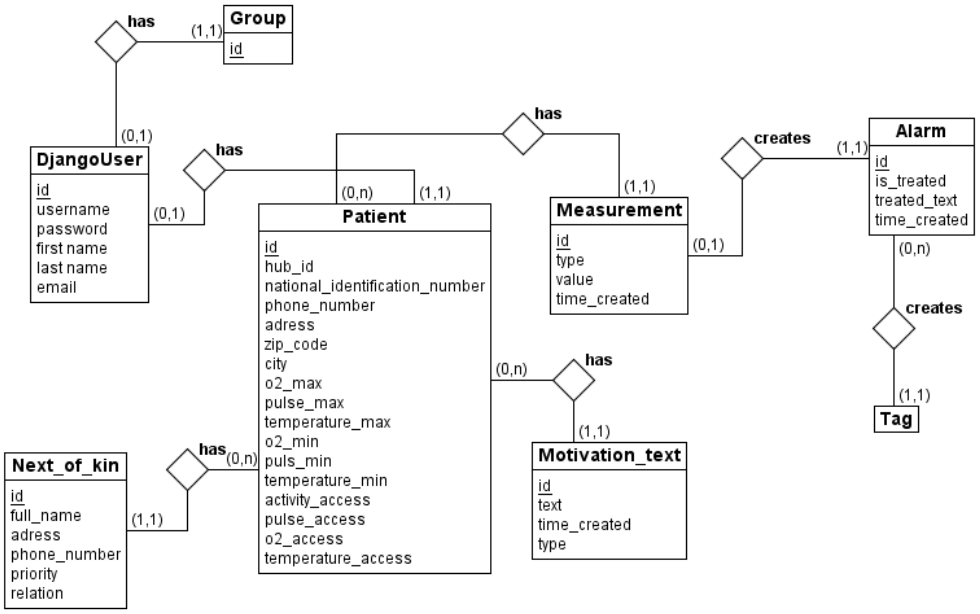

Figure 9.1: ER diagram at the end of Sprint 2.

used to communicate with their API and get the sensor data. This library did not support all the values of the Withings Pulse O_2 sensor, so the library was extended to include these values.

Since the sensor needed to be synced with a phone to get the data to the Withings server and further onto the hub, the hub would not always receive updated data when requesting data from the Withings API. This led to some problems when caching data, and making sure the database was updated.

9.4.2 Cache Data on Hub

The caching of the data on the hub seemed straight forward. There were however some problems while caching the data from the Withings server. It was possible to receive new data from the Withings server from the day before, even if the hub had tried to receive data earlier that day, because the sensor was not synced with the phone.

Considering this, the following process describes the caching of the data on the hub.

1. The hub asks the Withings server for data from the last update of the sensor to the current time.
2. The received data is checked against the cache to check for duplicate measurements.
3. All new measurements are saved in the cache as a file with a time stamp of the moment it is saved.
4. The hub saves the time stamp of the newest measurement as the latest update of the sensor.

9.4.3 Generate Alarm for Anomalies

When a measurement from a sensor is received, it is checked against the current threshold values for the user. O_2, pulse and temperature data are the only data types that can potentially generate alarms. O_2 can only generate alarm if the value is too low, not too high. If there is already an unhandled alarm of this type for this user, a new alarm is not created. This is done to prevent spamming multiple alarms for the same incident.

9.4.4 Display Registered Users

The team created an API endpoint for users. To make it easy to follow REST framework standards, a model view set was extended. This view set

provides an endpoint for a list of users and an endpoint for details about a specific user. To show a list of users on the front-end, angular uses the HTTP module to call the URL for the user list endpoint. If the client is authorized as health professional, the API returns JSON data with a list of users. When the front-end receives this data, it is rendered in the angular template.

9.4.5 Search for Users

Angular's filter function is used for searching in the list of users. This gives us an easy way to filter by name, national identification number, date of birth and age. The search is done on the client. This is memory intensive and slow for massive amounts of data, but really quick for small amounts of data. As the customer has said that the number of users will be around 4000, the team decided that a front-end based search was a good solution.

9.4.6 Configure Threshold Values

Every user has a list of threshold values. These threshold values are shown as coloured areas on the graphs. However, when it comes to configuring threshold values, there is a user object that exposes only the newest threshold value of each type. These values are shown in the user form. Whenever that form is saved, the front-end checks which threshold values have been changed, and shows a modal that presents the changes, if any. The healthcare professional must confirm these changes to be able to save them. Whenever that user object is posted or patched to the API, the back-end checks which threshold value have been changed. For each changed threshold value, a new threshold value row with current time is created.

9.4.7 Configure Information Visible for Users

In the back-end there is a boolean field for each data type, O_2, heart rate, temperature, activity. This data is shown as a set of check boxes in the user form.

9.4.8 Display Data as Graphs

The API provides a graph data endpoint. This endpoint provides data filtered by patient and data type. The JSON object includes three series: Lower threshold values, upper threshold values and measurements. Each object in a series includes an x attribute which is Uniplexed Information and Computing Service (UNIX) time, (Coordinated Universal Time (UTC)) and a y

attribute which is the value for that data point. Additionally, a measurement object in the measurement series may or may not include an alarm object. When a patient is opened in the front-end, the four different kinds of graph data is requested. When it is received from the API, the data is processed to find the span of the y axis and to determine the coloured areas for the thresholds. The Highcharts library takes care of drawing the chart based on the data. Normal measurement points are drawn in blue, points that have a treated alarm are drawn in red, while points that have an untreated alarm show a clickable alarm icon. When this icon is clicked, a modal for that alarm appears.

9.4.9 Change Timespan of Graphs

There is a bootstrap button group with buttons for several spans: Day, week, month, year. When a button in the button group is clicked, the span of the x-axis is set accordingly. Highcharts takes care of drawing the correct range.

9.4.10 Display Alarms

A model viewset was used in the back-end. The endpoint for the list of alarms should serialize information about the user and the measurement. To achieve this in an efficient manner, Django's select_related function was used. This creates a complex join-based Standard Query Language (SQL) query behind the scenes, rather than having one or more database queries for each alarm row, which is inefficient for large sets of data. The front-end simply loads data from this endpoint and shows it in a list.

9.4.11 Display a Tabbed Interface

Golden Layout is an advanced layout manager written in JavaScript. It is relatively new (released in October 2014) so it had a few bugs in the first few versions. The lead developer has contributed to the open source project in order to reduce the amount of bugs, and improve the usability. Among the features in Golden Layout is a tabbed layout. There were three challenges when implementing Golden Layout:

1. The style did not look anything like the bootstrap theme that was used elsewhere. This was resolved by rewriting the CSS of the default light theme of Golden Layout.

2. It was hard to integrate Golden Layout components with templates, controllers and scopes in AngularJS. This problem was dealt with by

using ng-init (for passing variables to new components), ng-include (for loading templates) and Golden Layout's event Emitter to pass layout-related events between AngularJS scopes and Golden Layout.

3. Bootstrap's responsive columns did not work nicely with Golden Layout's split view. At first, this seemed like a very hard issue. After a lot of research about how Bootstrap's responsive grid system works and how Golden Layout handles component sizes, the lead developer ended up setting a custom CSS class on each component based on its size. These custom CSS classes would override some of Bootstrap's default styling behaviour for responsive grids.

Bootstrap also provides a tabbed interface. The team has used both Golden Layout tabs and bootstrap tabs. They look the same, but the upper tabs are draggable Golden Layout tabs.

9.5 Sprint Testing

In this sprint, the team started writing tests for the code in the back-end. It was important that all new features written also had test coverage. The tests that were produced was continuously run before committing new code, and the tests had to be a success before committing. If any bugs were found, they would have to be fixed before proceeding implementing missing features.

Since the tests were required to be a success, no larger test evaluation was done at the end of the sprint. This was not ideal, but due to lack of experience in testing and lack of features, the testing was not so extensive at the end of sprint 2.

Front-end testing was postponed to sprint 3.

9.6 Customer Feedback

The customer gave a lot of feedback in this sprint which resulted in new requirements for the project, see section 4.4.2. They approved the choice of front-end layout and liked how it looked, but some of the text needed rephrasing. It was made clear that the customer wanted to have a functioning product at the end of the project.

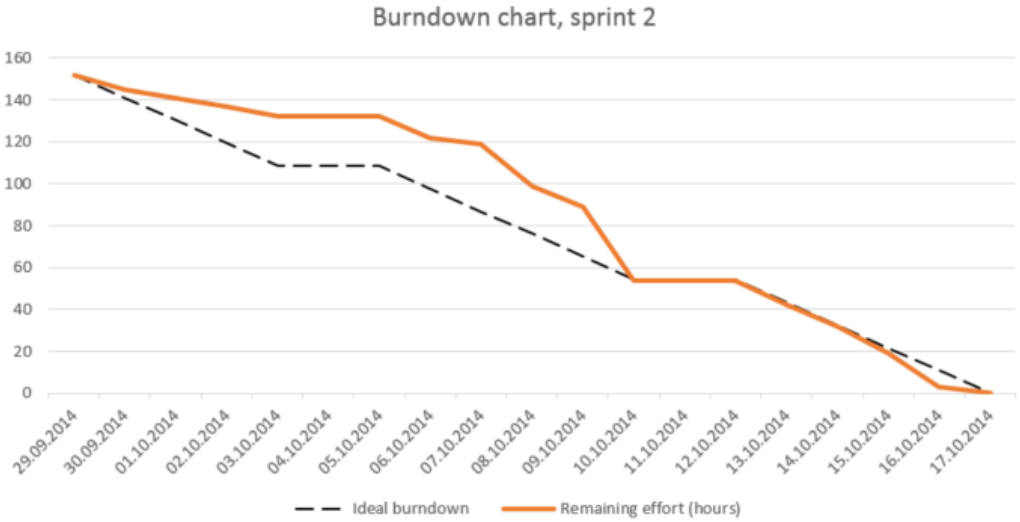

Figure 9.2: Burn down chart, sprint 2

9.7 Sprint Evaluation

9.7.1 Review

During this sprint only one meeting with the customer was held, however the meeting was very efficient. The customers gave answers to all the questions that were asked, and new requirements were determined.

The communication between back-end and front-end were focused on in this sprint. At the end of the sprint the team were able to complete the implementation of the API for patient, measurement, motivation text and next of kin. However, the team felt progress was slow. It was during this sprint the team members realized how far behind they were in documentation. Many team members were unaware of the detailed requirements needed in the report and documentation was lacking.

Figure 9.2 shows the burn down chart for the sprint.

9.7.2 Positive Experiences

- The backup-sensor was easy to implement.
- The front-end had good progress.
- The main functionalities were implemented.

- Dividing the team in three sub-teams (back-end/front-end/hub) was effective.

9.7.3 Negative Experiences

- During this sprint only one meeting with the customer was held. The team had many questions, and all of them had to be answered during that one meeting
- The backup sensor was worse than anticipated. It did not automatically measure pulse nor O_2, which was not stated anywhere on their web page.
- The team did not use Scrum optimally, the Scrum meetings were lacking and the documentation was not where it should have been.
- Because of the customer lacking technical insight it was hard to specify requirements.

9.7.4 Planned Actions

During sprint two, the team realized that they had been spending too little time on the project. The team leader decided to be more strict and everyone agreed that they should spend more time on the project. Some team members were allocated to work on the report.

9.7.5 Barriers

Because of lack of regular communication with the customer at the end of the, it was hard for the team to update their requirements as often as needed. Tasks that needed clarification with the customer were put on hold. The biggest barrier however was the fact that the back-up sensor was limited compared to the original sensor. The back-up sensor did not fulfil the requirements of the customer, and this would mean that an implementation with this sensor is not a sensor the customer can use in a real deployment. In the end of the sprint the team also got a message that said that the sensor was postponed, which meant that it would come too late and the backup sensor would therefore be used for the presentation.

Chapter 10

Sprint 3

10.1 General

This chapter describes the planning, execution, result and evaluation of the third and final sprint.

10.2 Sprint Planning

This was the last sprint, as a consequence, adding new requirements at this point would have to be considered carefully. After having done little work on documentation in earlier sprints the team now had to shift the focus to documentation.

10.2.1 Duration

Sprint 3 began 20th of October and ended 14th of November, lasting 26 days.

10.2.2 Sprint Goal

The goals of sprint 3 was to complete the system to a level where it would have full functionality. Also the report should be nearing completion. There was still a few days after the sprint available to finalize the report, but all major sections should be written.

10.2.3 Backlog

Table 10.1 shows the backlog for sprint 3. Requirements/tasks that were not implemented in this sprint, either because there was not enough time to complete them, or because the requirements from the customer changed (see section 4.4.3), are written in italics, and the ones that have been completed in previous sprints have a strikethrough.

		Hours	
User story	Req. and description	Est.	Act.
US02	FR-S4: The sensor should send data to hub automatically	**24**	**30**
	FR-S4-A: Make sensor send data to hub when new data is recorded	24	30
US01	FR-H1 Hub should send data to server when new data is received by sensor	**28**	**24**
	FR-H1-A: Make hub send data to server when new data is received from sensor	28	24
US32	FR-H4: Hub should be able to start and resume work after a power outage	**10**	**2**
	FR-H4-A: Make sure hub starts and continues work automatically after a power outage	10	2
US02	FR-B1: The monitored values must be stored in a database	**16**	**19**
	FR-B1-E: Store data from the hub in database	16	19
US28	FR-B4: Consecutive abnormal measurements of the same type should not generate more than one alarm	**4**	**5**

		Hours	
User story	Req. and description	Est.	Act.
	FR-B4-A: Create only one alarm per incident	4	5
US10	FR-F2: Web page should display historical data for monitored values to the user	**8**	**12**
	FR-F2-A: Code for getting all data about a user from database to front-end	8	12
US13	FR-F9: Front-end needs to display data as a table	**10**	**8**
	FR-F9-A: Front-end for list view	10	8
US12	FR-F11: Front-end needs to present a list of unhandled alarms	**5**	**4**
	FR-F-C: The alarm tab should show the number of unhandled alarms	3	2
	FR-F-D: There should be an option to only show unhandled alarms in the alarms list	2	2
US15	FR-F13: Front-end needs to have a customized, mobile-friendly view for when a user is logged in	**10**	**9**
	FR-F13-A: Customized front-end view for users	5	6
	FR-F13-B: Make view mobile-friendly	5	3
US16	FR-F14: Front-end needs to allow changing of global system settings	**6**	**8**
	FR-F14-A: Front-end for global system settings	3	4

		Hours	
User story	Req. and description	Est.	Act.
	FR-F14-B: Back-end for global system settings	3	4
US17	FR-F15: Front-end needs to use HTTPS	**16**	**24**
	FR-F15-A: Use HTTPS	16	24
US19	FR-F17: The front-end interface for users should make it possible to give voice messages to the users	**11**	**6**
	FR-F17-A: Front-end implementation of voice message for healthcare professionals	5	2
	FR-F17-B: Front-end implementation of voice message for users	3	2
	FR-F17-C: Back-end storing of voice messages	3	2
US21	FR-F19: The front-end interface for users should have the ability to notify the healthcare professionals that the user would like to be called	**4**	**3**
	FR-F19-B: Handle call request on back-end	2	1
	FR-F19-C: Display call request in alarm list	2	2
US23	FR-F21: The front-end interface should allow healthcare professionals to choose what kind of information they want to see from each user	**5**	**5**
	FR-F21-A: Settings for changing the visible information	5	5

		Hours	
User story	Req. and description	Est.	Act.
US24	FR-F22: The front-end interface should have a form for adding new users to the system, with standard values pre-set in the form.	**5**	**4**
	FR-F22-A: Front-end form for adding new users	5	4
US25	FR-F23: The front-end interface should highlight abnormal values in the list view	**1**	**1**
	FR-F23-A: Front end for highlighting of abnormal values	1	1
US26	FR-F24: The front-end interface should play a sound when new alarms are received	**5**	**7**
	FR-F24-A: Implementation of notification sound for alarms	4	6
	FR-F24-B: Setting for muting of notification sound	1	1
US27	FR-F25: Graphs in the front-end interface should display handled alarms differently than unhandled alarms	**2**	**1**
	FR-F25-A: Give handled alarms a different icon than unhandled alarms	2	1
US29	FR-F26: Graphs in the front-end interface should update automatically with regular intervals to get live data	**3**	**4**
	FR-F26-A: Update graphs automatically	3	4

		Hours	
User story	Req. and description	Est.	Act.
US30	FR-F27: The front-end interface should display a message instead of no values if there are no data	**5**	**3**
	FR-F27-A: Show a message instead of an empty graph if there are no graph data	1	1
	FR-F27-B: Show a message instead of an empty list if there are no list data	1	0.5
	FR-F27-C: Show a message instead of an empty alarm list if there are no alarms	1	0.5
	FR-F27-D: Show a message instead of an empty motivational texts list if there are no motivational texts	1	0.5
	FR-F27-E: Show a message instead of an empty informative texts list if there are no informative texts	1	0.5
US31	FR-F28: Phone numbers in the front-end interface should be clickable	**1**	**1**
	FR-F28-A: Make phone numbers in front-end clickable	1	1
	Total	179	180

Table 10.1: Sprint backlog, sprint 3

10.3 System Design

10.3.1 System Overview

After this final sprint the system had complete functionality as specified by the requirements in the backlog 10.1. The finished system is described in greater detail in section 11.2.1

10.4 Implementation

10.4.1 Sensor Should Send Data to Hub Automatically

As the focus was changed from the Angel sensor to the backup sensor, the implementation details regarding how the sensor should send data changed accordingly. As long as the sensor had a mobile device with Bluetooth activated nearby running the Withings app it would synchronize automatically with the mobile device. The mobile device would again synchronize automatically with the Withings servers. This was functionality already made by Withings. For the data to arrive at the hub, it was necessary to use the Withings API.

The hub executed HTTP GET requests to the Withings API at specified intervals. If there was any new data in the response, this data was cached in the hub.

10.4.2 Hub Should Send Data to Server When Received by Sensor

A thread in the hub pulls data from the Whitings API and places it in the cache. Another thread checks the cache at given intervals for new data. This was done to avoid duplicated data. It is important to note that this interval was different than the interval mentioned in section 10.4.1. The reason behind having a different interval for when the sensor is checked for new data and the hub posting the data to the API is that the traffic between the hub and the API should be limited. If there are 4000 users, all with their own hub, the traffic into the central server would be substantial unless limited. If the cache contained data that was not previously sent, it would send it. Since the API was a REST API, the easiest way to send data from the hub was to let it execute HTTP POST requests to the API, posting JSON files. The JSON files comprise a list of measurements and hub-id. There is no personal information about the user in the data sent from the hub. The hub-user relationship is specified server-side.

10.4.3 Store Monitored Values in a Database

When data is received at the server after being sent from the hub, the backend iterates over the list of measurements posted. For every measurement, it creates a Measurement object containing the information received and links the included hub-id with user-id to map the measurement to a specific user.

10.4.4 Display Historical Data for Monitored Values to the User

The API provides a graph data endpoint for current user. This endpoint gives a simple list of measurements (no alarms are specified) from the last 7 days. If the user tries to access data he or she is not allowed to see, HTTP 403 (Permission Denied) is returned. Highcharts is used to visualize the data in front-end.

10.4.5 Display Data as a Table Front-end

The list view uses the same data as the charts, but the ordering should be reversed in the list view. To achieve this, Angular's orderBy filter was used. To use the real estate efficiently, all four lists are shown beside each other, as bootstrap columns. If the display is too narrow for this, the lists are placed below each other. By default, only the 15 newest measurements are shown in a list. This is done to have a manageable amount of data to scroll through when viewing the UI on a tablet. At the end of the list, you will find a button for that can be clicked to expand the list with up to 200 more measurements. For each measurement that has an alarm, the background for the value cell becomes red.

10.4.6 Customized, Mobile-Friendly View for When a User is Logged In

In order to make things easy to read for visually impaired users, the text size is set to a relatively large size. As bootstrap is responsive and mobile-friendly out of the box, basically, all that is needed is adding a viewport meta tag that sets the width of the page to the width of the device.

10.4.7 Change Global System Settings in Front-end

The two global settings are sound on/off and show only untreated alarms. These are shown as button button groups. The active setting is highlighted with a darker background colour. When the mouse is hovered over the button group, a pop-over that explains the setting appears on the right. When a setting is selected, it is saved in the browser's localStorage.

10.4.8 Use HTTPS

If HTTP (port 80) is used, a man in the middle could obtain the client's auth token (and other data) by sniffing data packets, log in as the user for that token and potentially get access to secret data. It is important to avoid this serious security vulnerability, and it can be done by using HTTPS. This technology makes sure that the connection is encrypted from end to end, so that the data packets cannot be sniffed by a man in the middle. The team lends a linux server from NTNU. This server was initially set up to a sub domain of ntnu.no. In order to get an Secure Sockets Layer (SSL) sertificate for our system, a custom domain (not a sub domain of ntnu.no) was needed. Therefore, the domain angelika.care was bought, along with a Comodo PositiveSSL certificate. In order to be able to prove the fact that the team owns angelika.care, the email address postmaster@angelika.care was set up. When the keys and certificates were generated and obtained, nginx [48] on the server was set up to serve the API over HTTPS (port 443) on api.angelika.care. The API was made accessible only through HTTPS and not HTTP.

10.4.9 Audio Recordings

Before developing the sound recorder feature, the lead developer had to choice among two technologies: Flash and HTML5.

Keys points for HTML5:

- Open standard.
- Works on mobile devices.
- Future-oriented.

Key points for Flash:

- Closed standard.
- Not compatible with mobile devices.
- Apple claims that it is dying technology [1].

These key points made it easy to see that HTML5 was the obvious choice. The lead developer found an open source library called Recordmp3js. This library uses getUserMedia and Web Audio API for getting sound input from

[1] `http://www.apple.com/hotnews/thoughts-on-flash/`

the healthcare professional's microphone and record it. getUserMedia is implemented in many browsers, including Firefox, Google Chrome and Opera. After the sound is recorded, the raw sound data is converted to Moving Picture Experts Group, Audio Layer III (MP3) with a JavaScript library called libmp3lame. Since Recordmp3js is not compatible with AngularJS, the lead developer had to develop an Angular service module that wrapped the functionality of Recordmp3js in order to integrate it. To transfer the sound data to the server, the MP3 data is encoded to base64 and included in the patient JSON object. When the API receives this base64 data, it is decoded and saved as an MP3 file. Afterwards, the API provides absolute URLs to these MP3 files. The MP3 files are playable in the web application by using the standard audio HTML tag, which is widely adopted.

10.4.10 Optimizing front-end for mobile devices

The team worked on making the front-end of the system scale to any screen dimensions, making it compatible with any mobile device. This included front-end view for both users and healthcare professionals. The latter proved most challenging, as this view uses the layout manager Golden Layout, which does not have touch support in its current version. This meant that the lead programmer had to contribute to the Golden Layout project and, in some cases, create work-arounds in order to make it work. Figure 10.1 and figure 10.2 shows the front-end view for a healthcare professional on an iPhone 6 and an iPad 4 respectively.

10.5 Sprint Testing

10.5.1 Types of Tests

A performance test was performed this sprint. It was an important step to ensure that the system performance was at an acceptable level. For more information see A.2.

Front end was also tested for compatibility with different browsers and devices, see appendix B for more information.

Testing was done continuously on the back-end during development. The UAT towards the end of the sprint would be a test of the whole system, and an important step to verifying that the system covered all the requirements.

10.5.2 Test Results

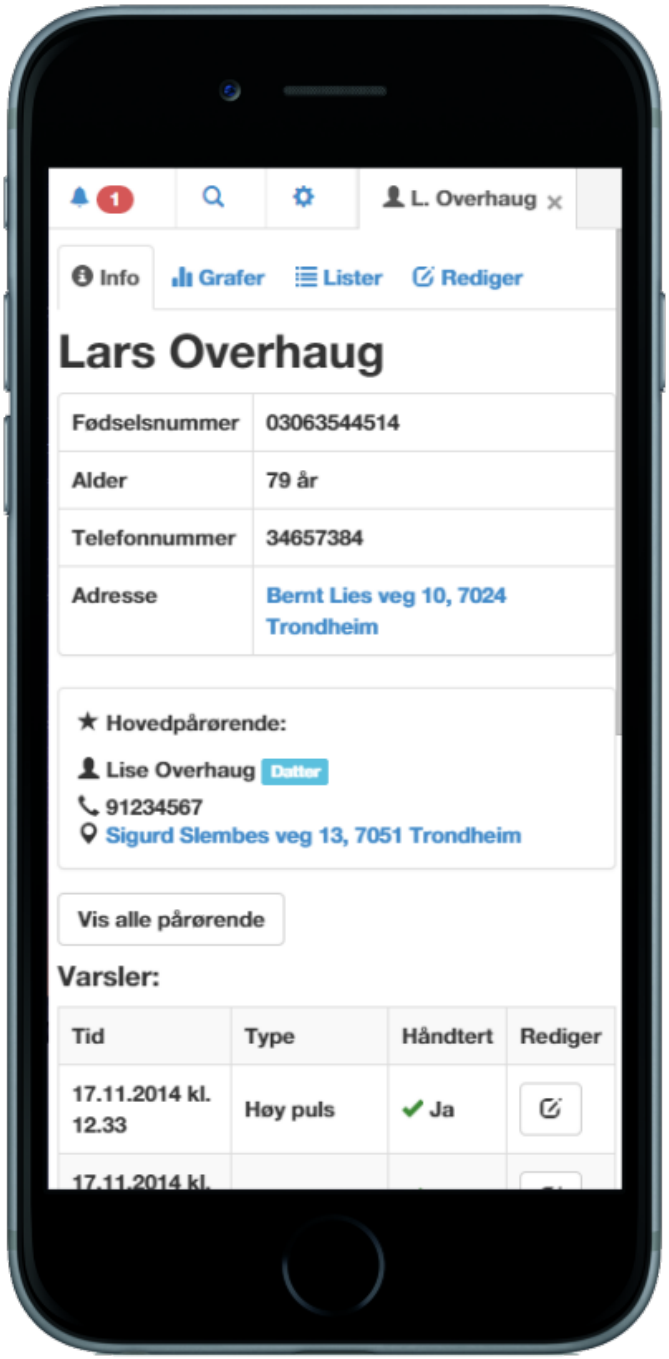

Figure 10.1: Front-end on an iPhone 6

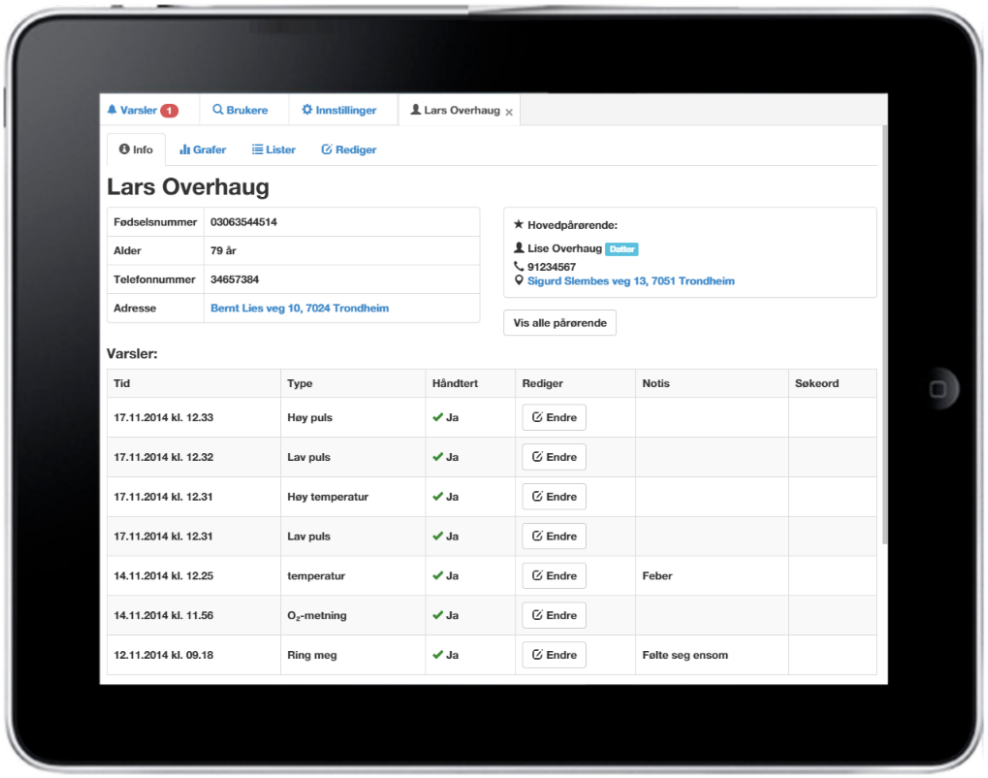

Figure 10.2: Front-end on an iPad 4

Test Case ID: TID01

Test Case Name: Module Patient Test

Tester: Sigurd Sandve

Description: Testing of all the interactions of the patient model

Test unit: PermissionTests

Authorization is tested for wrong behaviour, by trying to retrieve the list of patients with different permissions.

Test name	Exp. Res.	Act. Res.	Pass/fail
unauthorized	401	401	Pass
authorized	201	201	Pass
no permission	403	201	Pass

Test unit: PatchTest

Tests functionality the healthcare user should be able to perform after logging in.

Test name	Exp. Res.	Act. Res.	Pass/fail
activity access	True	True	Pass
add next of kin	-	-	Pass
update next of kin	-	-	Pass
reorder next of kin	-	-	Pass
remove next of kin address	-	-	Pass
update, reorder, remove, add nok	-	-	Pass
add motivation text	-	-	Pass
update motivation text	-	-	Pass
remove motivation text	-	-	Pass
add information text	-	-	Pass

update information text	-	-	Pass
remove information text	-	-	Pass
add, remove info and motiv text	-	-	Pass

Test unit: GetGraphDataTests

Tests the graph data recieved from the sensor, and interaction with it.

Test name	Exp. Res.	Act. Res.	Pass/fail
patient graph data endpoint	-	-	Pass
no permission	403	403	Pass
threshold values	-	-	Pass
graph data measurements	-	-	Pass
graph data alarms	-	-	Pass

Test unit: CurrentPatientTests

Tests the ”call me” functionality avaliable to the user in the user view

Test name	Exp. Res.	Act. Res.	Pass/fail
call me request	-	-	Pass
call me request repeatedly	-	-	Pass

Test unit: PostTests

Tests the creation of a new user.

Test name	Exp. Res.	Act. Res.	Pass/fail
create patient	-	-	Pass
create patient minimum data	-	-	Pass
unique identification number	-	-	Pass

Test unit: UsernameHelperTests

Tests the automatic generation of usernames.

Test name	Exp. Res.	Act. Res.	Pass/fail
generate username	-	-	Pass
from weird characters	-	-	Pass

Table 10.2: Sprint 3 patient testing

Test Case ID: TID02

Test Case Name: Module Motivation- and Information Text Test

Tester: Sigurd Sandve

Description: Testing of all the interaction of the motivation model

Test unit: TestMotivation

Tests the creation of motivation and information text

Test name	Exp. Res.	Act. Res.	Pass/fail
add motivation text	-	-	Pass
current information text	-	-	Pass
current motivation text	-	-	Pass

Test unit: TestCronjobs

Tests if the cronjob deletes old motivation texts

Test name	Exp. Res.	Act. Res.	Pass/fail
delete old motivation text	-	-	Pass

Table 10.3: Sprint 3 motivation- and information text testing

Test Case ID: TID03

Test Case Name: Module Alarm Test

Tester: Sigurd Sandve

Description: Testing of all the interaction of the alarm text model

Test Unit: PermissionTests

Tests the permissions for accessing alarm data

Test name	Exp. Res.	Act. Res.	Pass/fail
unauthorized	-	-	Pass
health professional	-	-	Pass
no permission	-	-	Pass

Test Unit: GetTests

Tests the retrieval of alarms list

Test name	Exp. Res.	Act. Res.	Pass/fail
unfiltered	-	-	Pass
filtered by patient	-	-	Pass
parse error	400	400	Pass

Test Unit: PostTests

Tests the posting of alarms

Test name	Exp. Res.	Act. Res.	Pass/fail
handle	-	-	Pass
handle without motivation	-	-	Pass

Table 10.4: Sprint 3 alarm testing

Test Case ID: TID04

Test Case Name: Module Measurement Test

Tester: Sigurd Sandve

Description: Testing of all the interaction of the measurement model

Test unit: PostMeasurementTest

Tests all the interaction with posting new measurements

Test name	Exp. Res.	Act. Res.	Pass/fail
post ignored measurements	-	-	Pass
post when not authenticated	-	-	Pass
post bad hub id	-	-	Pass
post abnormal low	-	-	Pass
post abnormal high	-	-	Pass
post abnormal low repeated	-	-	Pass
post low multiple	-	-	Pass
post update daily activity	-	-	Pass

Table 10.5: Sprint 3 measurement testing

10.5.3 Test Evaluation

The tests were set up to monitor the system as code was developed. If any test failed, it was either due to new code breaking something, or that new code changed the way the system worked. So a failure in a test was not a definitive proof that a bug was introduced, but could mean that the change had altered the interaction between units, and the programmer would have to be extra careful to see how this might effect other code in the project. If everything seemed correct, the test would have to be rewritten to reflect the new code. The tests worked really well for this purpose, and helped the team develop faster.

10.6 Customer Feedback

Along with the completion of the last pieces of the front-end, the customer was given access to try it out for themselves. This resulted in new requirements which is listed and described in section 4.4.3. It was confirmed that no third party should have access to the measured data, which is the case with the Withings sensor, see 11.3.2. The customer was pleased with the purchase of the domain angelika.care as well as the logo the team had developed. Test data should be included when delivering the system.

10.7 Sprint Evaluation

10.7.1 Review

As this was the last sprint, both the pressure and expectations were higher. The whole team agreed that the focus of the last sprint should be documentation. This was because a too small amount of work regarding documentation was done in previous sprints. However, there was still some implementation left to do, especially on the front-end, due to new requirements from the customer. This resulted in the lead programmer continuing with implementation during the whole sprint, while the rest of the team shifted focus to documentation after finishing their work on implementation.

It was gratifying to see that the backlog was finished and that all members of the team stepped up and worked both harder and more efficient.

The team also had to perform UAT in this sprint. However the customer was unable to give us a time slot to do this. The team would have preferred to set up the system at the customer simulating real world usage as closely as possible. However, lacking this option, a UAT was performed using the team members that had not been involved programming the front-end. This solution was not ideal, but better than not doing a UAT at all. The test showed that the system covered all the customers requirements except one, however this requirement was completed after the testing was done. The system was deemed compliant with the customers expectations. The full test can be read in the appendix A.1.

Figure 10.3 shows the burndown chart for the sprint.

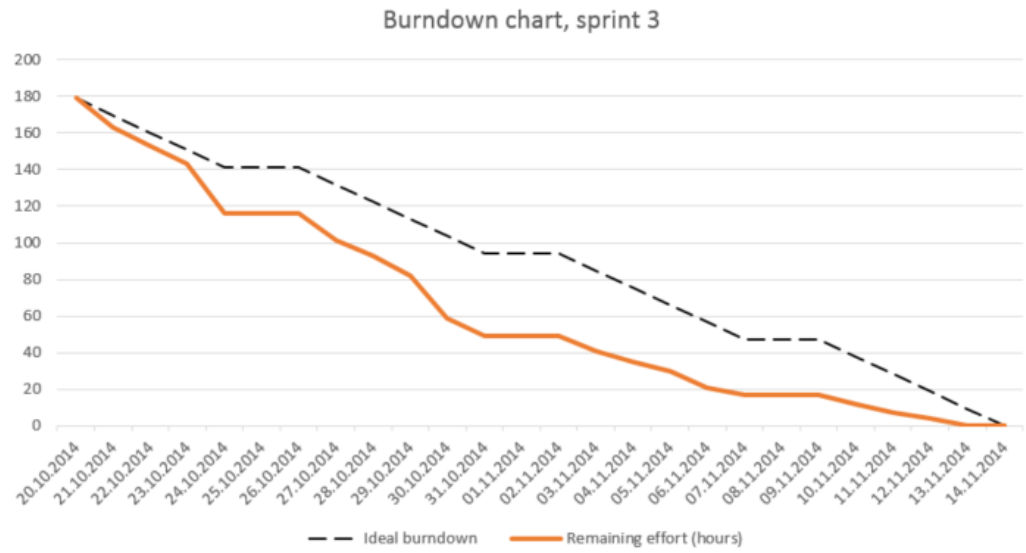

Figure 10.3: Burndown chart, sprint 3

10.7.2 Workshop with Designhjelpen

After having planned a workshop with Designhjelpen for a month, the workshop finally took place. The workshop agenda can be seen in figure 10.4. The whole team, representatives from TK, and several design students was present. It started out with all parties introducing themselves and explaining what their goal for the workshop was. Second the participants was divided into groups consisting of both members of the team and design students. These groups discussed, brainstormed and tried to determine ways of designing the service. At the end, each group pitched their idea.

As the workshop was arranged at such a late point, it was reassuring to see that the team already had thought of many of the ideas that emerged. All members felt that they gained a broader understanding of how the system could and should be used. The only downside with the workshop was that it was, as mentioned, arranged slightly too late for it to make a serious impact on how the system was designed. This was due to a high demand for workshops with Designhjelpen. They could not schedule the workshop earlier.

10.7.3 Positive Experiences

- Much changed in the early stages of this sprint, both with the Angel sensor not arriving and a great deal of new requirements. Despite all

Figure 10.4: Agenda for the workshop with Designhjelpen

these changes, the team worked steadily and managed the changes well.

- All the members of the team worked longer and harder.
- Since there was little work to be done on the back-end and the hub was finalized early in the sprint, more of the team could focus on documentation.
- Had the workshop with Designhjelpen.

10.7.4 Negative Experiences

- Since the documentation was not in focus during earlier sprints, there was much work to be done during this sprint.
- Many new requirements emerged for the front-end and the time the team had to finish them was scarce.
- The Angel sensor did not arrive in time.
- The workshop with Designhjelpen was perhaps slightly too late to have an impact on the project.

10.7.5 Barriers

This sprint was marked by the biggest barrier of the project, the Angel sensor not arriving on time. The Angel sensor is, as mentioned in section 3.3.1, not a finished product yet. The same day as this sprint began, the team got the message that the sensor was delayed awaiting EC [55] and FCC [56] declarations of conformity.

The risk assessment and mitigation plan for the Angel sensor is described in table 2.8 on page 23. The consequence of the sensor not arriving was estimated to be high, but by implementing the the mitigation plan and finding a replacement sensor, the current impact on the project was minimal. The team designed the system to work with the replacement sensor and at the same time be extendible and modifiable enough to accept other sensors such as the Angel sensor at a later point.

Part III

Conclusion and Evaluation

Chapter 11

Conclusion

11.1 General

In this chapter the final state of the system will be described. This chapter will also discuss opportunities for further development, along with a short discussion of how testing was used in the project.

11.2 System Overview

11.2.1 System Summary

The finished system is a prototype, designed to be stable and functional to a degree that makes it suitable for testing its functionality in near reality environments. Structurally the system consists of four main components: A sensor, a hub, the system server and the front-end web application. Two additional components have been added as a result of changes that happened in sprint three, see chapter 10. The additional components are an app running on a smart phone and a third party server. These were introduced in order to facilitate a replacement sensor, but much of the original architecture was maintained in order to make the system modifiable and thus compatible with other sensors.

The sensor monitors the wearers heart rate, blood O_2 saturation, and activity. Data is sent from the sensor, via a mobile app and a third party server, to the hub. The hub then passes the data to the system server, where it is stored and analysed for abnormal values. The server maintains a list of alarms that it generates when it receives abnormal values from a sensor. The data on the server is accessible for healthcare professionals and users using the system's web application. While healthcare professionals have access to

the information of all users, users only have access to the parts of their own information that healthcare professionals have approved access to.

11.2.2 Hub

The hub is developed with emphasis on modifiability regarding switching of sensors. In figure 6.3 the final structure of the hub is shown.

The hub's responsibilities are to pull data from the sensor, cache it and forward it to the server. In the figure referenced above, the hub script handles the different threads that is responsible for these tasks. The thread that pulls data from the sensor tries to do this at specified intervals. If there are any data to be pulled, it is temporarily cached as a JSON file. A different thread checks the cache at specified intervals and forwards any data that is not previously forwarded to the server.

The modifiability regarding switching sensors is obtained by writing code that is specific for the sensor in a subclass of the abstract Sensor class. By implementing the methods required by the abstract Sensor class, a new sensor can be integrated into the system without changing the rest of the structure.

11.2.3 Back-end

The back-end is developed with Django and Django REST Framework. These popular and well-documented open source frameworks are designed for high security, dynamic development and readable code. The code is tested using Django's test framework. The database queries generated by Django's Object-relational mapper are optimized for high performance in a single-server environment with a PostgreSQL database. However, Django makes it easy to swap the database, in case that should be necessary later. During development, permissions has been a focus. The data stored in the system is sensitive personal data and has been made inaccessible to users without the correct permissions.

The team has also focused on writing tests for the API. The main reasons are; check that the code works as expected when implementing new features, and after having modified old code, ensure that the existing behaviour of the application has not changed unexpectedly. Django's test-execution framework makes this easy.

The back-end provides two different front-ends: The REST API, which is browsable, and Django Admin. The REST API is standardized and provides typical endpoints for Create, Read, Update, Delete (CRUD). The REST API is what provides data (in JSON format) for the front-end. Django Admin is an admin interface that is only available for super users (system

administrators). It provides forms for all the models in the system. This interface can be used to do tasks that are impossible in the Angular front-end, such as connecting a user with a hub, and deleting a user.

11.2.4 Front-end

The front-end is developed with focus on usability, a diagram describing structure of the front-end can be seen in figure 6.2.

The front-end pulls data from the server, or back-end, and presents this to the system users. It provides different views for the two user groups, healthcare professional and user. The healthcare professional can view all information about all users, edit this information, and control what sensor data users should be able to see. Healthcare professionals can also view and handle alarms. Users can only view sensor data from themselves that have been approved by a healthcare professional, they can not see any alarms.

The front-end's main responsibilities is to get the latest alarms from the back-end and notify healthcare professionals when a new alarm is generated. It also provides an easy to use interface for the healthcare professional to view and edit user information.

11.2.5 Production Server

- Front-end URL: `http://angelika.care`
- API URL: `https://api.angelika.care`
- Django Admin URL: https://api.angelika.care/admin/
- Owner: NTNU
- Geographical location: NTNU Gløshaugen, Norway
- Operating system: Ubuntu 14.04
- HTTP server: nginx
- Web Server Gateway Interface (WSGI) server: uWSGI
- SSL Certificate: Comodo PositiveSSL
- GZIP [47] compression enabled

11.3 Further Development

11.3.1 HL7

If the system is to be integrated with other healthcare systems in Norway it has to use nationally recognized standards. HL7 is on its way into the Norwegian system, but other standards might be considered. The reason that this was not implemented in the time of this project is because the standard and the national guidelines for its use is not quite mature. The customer therefore removed this requirement during sprint three, see section 4.4.3.

11.3.2 Sensor

Using the Withings sensor raises privacy related issues. The Withings sensor can only synchronize its measured data with a mobile device running the Withings app. This app automatically sends all measured data to the Withings servers. If the data is to be used in any way, it has to be retrieved through the Withings API. During a meeting with the customer it was clearly stated that any data measured could only be owned by TK and neither be owned or accessed by any third party, effectively maintaining the users privacy. The backup sensor will therefore not work as a replacement of the Angel sensor, but as a backup to show the potential of the system.

Modifying the system to use the Angel sensor, or any other sensor, has been a major focus for this project. Thus, modifying the system to accept another sensor that is both more versatile and less restricted by privacy issues is a logical step ahead. The reason why this was not done during the project is, as stated in chapter 10, that the Angel sensor did not arrive at the planned time.

11.3.3 Further Improvements to the Front-end

The front-end interface has received a great deal of focus, but it could still benefit from further development. For example by improving the graphic representation of data, both for healthcare professionals and users, but primarily for the user. The user side of the service have never been the focus for the customer, but it is still an area with many interesting possibilities. For further development of the system, one might want to research ways of visualizing the sensor data for the user in an understandable and motivating manner.

Towards the end of the project the team experimented with the idea of presenting user with a fuel gauge. By combining the measurements for a user

into a single value that can fill the gauge. This will give the user a simple representation of how well he/she is doing.

11.4 Testing

11.4.1 Testing Methods

Testing was somewhat overlooked at the beginning of the project. This was partly because the team had a very slow start where planning and requirement specification took up most of our time. Lack of experience writing and performing testing was also a factor. However, the team took necessary steps to learn about this later in the project.

The team had around 50 tests for the back-end running continuously making sure new code did not break old code. This was especially important to ensure that data management and logic was kept intact as the system grew more complex.

At the end of Sprint 3 the team completed a full UAT where all the requirements where tested by using the user stories 4.6 which were produced using the requirements gathered from the customers. The UAT passed with all points except for one user story. However this user story was only a minor feature and not vital for the system to function.

11.4.2 Testing Conclusion

Overall, testing could have been done better, however the team is happy to have had tests during development of the most important aspect, handling of the data. Doing a full system test A.1 at the end making sure the requirements was met was also an important milestone making sure nothing was overlooked, and that the system performed as expected.

11.5 Summary

The project objective was to create a service that utilizes a sensor to monitor a user's vital signs, and makes this data available to healthcare professionals. Providing healthcare professionals with this data, the system can make their work more manageable by examining users' current situations faster, limiting home visits, and improving communication with users. Another objective for the team was to create a solution with high modifiability, in order to make the service adaptable and able to change according to the customer's needs.

The team believes that the solution presented in this document achieves these objectives and will be a sound resource to build future development on.

Chapter 12

Project Evaluation

12.1 General

This chapter provides an evaluation of the project, mainly describing how the team felt about different aspects of the project after it was completed.

12.2 Team Dynamics

12.2.1 Goals and Team Building

The team consisted of seven students from NTNU, chosen randomly from the class. At the start of the project, the team did some exercises to improve relations within the group. The team also had to share previous experience and knowledge to help planning and choosing project roles. There was also a mandatory lecture focusing on team building. Here the team decided on some group goals, reiterated below:

- Working prototype
- Good presentation
- Well written report
- Satisfy the customer
- Easy to use product
- Learn as much as possible

12.2.2 Team Evolution

The team evolved a lot during the group project. From knowing close to nothing about each other, to knowing each other's strengths, weaknesses, and personalities. There were not any internal conflicts within the team during the beginning of the project, but this was not necessarily a good thing. In fact, a reason for this might be the fact that everyone did what they wanted to do in the earlier stages of the project. The team lacked understanding of the amount of work required, and progress was slow. As the project progressed, everyone agreed to increase their effort. The team members having put less hours into the project had to start working more in order to finish on time. A combination of time pressure, and that team members now knew each other actually created some heat within the group. The team evolved from a state where everyone did what they wanted, to a state where a leader assigned tasks. The team was not afraid to be critical of each other and pushing each other to perform as expected.

12.2.3 Roles and Responsibilities

There were some changes in the assigned roles during the project. The sub-team division where members worked on different layers worked out nicely, but other parts of the project needed more focus. This caused a reshuffle of the roles where many team members wrote the report full time.

The team found that the members of the different sub groups spent considerable time learning the different technologies, there was simply no time to have people learn more than the technology for one layer. By changing responsibilities new ideas might have been introduced to the different divisions of the project, but this gain was estimated to be smaller than the time it would take re-training a team member.

12.3 Risk Handling

R1. Conflict between team members There were minimal conflict between the team members, but towards the end of the project, some tension sparked. Some team members were stressed about finishing the project on time, and to a high standard, and did not feel that all team members were pushing their own weight. The problems were discussed and the tension between the team members was relieved.

R3. Other commitments or coursework interfere with the project Other coursework did interfere with the project somewhat, but as most

of the team members do not take the same courses, the weight of one team member having to prioritize other school work was relieved. Extracurricular activities were a bigger liability for the team, some team members prioritized leisure activities before project work.

R4. Failure by team members to attend meeting As the project progressed, team members were less and less inclined to come to meetings. Cake punishment were observed for a time, but the incidents became so frequent that the team resigned in trying to enforce the attendance duty. The impact was minimal though as most team members were hard working and able to distribute the added work evenly.

R5. Illness/absence This risk is related to R3 and R4, illness was only an issue at the end of the process when a team member became ill with diarrhoea.

R6. The Angel sensor does not arrive in time This is described in section 10.7.5. This did occur, but was handled with minimum negative effect on the system and the course of the project.

12.4 The Scrum Process

At the start of the project, during the pre-study, the team decided to use an agile development method for the development process. Only one of the team members had any previous experiences with agile development, apart from attending lectures and reading about the theory behind the methodology. During the first sprint, this inexperience with the process lead the team to do far to little planing. The sprint was started to early and thus the team had to work on what had not been established during pre-study and planning phase. This meant that little development could be done during the first sprint. Another aspect with the Scrum process that was problematic for the team to carry off, was the delivery of the product at the end of each sprint. The customer did not always have the opportunity to schedule a meeting at the end of each sprint. Thus, the deliveries could not be carried out as specified by the strict Scrum rules. The team had to adapt a more pliant form of agile development where the customer was precedented with the current state of the product at every customer meeting. This was useful for the team, since it proved hard to gather requirements from the customer without showing them explicit examples of how the team imagined the finished product. One could say that the team adapted a development methodology uniting customer-driven design and Scrum.

The main experience that the team has had from using, or attempting to use, Scrum in this project is that it is exceedingly challenging for an inexperienced team to keep to the very structured style of Scrum. Without having at least one member with significant Scrum experience to drive the process, it might actually be near impossible to manage the process in a satisfactory manner. Despite the difficulties the team has had with Scrum, the process has been quite instructive and the members of the team have learned much about the methodology.

12.5 Time Estimation

Estimating time usage is always challenging, especially for an inexperienced team. One aspect of time estimation that was not taken into account from the start of the project, but became evident when development started, is how much time would be spent waiting for requirements. When development started, the team had to have regular meetings with the customer in order to keep the flow of the development up. When the time between meetings was to long, the team was pacified by not having clarified all functionality with the customer. Not all aspects of the functionality could be determined in advance, some questions about functionality only became apparent as an effect of implementing other, customer specified, functionality.

The time that was estimated by the course staff to finish the project, was approximately 2200 hours, the team's effort have been slightly over 2000 hours. This 200 hour variation from the estimated time is in part due to having to wait for specifications from the customer. This is not because the customer was uncooperative, it was merely time-consuming to plan meetings to get feedback on the progress of the project. This sometimes left the team unable to continue the implementation until after meeting with the customer.

12.6 Quality Assurance

This section will discuss how the measures for quality assurance that were decided in section 2.4 have affected the quality of the project.

12.6.1 Routines for Producing High Quality Internally

The use of pull requests assured a high level of quality for the code, this practice stopped numerous mistakes and lines of substandard code ending up in the main branch.

The different development teams worked together in teams most of the time. This did as suspected contribute to knowledge transfer between the more experienced members of the team and the less experienced.

12.6.2 Routines for Approval of Phase Documents

The routines were changed a bit as the project progressed, the team needed feedback on the state of the product more often than what was proposed in the planing phase. Therefore the team presented and received feedback on the state of the product on every customer meeting after the implementation had started.

12.6.3 Procedures for Customer Meetings

At the start of the project the team produced and sent agendas as proposed in the planning phase, but as the project progressed the agenda was replaced by a list of questions and the meetings became less formal and more direct in terms of gathering requirements from the customer. This was a concious choice by the team because the customer had limited time for meetings.

12.6.4 Procedures for Advisor Meetings

Having frequent advisor meetings was important for the team to ensure progress and resolving issues or uncertain elements involved in the project. Also the advisor provided contacts to people that could help answer questions the team had during development.

12.6.5 Document Templates and Standards

Producing templates (See Appendix D) for documents concerning meetings and reporting was a good choice to save time and making sure that everything was consistent from week to week.

12.6.6 Coding Routines and Standards

Coding in the back-end was done following the PEP8 standard, which is the de-facto code style for Python [45]. This standard covers code layout, naming convention, and commenting.

12.6.7 Version Control Procedures

Using Git version control system proved to be a good choice. There were no problems during the project synchronizing the code and making sure everyone had the latest builds.

12.6.8 Internal Reports

Using Trello 2.2.3.4 for assigning tasks and keeping track of progress was a very neat way of organizing the project. Although this tool was not used to its full potential by the team, it still helped the team to organize.

Google Drive (see section 2.2.3.2) was a great way to organize documents shared among the team members, and allowing multiple members to work on the same document seamlessly.

12.7 Customer Relations

Throughout the project the team and the customer maintained a good relation with frequent meetings and communication by e-mail. The team were originally assigned one customer contact, but for the duration of the project the team maintained contact with two other customer representatives as the originally assigned contact rarely had time for meetings. The customer representatives have been easy to get in touch with, and very helpful. In the start up period of the project the team and customer had meetings every week, but as the development process progressed the frequency diminished. Towards the end of the project meetings were only held once every other week. Even though there were fewer meetings, the team still shared updates with the customer by e-mail.

The biggest challenge with the team had with the customer, was that the representatives had no specific idea of what they wanted. This meant that the team had to use a lot of time trying to find out what the customer wanted. By applying user-centred design principles, the process of determining demands was made more manageable. User-centred design does not help with determining all requirements, but it was helpful when finding requirements for the front-end. This is reflected in the amount of requirements the team was able to gather for the front-end functionality, and the lack of customer specified requirements that were gathered for more technical aspects of the system.

Having a customer who was unable to hand over a list of requirements for the system was, in addition to being a challenge, also the most fun and

interesting part of the project. It meant that the team had to do a lot of requirement specification, talking with the customer and making mock-ups and presenting suggestions. This made the process very creative and fun. This might not have been the case though, if the customer representatives had not been equally enthusiastic and interested in creating a great solution.

12.8 Advisor Relations

The team's advisor, Jon Atle Gulla, was an important resource for the project. He helped push the team to work more and endeavour to create a high quality product. Gulla was also able to suggest people that could help with the project. The weekly advisor meeting was a driver for the team, something to work towards. At the meetings the team would have to be able to justify choices that had been made and present what progress had been made the previous week. The advisor often had other conditions than the customer for what was good for the project. By pleasing both parties the team was of the conception that the final product would be of a higher quality.

12.9 Course Evaluation

Overall the team's experience of the course has been very positive. There are still some points that may have been done better.

- Structuring the report was demanding, the team had to spend time looking through old reports in order to make a structure for the report. This time would have been better spent writing real content for the report. It would have been very helpful if there had been a guest lecture with focus on structuring and writing a big report.
- The team had problems using Scrum in an optimal way, the guest lecture was not very helpful in enabling the team to carry out the process. It might be better to use a workshop approach, where the teams could be guided through and try the Scrum methodology, supported by an experienced Scrum user.
- The lecture in group dynamics was too late and the format did not work. Putting all the course participants in one big lecture hall was not optimal. It might be better to include this in the proposed Scrum workshop, letting the team members get to know each other through working with Scrum.

- The team found that a workshop with Designhjelpen was very helpful in exploring the problem for the project and collecting requirements from the customer. It might be a good resource for all projects in the course, especially early in the proses as it gives the team a kick start on the project.

12.10 Summary

Taking this course and working on this project for TK has been an educational experience. The team has learned about software development, keeping customer relations, and working as a group. By working on a product for welfare services the members of the team also learned about the healthcare sector in Norway and challenges the sector is facing.

The team have been fortunate to work with such a relevant and interesting problem, and all team members believe that this has been a valuable experience.

Part IV

Appendices

Appendix A

Test Cases

A.1 User Acceptance Testing

UAT does three things for our project: It provides a measure of how well the system is compliant with the customers requirements. It also provides a way to expose functional logical problems that regular testing might have missed out on. Finally it provides a measure of how "done" the system is.

The UAT was done one the 13th of November, one day before sprint 3 was scheduled to be done. Sprint 3 was the last sprint, so most functionality was expected to be done by this point. Furthermore it was important to perform this test to get an overview and make sure that no important functionality was missing.

The test was performed with a user unfamiliar with the front-end. The user would perform the tasks described in the user stories, with no help from the test leader, however the user was allowed to ask questions. This way confusing elements or poor design might also be discovered by the test. Some of these things were written down as cards for the front-end team, but are not documented here.

The UAT should ideally be performed with the customer in the real environment the system is intended for, however this was not possible with the resources available for the customer. This is why the test was done with a team member instead.

Test Case ID: UAT01

Test Case Name: User Acceptance Test

Tester: Sigurd Sandve

Description: Testing of the full system

Test unit: User Stories

Each user story will be tested using a user unfamiliar with the system

User Story	Result	Comment
US01	Pass	
US02	Pass	
US03	Pass	
US04	Pass	
US05	Pass	
US06	Pass	
US07	Pass	
US08	Pass	
US09	Pass	
US10	Pass	
US11	Pass	
US12	Pass	
US13	Pass	
US14	Pass	
US15	Pass	
US16	Pass	
US17	Pass	
US18	Pass	
US19	Not done	This feature was not implemented at the time of testing
US20	Pass	
US21	Pass	
US22	Pass	
US23	Pass	
US24	Pass	
US25	Pass	

US26	Pass
US27	Pass
US28	Pass
US29	Pass
US30	Pass
US31	Pass
US32	Pass

Table A.1: User Acceptance Testing

US19 is found in table 4.6, but is also seen below. The reason this requirement was unsuccessful was because this feature was not implemented at the time of the testing. This feature was implemented after the the completion of this test and tested alone.

"As a healthcare professional, I want to make an audio recording of a message to a user, because this will both save me time and the user might find it easier to listen to a message rather than to read it"[1]

The result of the testing was that the system performed as expected and that the system was capable of satisfying the requirements set by the customer. No major faults were found in the system.

A.2 Performance test

A performance test was done at the end of sprint 3, where a script was created that generated a lot of data for the system. This was done to detect potential weaknesses in the system. Around 5800 users were created, 150 000 threshold values and about 2 700 000 measurements. It takes 0.3 seconds to get a set of measurements and 6.2 seconds to get all the users. Searching for a specific user takes 0.1 seconds. A stress test was also performed where

[1] User Story 19

about 20 000 alarms were created in a short time span, and this caused no problems for the server.

Appendix B

Front-end testing

Test Case ID: FET01	
Test Case Name: User Acceptance Test	
Tester: Iver Jordal	
Description: Testing of the front-end on different systems	
Smart phones:	iPhone 6 (iOS 8)
	HTC One X (Android 4.0)
	Nokia Lumia 820 (Windows Phone 8.1)
Tablets:	iPad 2 (iOS 7)
	Acer Iconia A1 (Android 4.4)
Desktop browsers:	Chrome
	Firefox
	Opera
	Internet Explorer
	Safari

Table B.1: Front-end Testing

This table shows the different types of environments that were tested after development was done. The front-end works on all devices listed in the table. The design is responsive, so that it adapts nicely to all the different screen widths. One thing worth mentioning is that Safari and Internet Explorer don't have the features required for recording sound with HTML5.

Appendix C

User and Developer Manual

C.1 Front-end

C.1.1 Introduction

This is the user manual for the system, aimed at teaching healthcare professionals how to use the system.

C.1.2 Logging in

Figure C.1 shows the login screen. Insert username and password and hit the "Logg in" button to access the system.

C.1.3 Tab interface

The system uses a tab based interface, meaning that several users can be open at the same time, represented by a tab. This is illustrated in the figures below. Figure C.2 shows the toolbar as it is when the system is first started. These three tabs contain key functionality and cannot be closed. Each of these tabs will be described in detail later. The number next to "Varsler" shows the number of unhandled alarms. In figure C.3, three users have been opened, and the active user is highlighted in blue. To switch active tab, simply click on the desired tab. A tab can be closed by clicking the 'x' on the right side of the tab. A tab can be moved by clicking and dragging it. Figure C.4 shows what happens when a tab is dragged to the right of the screen: the screen will be split horizontally with the dragged tab on the right and the other tabs on the left. Figure C.5 shows this view. A tab can also be dragged to the left, top or bottom part of the screen, resulting in the tabs

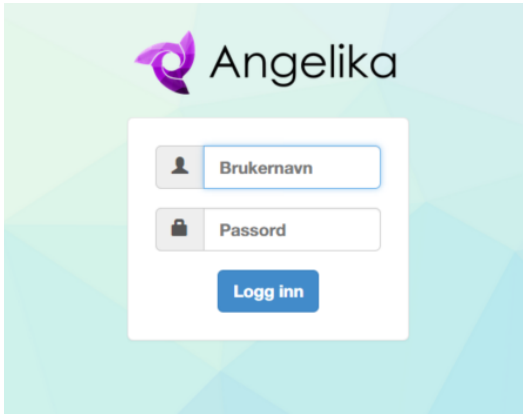

Figure C.1: Login screen

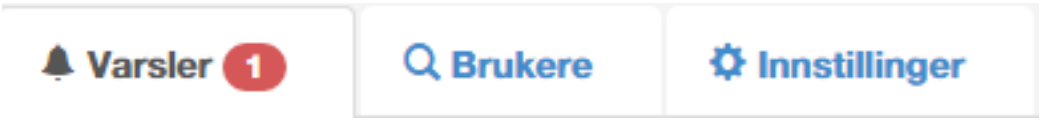

Figure C.2: Tab interface, no open users

splitting the screen the same way. Figure C.6 illustrates how to place a tab back among the other tabs.

C.1.4 Alarm overview

The tab "Varsler" (English: "Alarms") provides an overview over recent alarms in the system. This is shown in figure C.7. It shows a table with the name of the user, the time of the alarm, the alarm type, and whether the alarm is handled or not. To get more information about an alarm, simply click on it (see section C.1.9).

C.1.5 Users list

The tab "Brukere" (English: "Users") shows a list of all registered users in the system. See figure C.8. The search field enables search on name, date of birth or social security number, and updates the list dynamically on every key press.

Figure C.3: Tab interface with open users

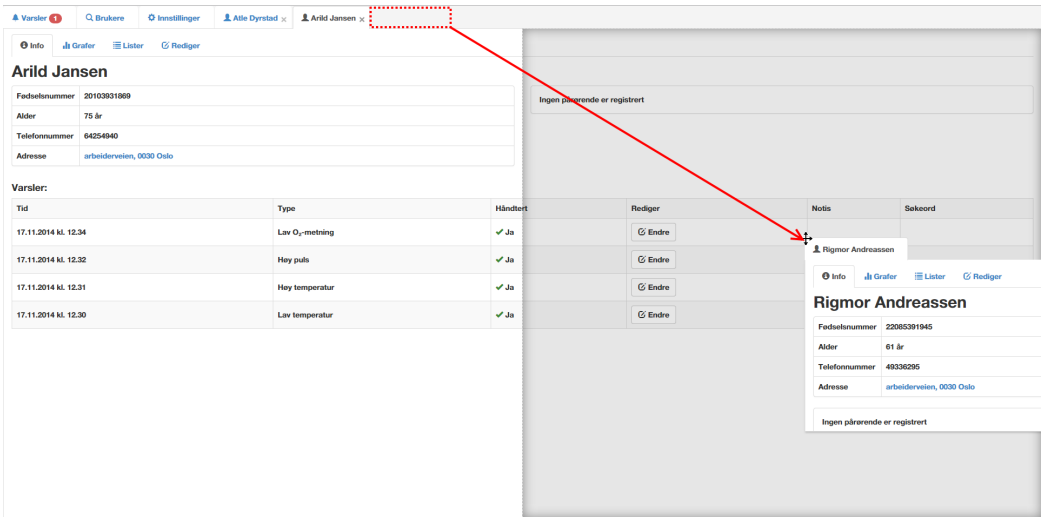

Figure C.4: Dragging a tab to the right of the screen

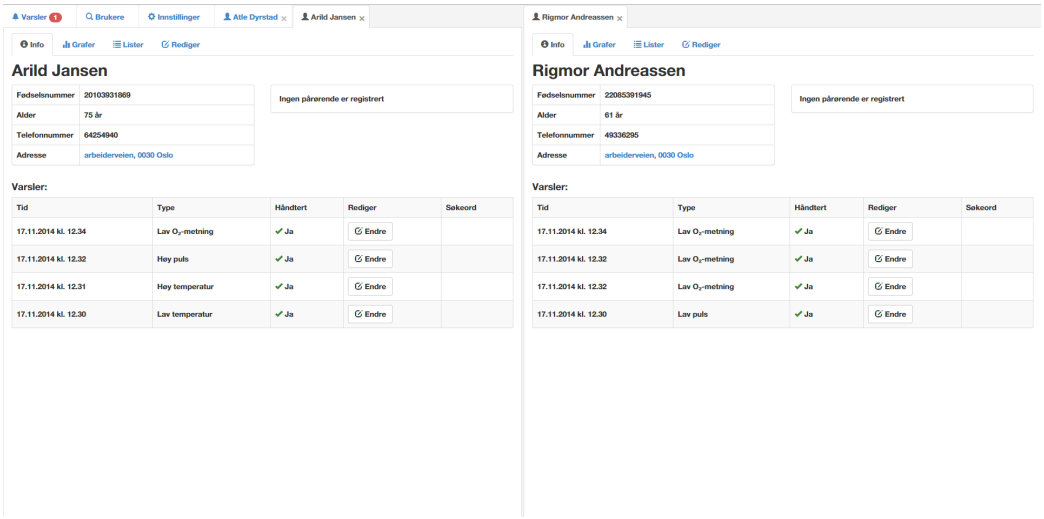

Figure C.5: Two tabs splitting the screen horizontally

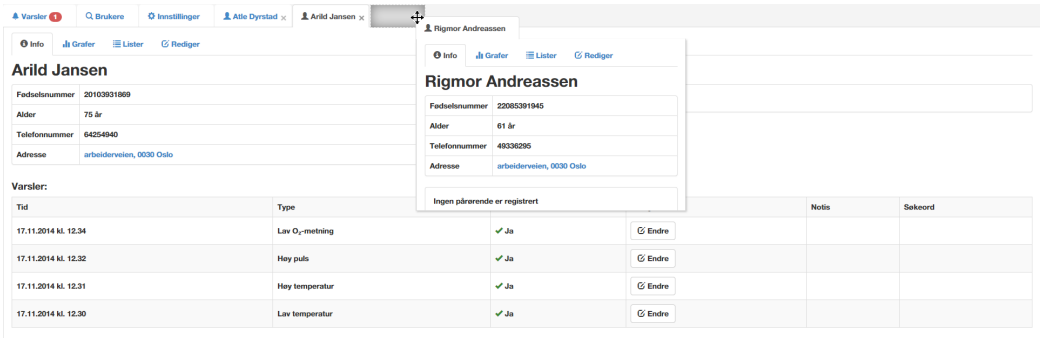

Figure C.6: Dragging a tab to place it next to other tabs

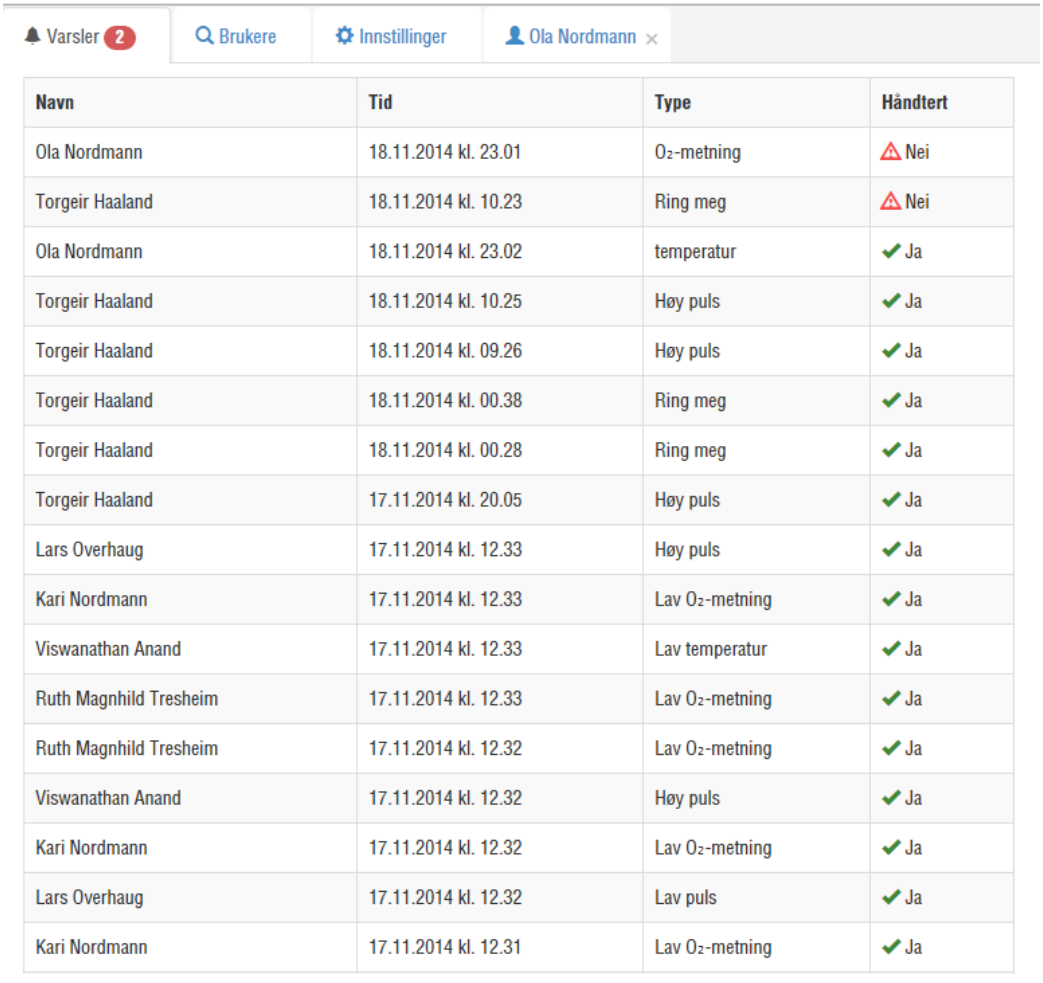

Figure C.7: Alarm overview

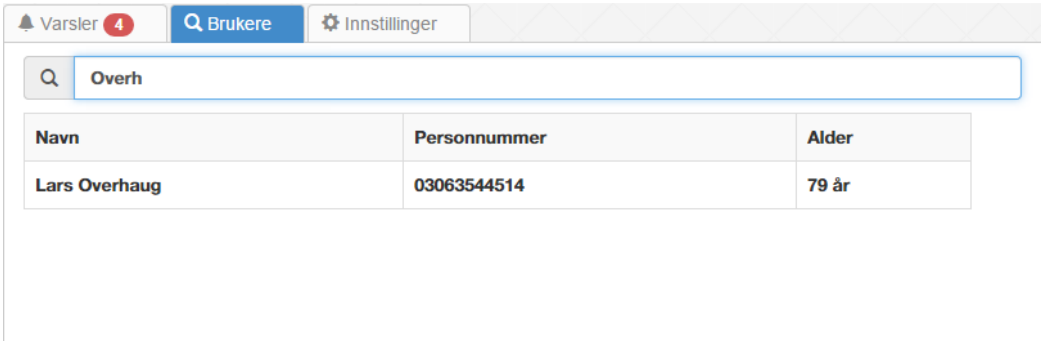

Figure C.8: User list

C.1.6 Settings

The settings tab (see figure C.9 contains a button for adding a new user (see section C.1.7), switches for enabling/disabling sound and filtering of alarms (showing only unhandled alarms), and a Log out button.

C.1.7 Adding new user

The 'New user' tab is divided into five sub-tabs, shown in figure C.10.

First, there is a tab for adding general information, like name, national ID number and address (see figure C.11).

Next, the tab "Normalverdier" (Eng.: "Threshold values", see figure C.12) is for specifying the threshold values for the user. Max O2 saturation is set to 100 % by default.

The tab "Synlige målinger" (Eng.: "Visible measurements", see figure C.13) contains check boxes for setting the visibility of the different types of measurements, both for the user (on the left, everything disabled by default) and healthcare professionals (on the right, everything enabled by default).

"Meldinger til bruker" (Eng.: "Messages to user", see figure C.14) has got buttons for adding motivational and informative texts that the user can see. It is also possible to record an audio message attached to a motivational text by hitting the "Spill inn talebeskjed" (Eng.: "Record audio message") button. You may need to allow the web browser to access the microphone in order to do this. While the system is recording (see figure C.15), a bar will display how long the audio clip will be. When the recording is done, hit the "Stopp" (Eng.: "Stop") button. Finally, the "Pårørende" (Eng.: "Next of kin", see section C.16) tab can create one or more next of kin for the user.

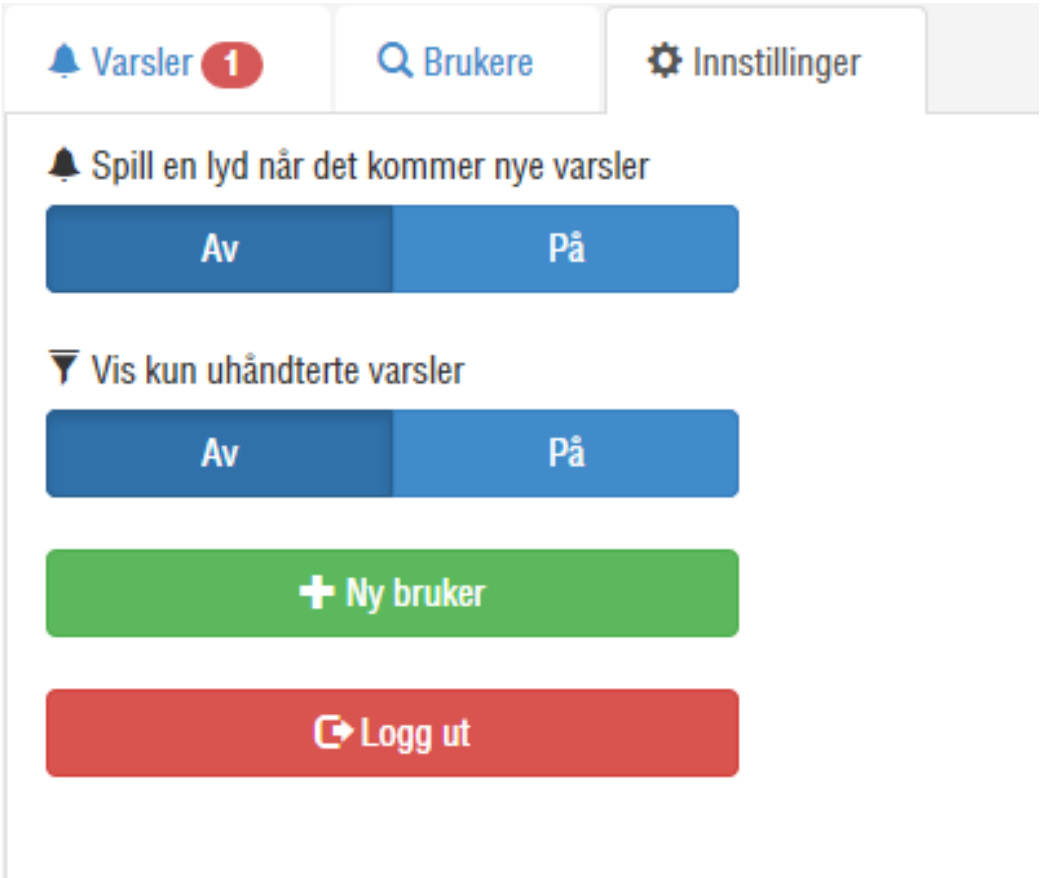

Figure C.9: Settings

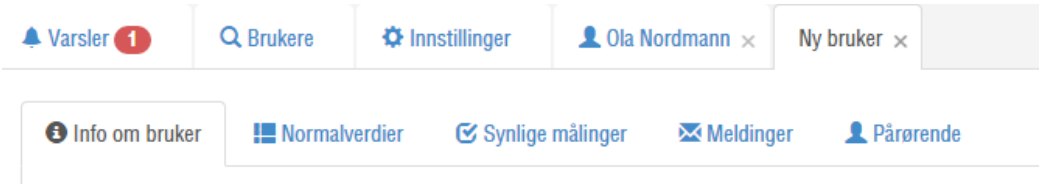

Figure C.10: Sub-tabs in new user tab

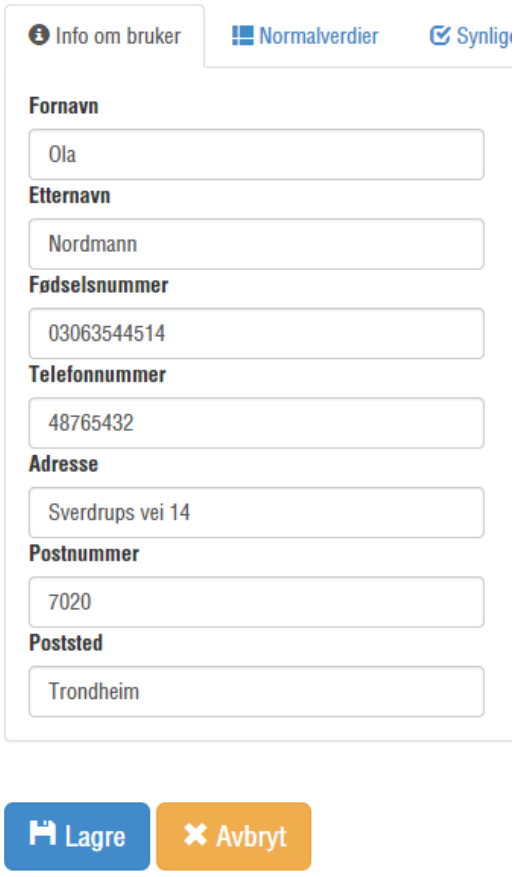

Figure C.11: New user tab, next of kin

Figure C.12: New user tab, general info

Figure C.13: New user tab, threshold values

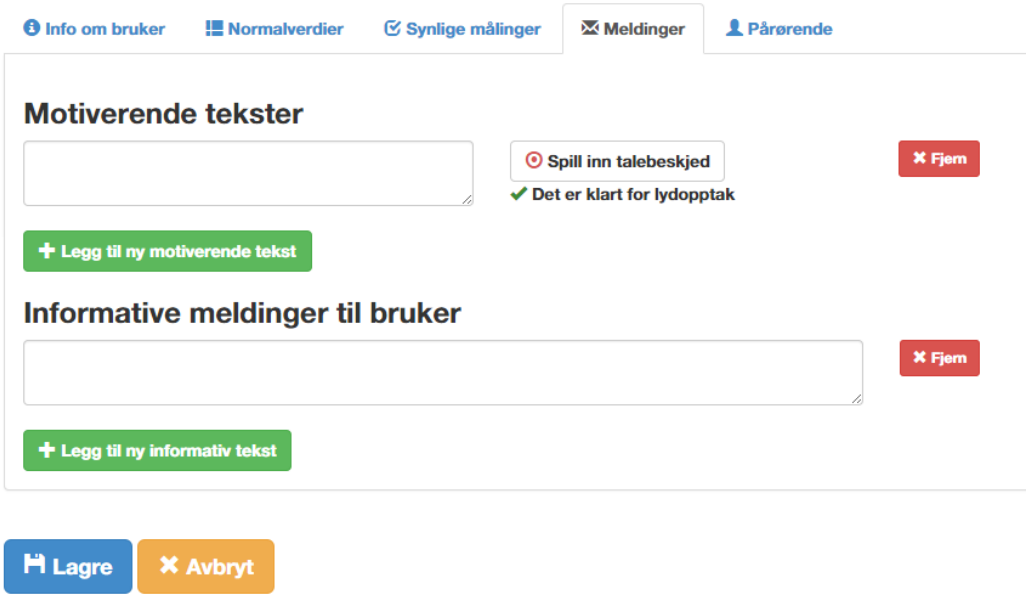

Figure C.14: New user tab, messages

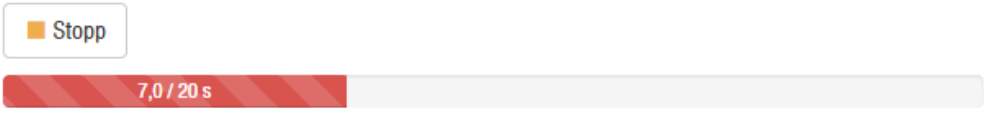

Figure C.15: New user tab, recording audio message

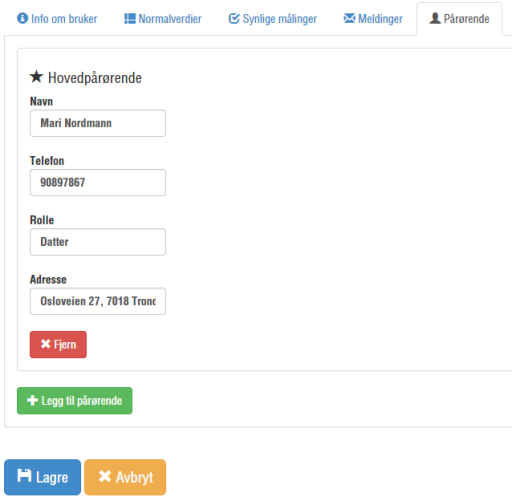

Figure C.16: New user tab, messages to user

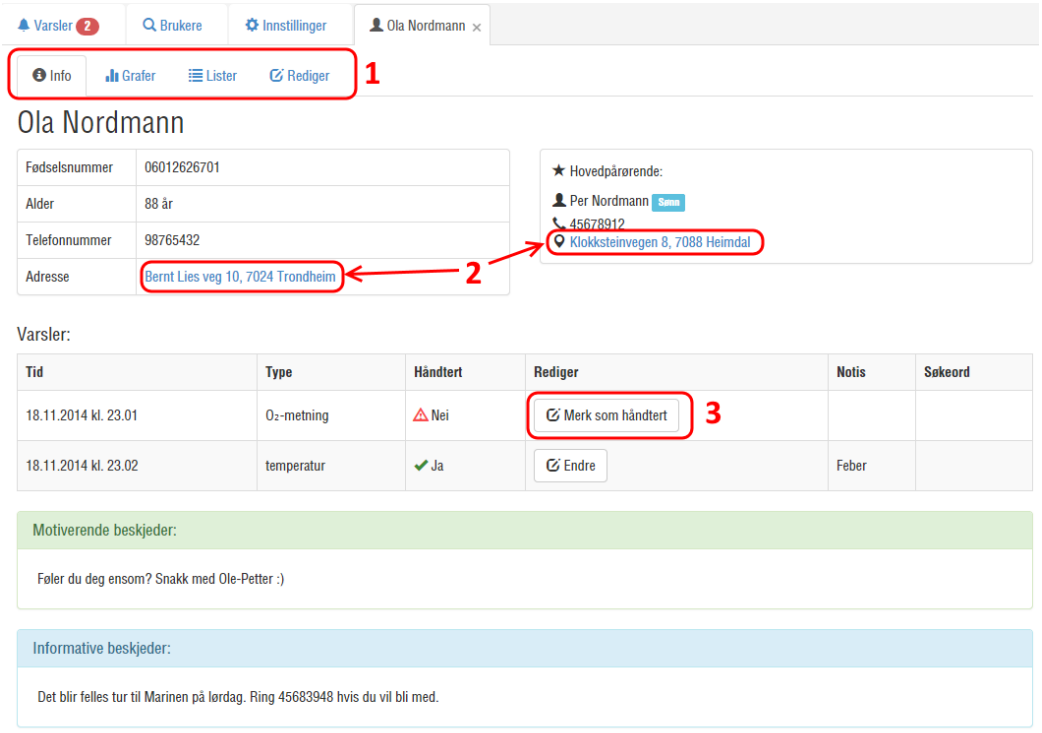

Figure C.17: User info

C.1.8 User info

The user info page shows a summary of information about the selected user. Figure C.8 illustrates this. One the top (marked with 1) are four sub-tabs: "Info" (described in this section), "Grafer" (Eng.: "Graphs", see section C.1.10), "Lister" (Eng.: "Lists", see section C.1.11) and "Rediger" (Eng.: "Edit", see section C.1.12). Below are key information about the user. Addresses (marked with 2) are clickable, which will open a new window with Google Maps on this address. There is also a section for alarms. For each unhandled alarm, there is a button called "Merk som håndtert" (Eng.: "Set handled", marked with 3 in the figure). Motivational and informative texts for the user are displayed at the bottom.

C.1.9 Handling alarm

The handle alarm dialogue is shown in figure C.18. It starts by listing the time, type, value and normal (threshold) values for the alarm. To mark an alarm as handled, check the check box "Håndtert" (Eng.: "Handled").

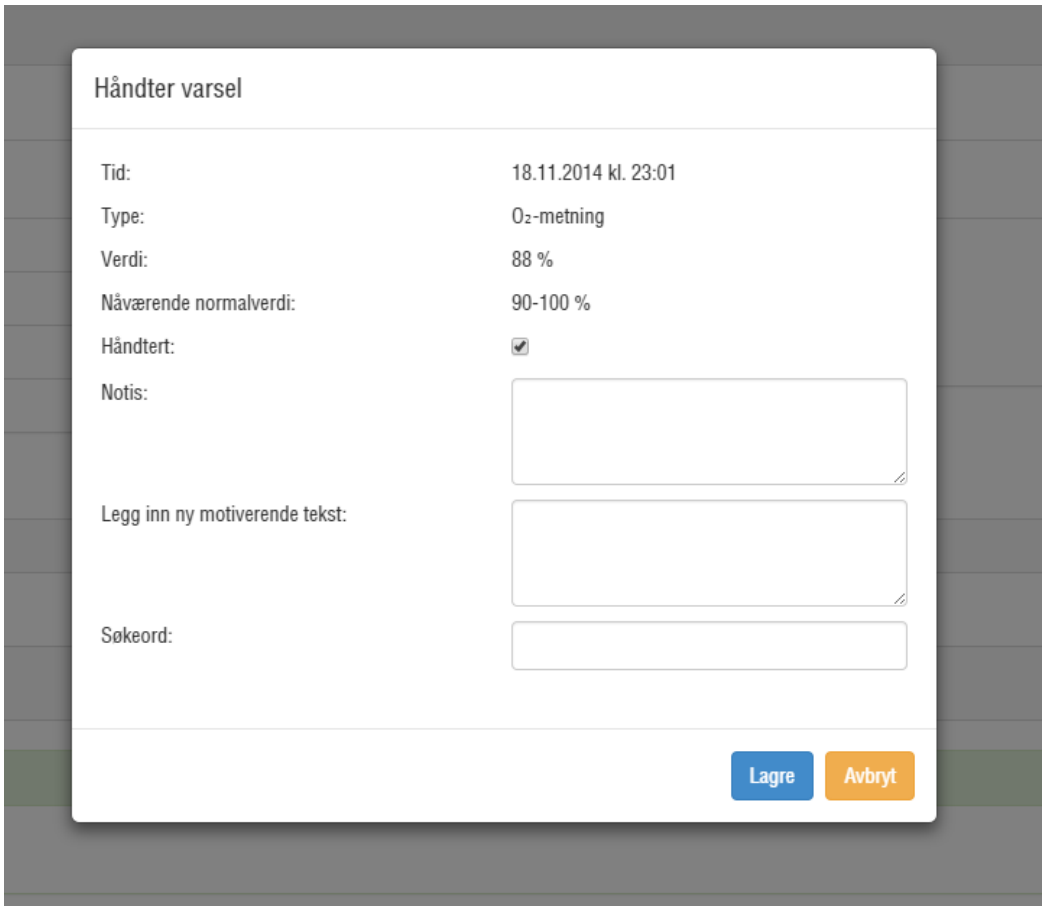

Figure C.18: Handle alarm

It is also recommended to write a note about what happened, and it is also possible to add a new motivational text, and a keyword for the incident. Click "Lagre" (Eng.: "Save") to save or "Avbryt" (Eng.: "Cancel") to discard changes.

C.1.10 Graph view

The graph tab displays sensor data in graphs (see figure C.19). The line graphs have green and red background colours that indicates threshold values. If a point is inside the red area, it means it is an abnormal value. An unhandled alarm is indicated by an exclamation mark icon, marked in the figure with 1. When the alarm has been handled, the icon changes to a red dot (marked with 2). To handle an unhandled alarm, click on the icon, and the handle icon dialogue will open (see section C.1.9). To change the time

span of the graphs, click one of the buttons in the upper left corner (marked with 3).

C.1.11 List view

The list tab displays sensor data in table form (see figure C.20). A red background colour on a cell indicates an abnormal value. Click on a red cell to open the 'handle alarm' dialogue (see section C.1.9).

C.1.12 Edit user

The edit tab is identical to the 'new user' tab. See section C.1.7.

C.2 Configure a new hub

It is important to note that before a hub can be configured, the user who will use the hub must be created from the front-end interface and the hub itself must be created in the back-end administration interface along with the relation between them.

To create a hub, navigate to the back-end administration interface and create a new user (legg til bruker) fill out username and password and click "lagre". This information is needed later when configuring the hub. More options becomes available. Under "grupper" select "hubs" and click "lagre". Now, find the user that the hub should belong to under "Patients" and add the newly created hub from the drop-down menu on that patient.

Software to format and write to an Secure Digital (SD) card is required. SDFormatter [51] is recommended for formatting and Win32DiskImager [52] is recommended for writing. To configure a hub, the following is needed:

- Raspberry Pi, model b+ was used during development.
- Ethernet cable connected to the internet.
- Micro USB cable to power the Raspberry Pi.
- SD memory card, either mini-SD card or micro-SD card, depending on the Raspberry Pi model. Must be at least 4 Gigabyte (GB). Micro-SD card is needed for Raspberry Pi model b+.
- Equipment to read/write the SD memory card, for example a computer with an SD memory card slot or a USB SD card dongle.

Figure C.19: Graphs

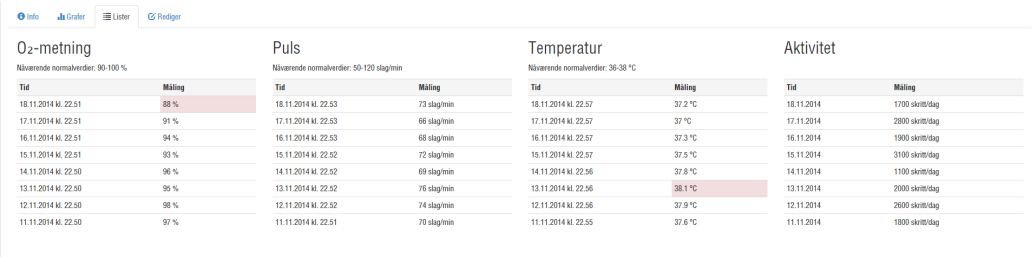

Figure C.20: Lists

- USB keyboard.
- High-Definition Multimedia Interface (HDMI) cable.
- HDMI compatible screen.

`Text typed in this style are commands that should be typed.`

Commands must *always* be followed by pressing enter. To configure a hub, follow these steps:

1. Insert the SD memory card into the SD card reader and make a note of the drive letter allocated to it. For example G:/.
2. Open SDFormatter, choose the correct drive letter and make sure the QUICK option is chosen. Click Format.
3. Open Win32DiskImager. Under Image file, browse and select the HUB .img file. Choose the correct drive letter under Device. Click Write and wait for the message "write successful".
4. Safely remove the SD card from the computer and insert it into the Raspberry Pi.
5. Connect a screen with HDMI, keyboard with USB and Ethernet cable to the Raspberry Pi.
6. Connect the micro-USB power cable to the Raspberry Pi. It will now power up.
7. Wait until the Raspberry Pi has started and the text "raspberrypi login" is shown.
8. To log in, write

```
pi
```

9. The password is

   ```
   gruppe11
   ```

   Nothing will happen while typing the password, this is normal.

10. To make the hub script start automatically every time the Raspberry Pi is turned in, write

    ```
    sudo cp rc.local.new /etc/rc.local
    ```

11. Start the hub script, configuration information is needed

    ```
    sudo python angelika-hub/hub/src/hub/hub.py
    ```

    - hub_id: Name of the hub as specified in the server admin interface.
    - password: Password as specified when creating a hub in the server admin interface.
    - server_url: The URL of the server API. During development `https://api.angelika.care/` was used. Remember trailing slash.
    - server_interval: How often the hub should send data to the server, in seconds.
    - server_wait: How long the hub should wait before starting to send data, in seconds.
    - sensor_name: Specify a name, does not matter what. Should describe the sensor used.
    - sensor_type: For the Withings sensor

      ```
      withings_pulseo2
      ```

      must be used.
    - sensor_interval: How often the hub should check the sensor for new data, in seconds.

12. After this the hub should work and a log of what it is doing should be printed to the screen.

13. The hub needs to be restarted. Press

```
CTRL + c
```

and wait for the script to stop.

14. Restart the hub by typing

```
sudo reboot
```

Wait for the Raspberry Pi to boot.

15. When the log is printed to the screen, everything is OK. Disconnect the keyboard and screen.

C.3 Configure a new server

- You should have a server running Ubuntu (preferably the latest version)
- Install python-dev, python 2.7.x, pip, virtualenv, uwsgi, nginx, postgres, openssl
- `git clone https://github.com/sigurdsa/angelika-api.git`

  `/srv/www/angelika-api`
- `cp /srv/www/angelika-api/api/settings/local.py.example`

  `/srv/www/angelika-api/api/settings/local.py`
- `git clone https://github.com/iver56/angelika-web.git`

  `/srv/www/angelika-web`
- `cp /srv/www/angelika-web/js/static.js.example`

  `/srv/www/angelika-web/js/static.js`
- Set the correct API URL in `/srv/www/angelika-web/js/static.js`
- Set up the postgres database (call it angelika)

- In local.py, set DEBUG and TEMPLATE_DEBUG to False, configure the postgres database settings, insert a random Message-Digest algorithm 5 (MD5) hash for SECRET_KEY and CRON_KEY (the two hashes must be different), specify values for CORS_ORIGIN_WHITELIST and ALLOWED_HOSTS

- In `/srv/www/angelika-api/` run make install (this will install all the dependencies of angelika-api)

- In /srv/www/angelika-api/ run

  ```
  make migrate
  ```

  (this will set up the postgres database "angelika" with the correct structure)

- In /srv/www/angelika-api/ run

  ```
  python manage.py collectstatic
  ```

  to copy static files to the static directory

- In `/srv/www/angelika-api/` create the media directory if it is not present

- Create a superuser for Django

- Make sure you have a domain for your server. Buy one if you do not. Set up a sub domain "api" for the API.

- Set the correct API URL in `/srv/www/angelika-web/js/static.js`

- Configure uWSGI[1]

- Generate/get and cat certificates for SSL. This is needed for the API, but not necessarily for the front-end.[2]

- Configure Nginx to serve the front-end and the API. The API must be accessible only over HTTPS, for security reasons.

[1] See this guide for tips about how to configure Nginx and uWSGI: `http://uwsgi-docs.readthedocs.org/en/latest/tutorials/Django_and_nginx.html`

[2] The following guide may help with setting up SSL: `http://www.westphahl.net/blog/2012/01/03/setting-up-https-with-nginx-and-startssl/`

- Make sure that all the files in /srv/www/ are owned by www-data. Run

  ```
  sudo chown -vR www-data:www-data /srv/www/
  ```

- Log in to Django Admin and add groups with the following names: admins, health-professionals, hubs, patients

- Set up a SetCronJob service (see `https://www.setcronjob.com/`) to post to https://<API sub domain>/motivation_texts/delete_old/?cron_key=<CRON_KEY >every night

Appendix D

Templates

D.1 Agenda

Figure D.1 shows the template for the meeting agenda document.

D.2 Status report

Figure D.2 shows the template for the weekly status report document.

D.3 Time sheet

Figure D.3 shows the template for the time registration spreadsheet with example data.

The *Sum* column shows the total amount of hours on that row, and the *Sum since start* column accumulates all the sum cells from this row upwards, showing how many hours have been spent since the start of the project. *Expected workload since start* shows how many hours the team should have worked since the start of the project, in order to fulfil the workload requirement for the project. The *Diff* column shows the difference between the columns *Sum since start* and *Expected workload since start*.

Figure D.4 explained the codes used in the time sheet.

Agenda

Group 11: Angelika

Group/advisor/customer meeting

Time: YYYY-MM-DD HH:mm-HH:mm

Chairperson: Martin Solheim Secretary: Iver Jordal

Agenda:

1. Issue 1
2. Issue 2
3. Issue 3

Figure D.1: Agenda template

Status report - week xx

[Summary of work done this week.]

Front end:

- Implemented functionality 1
- Implemented functionality 2
- Implemented functionality 3
- ...

Hardware:

- Implemented functionality 1
- Implemented functionality 2
- Implemented functionality 3
- ...

Backend

- Implemented functionality 1
- Implemented functionality 2
- Implemented functionality 3
- ...

Documentation

- First section documented
- Second section documented
- Third section documented
- ...

Figure D.2: Status report template

Week	Date	Iver	Torgeir	Martin	David	Lars Tore	Sigurd	Tan	Sum	Sum since start	Expected workload since start	Diff
35	26.08.2014	LE 2	LE 2	LE 2	LE 2	LE 2	LE 2	LE 2	14	14	32	−18
	27.08.2014	PL 4	PL 4	PL 4	PL 4	PL 4	PL 4	PL 4	28	42	64	−22
		SS 4	SS 4	SS 4	SS 4	SS 4	SS 4	SS 4	28	70	64	6
	28.08.2014	MC 1	MC 1	MC 1	MC 1	MC 1	MC 1	MC 1	7	49	96	−47
		PS 3	PS 3	PS 3	PS 3	PS 3	PS 3	PS 3	21	70	96	−26
		MI 2	MI 2	MI 2	MI 2	MI 2	MI 2	MI 2	14	84	96	−12
		PL 3	PL 2	PL 3	PL 2	PL 2	PL 1	PL 2	15	99	96	3
	29.08.2014	RS 4	RS 4	RS 4	RS 4	RS 4	RS 4	RS 4	28	127	128	−1
		PR 1	PR 1	PR 1	PR 1	PR 1	PR 1	PR 1	7	134	128	6
	30.08.2014								0	134	128	6
	31.08.2014								0	134	128	6
36	01.09.2014								0		160	
									0		160	
									0		160	
	02.09.2014								0		192	
									0		192	
									0		192	
	03.09.2014								0		224	

Figure D.3: Time sheet template with example data

Code	Category
LE	Lectures
SS	Self study
MI	Meetings, internal
MC	Meetings with customer
MA	Meetings with advisor
MO	Meetings, other
PL	Planning
PS	Pre-study
RS	Requirements specification
DE	Design
PR	Programming
DO	Documentation
PD	Presentation / demonstration

Figure D.4: Time sheet codes

Bibliography

[1] (2014, November 5). *iOS8 Health* [Online]. Available: `https://www.apple.com/ios/whats-new/health/`

[2] (2014, November 2). *e-Health Sensor Platform V2.0 for Arduino and Raspberry Pi [Biometric / Medical Applications]* [Online]. Available: `http://www.cooking-hacks.com/documentation/tutorials/ehealth-biometric-sensor-platform-arduino-raspberry-pi-medical`

[3] D. Pierce (2014, April 14). *Samsung Galaxy S5 review* [Online]. Available: `http://www.theverge.com/2014/4/14/5608222/samsung-galaxy-s5-review`

[4] (2014, November 5). *iHealth* [Online]. Available: `http://www.ihealthlabs.com`

[5] (2014, November 2). *For Developers* [Online]. Available: `http://www.angelsensor.com/wristband/developers-2/`

[6] (2014, November 2). *UP System* [Online]. Available: `https://jawbone.com/up#system`

[7] (2014, November 2). *UP for developers* [Online]. Available: `https://jawbone.com/up/developer/endpoints`

[8] (2014 November 2). *Pulse O_X* [Online]. Available: `http://www.withings.com/eu/withings-pulse.html`

[9] (2014 November 2). *Withings API developer documentation* [Online]. Available: `http://oauth.withings.com/api`

[10] (2014 November 2). *Paspberry Pi* [Online]. Available: `http://www.raspberrypi.org`

[11] I. Sommerville, "Software Processes" in *Software Engineering, ninth ed.* Boston, MA: Pearson, 2011, pp. 29-32.

[12] L. Bass *et al.*, "Modifiability" in *Software Architecture in Practice, third ed.* Boston, MA: Pearson, 2013, pp. 121-123.

[13] L. Bass *et al.*, "Availability" in *Software Architecture in Practice, third ed.* Boston, MA: Pearson, 2013, pp. 79-99.

[14] L. Bass *et al.*, "Modifiability" in *Software Architecture in Practice, third ed.* Boston, MA: Pearson, 2013, pp. 117-128.

[15] L. Bass *et al.*, "Usability" in *Software Architecture in Practice, third ed.* Boston, MA: Pearson, 2013, pp. 175-183.

[16] L. Bass *et al.*, "Architectural Tactics and Patterns" in *Software Architecture in Practice, third ed.* Boston, MA: Pearson, 2013, pp. 205-210

[17] (2014 November 2). *Python* [Online]. Available: `https://www.python.org`

[18] (2014, November 2). *Smart Devices* [Online]. Available: `http://www.bluetooth.com/Pages/Bluetooth-Smart-Devices.aspx`

[19] (2014, November 2). *Understanding ZigBee* [Online]. Available: `http://www.zigbee.org/About/UnderstandingZigBee.aspx`

[20] The Editors of Encyclopædia Britanica (2014, November 2). *USB* [Online]. Available: `http://global.britannica.com/EBchecked/topic/1056046/USB`

[21] (2014, November 2). *About HL7* [Online]. Available: `http://www.hl7.org/about/index.cfm?ref=nav`

[22] (2014 November 3). *The Open Source Definition* [Online]. Available: `http://opensource.org/osd`

[23] (2014 October 12). *Velferdsteknologi(VFT)* [Online]. Available: `http://www.trondheim.kommune.no/velferdsteknologi/`

[24] (2014, November 3). *Git* [Online]. Available: `http://git-scm.com/`

[25] (2014, November 3). *GitHub* [Online]. Available: `http://github.com`

[26] (2014, November 3). *Disk* [Online]. Available: `http://drive.google.com`

[27] (2014, November 3). *Calendar* [Online]. Available: `https://www.google.com/calendar`

[28] (2014, November 3). *Trello* [Online]. Available: `https://trello.com/`

[29] (2014, November 3). *Facebook* [Online]. Available: `https://www.facebook.com/`

[30] (2014, November 3). *Doodle* [Online]. Available: `https://doodle.com`

[31] (2014, November 3). *The MIT License (MIT)* [Online]. Available: `http://opensource.org/licenses/MIT`

[32] (2014, November 5). *Attribution-NonCommercial 4.0 International*. Available: `http://creativecommons.org/licenses/by-nc/4.0/`

[33] (2014, November 3). *Shaping Up with Angular.js* [Online]. Available: `https://www.codeschool.com/courses/shaping-up-with-angular-js`

[34] (2014, November 3). *Angular JS Lessons* [Online]. Available: `https://egghead.io/technologies/angularjs`

[35] (2014, November 3). *Golden Layout* [Online]. Available: `https://golden-layout.com`

[36] (2014, November 4). *DjangoUser* [Online]. Available: `https://docs.djangoproject.com/en/1.7/topics/auth/`

[37] (2014, November 5). *PyCharm* [Online]. Available: `https://www.jetbrains.com/pycharm/`

[38] (2014, November 6). *Django Unit Tests* [Online] Available: https://docs.djangoproject.com/en/dev/internals/contributing/writing-code/unit-tests/

[39] R. Bergstrøm *et al.*, "Anbefaling på valg av standarder/rammeverk for velferddsteknologi," The Norwegian Dir. of Health, Oslo, Rep., Jun. 2014.

[40] *Ergonomics of human-system interaction – Part 210: Human-centred design for interactive systems*, ISO 9241-210, 2010.

[41] (2014, November 7). *Factory Method Design Pattern* [Online]. Available: `http://sourcemaking.com/design_patterns/factory_method`

[42] (2014, November 10). *POP Prototyping On Paper* [Online]. Available: `https://popapp.in`

[43] (2014, November 10). *Pidoco* [Online]. Available: https://pidoco.com/en

[44] (2014, NOvember 11). *WebStorm* [Online]. Available: https://www.jetbrains.com/webstorm/

[45] (2014, November 11). *Code Style* [Online]. Available: http://docs.python-guide.org/en/latest/writing/style/

[46] (2014, November 14). *Designhjelpen* [Online]. Available: http://designhjelpen.com/

[47] (2014, November 16). *gzip* [Online]. Available: http://www.gzip.org/

[48] (2014, November 16). *nginx* [Online]. Available: http://nginx.org/en/

[49] (2014, November 16). *Cron* [Online]. Available: http://www.unixgeeks.org/security/newbie/unix/cron-1.html

[50] (2014, November 17). *HL7 Norge* [Online]. Available: http://www.hl7.no/

[51] (2014, November 17). *SDFormatter* [Online]. Available: https://www.sdcard.org/downloads/formatter_4/

[52] (2014, November 17). *Win32DiskImager* [Online]. Available: http://sourceforge.net/projects/win32diskimager/

[53] (2014, November 17). *Debian* [Online]. Available: https://www.debian.org/index.nb.html

[54] (2014, November 17). *Raspbian* [Online]. Available: http://www.raspbian.org/

[55] (2014, October 7). *CE Marking* [Online]. Available: http://ec.europa.eu/enterprise/policies/single-market-goods/cemarking/index_en.htm

[56] (2013, March 22). *Equipment Authentication* [Online]. Available: http://transition.fcc.gov/oet/ea/procedures.html

[57] (2011, November). *Velferdsteknologi i Trondheim kommune - Handlingsplan* [Online]. Available: http://www.trondheim.kommune.no/content/1117730318/Handlingsplan-velferdsteknologi-pdf

[58] (2014, November 18). *HL7 Watch* [Online]. Available: `http://hl7-watch.blogspot.no`

Acronyms

API Application Programming Interface. 3, 26–29, 31, 34, 38–40, 87, 93, 95, 103, 107, 109–112, 121–123, 125, 133–136, 152–154, 187–190

CRUD Create, Read, Update, Delete. 152

CSS Cascading Style Sheets. 34, 35, 37, 123, 124

DOM Document Object Model. 34, 35

GB Gigabyte. 186

GUI Graphic User Interface. 84

HDMI High-Definition Multimedia Interface. 186

HL7 Health Level 7. 9, 32, 33, 40, 45, 47, 56, 59, 60, 68, 154, 196

HP Healthcare Professional. 6, 50

HTML HyperText Markup Language. 18, 34–37, 42, 135, 136

HTTP Hyper Text Transfer Protocol. 48, 110, 111, 122, 133–135, 153

HTTPS HTTP Secure. 48, 61, 72, 130, 135, 189

IDE Integrated Development Environment. 38, 41

IP Internet Protocol. 111

JS JavaScript. 36, 42

JSON JavaScript Object Notation. 95, 122, 133, 136, 152

MD5 Message-Digest algorithm 5. 188

MP3 Moving Picture Experts Group, Audio Layer III. 136

MVC Model-view-controller. 82, 84, 87, 92

NFC Near Field Communication. 45

NID National Identification Number. 46, 47, 52, 69, 108

NTNU Norwegian University of Science and Technology. vii, 4, 96, 135, 153, 157

PEP8 Python Enhancement Proposal 8. 19, 161

POP Prototyping On Paper. 95–99

REST Representational State Transfer. 38, 40, 41, 95, 121, 133, 152

SD Secure Digital. 185, 186

SPA Single-page Application. 18, 34, 35, 40

SQL Standard Query Language. 123

SSH Secure Shell. 111

SSL Secure Sockets Layer. 135, 153, 189

TK Trondheim Kommune. vii, 3–5, 14, 15, 145, 154, 164

UAT User Acceptance Testing. 78, 79, 136, 144, 155, 167

UI User Interface. 36, 95, 134

UNIX Uniplexed Information and Computing Service. 122

URL Uniform Resource Identifier. 33, 34, 90, 122, 136, 153, 187–189

USB Universal Serial Bus. 32, 40, 45, 87, 185, 186

UTC Coordinated Universal Time. 122

UX User Experience. 34

WSGI Web Server Gateway Interface. 153