A customizable dashboard platform for caregivers to aid telemonitoring of chronic diseases

Master thesis Computer Science

Dennis Cardinaels

Promotor: Professor Mieke Haesen

Assistant: Jens Brulmans

2017 - 2018

Preface

TODO

Abstract

TODO

Contents

1 Introduction

Due to the continuous growth of ubiquitous computing, the possibility of continuing rehabilitation at home rises. The caregiver monitors the patient at a distance and if needed, the patient is contacted. As a result, resources on both the caregiver and patient side can be saved. This process is called telemonitoring. Research primarily investigated telemonitoring for chronic disease management. Because of the enduring nature of these conditions, patients report the status of their condition using a device such as a smartphone and a digital blood pressure monitor.

The consulted telemonitoring studies resulted in the development of many software platforms, with each trying to monitor patients at home as effective as possible. However, there was never mention of using or expanding an existing platform, only of existing measuring devices. Using different applications to monitor, for example, diabetes and cardiovascular diseases would cause redundancy as parameters such as blood pressure are important for both disease groups. These values need to be updated at two locations, unless they share a common back end system. However, in research applications this is never the case. Data duplication raises several issues: more storage space, data needs more maintenance, and integration becomes increasingly difficult. If a certain workflow requires many applications, time is already lost by switching between them. Therefore, it can be beneficial if all health data is stored and monitored at a single location.

Most health care institutions have transitioned to electronic patient records. Software solutions interact with these records to automatically generate statistics, decision support, and many types of alerts. Monolithic software applications are often used for more general care while specific software applies to specific types of care. There are benefits and drawbacks to both, but no literature was found which reported on the integration of the specific applications with a monolithic system. In a perfect world, a single system provides all the required functionality, as was implied in the previous paragraph.

Another gap in health software research is the effect of customization. Each patient is different and a customized rehabilitation trajectory is preferred. Current health systems provide a general view for all patients where information is buried behind many screens and menus. For example, a patient who suffers from cardiovascular disease visits his general practitioner on a regular basis. Each visit the practitioner has to navigate to the data relevant to the disease. Should the software allow clinicians to indicate what is of importance, relevant information can be reached much faster. For example, a dashboard-like screen is generated which gathers all important data as indicated by the caregiver at one location. This dashboard can, for example, generate summaries while details are one click away.

The main research topic of this thesis is the effect of customization and system integration on health software. One can speculate that the presence these two components increases the efficiency of care delivery. On one hand, clinicians save time by having easy access to their necessary tools. On the other

hand, system administrators can easily expand a central software platform with, for example, 3rd party components. Also, maintenance should be easier as all health data is centralized. By saving man-hours, the expenditure of health institutions reduces. For patients, less time is spent at visits, which in turn lowers the costs.

This thesis proposes a dashboard design that allows the caregiver to customize its functionality according to the patient's needs in terms of treatment. Its components will mainly focus on chronic disease management for use in telemonitoring, due to customization possibilities and the amount literature that exists on this topic. The architecture of the dashboard prototype tries to facilitate easy integration. It should be simple to add, update, and remove functional components without changing the system at its core. After reading this thesis, an external developer should be able to create his own components and plug them into the platform.

First, to get a thorough understanding of related subjects, background information is given in section 2. The discussed topics include electronic health record systems, chronic disease management, a literature review on telemonitoring, dashboard design, and health data privacy and standards. Based on this background information a dashboard design is proposed (section 3). This section highlights the design process and the made choices. Also, personas, scenarios, and paper mockups give a first indication of how the system will be used. Implementation of the dashboard started after the design process. In section 4 the used frameworks and changes compared to the paper mockups are described. The realization of customization and integration is also covered in this section. As soon as the implementation was finished, a usability test was conducted. Section 5 describes the research questions, the test setup, the created documents, and information concerning the testers. The results of the test are discussed in the following section. Finally, the thesis concludes with a summary of its findings and steps for future work.

2 Background

A literature study has been conducted to situate the thesis topic. Multiple subjects lay at the base of the design mentioned in section 3. Because this design describes a health platform, a general overview of electronic health record systems is given first. The precise topics are the transition to, the components of, and the types of these systems. Next three key subjects are explained in the following order: chronic diseases, telemonitoring, and dashboard design. The last section briefly mentions the issues surrounding medical data privacy and standards.

2.1 Electronic health record systems

Health information systems are becoming an important part of health care. They support patient care as well as administrative and financial tools. At the heart of these systems lies the electronic health record. An electronic health record (EHR) is a repository of electronically maintained information about an individual's health status and health care, stored such that it can serve multiple legitimate uses and users of the record [1]. An electronic health record system (EHRS) provides tools to manage and interact with these records. These tools include reminder generation, data analysis, and decision support. It helps the clinician to organize, interpret and react to medical data.

The first section describes the transition from paper-based records to digital and its implications for the medical world. A summary of the main components present in an EHRS is described in the next section. The functionality of an EHRS can be categorized into two types: a monolithic system tries to provide more general care, while smaller and more specialized systems cater to more specific types of care. The last section compares the two types and highlights the benefits and drawbacks of both.

No existing EHRS were reviewed. Access to these systems is limited to open-source solutions as most are commercial and sometimes off-the-shelf products. Consequently, reviews of these systems are difficult to find as institutions only purchase software after thorough review of its features. Only one article was found which compared three open-source health systems [2]. However, no literature was found for proprietary systems.

2.1.1 Moving away from paper

For modern medicine, traditional paper-based medical records are not suited for today's world filled with technology. The drawbacks of information on paper are obvious when compared to digitally stored information.

Functionality Digital records allow systems to aggregate, process and create statistics of the data it contains. For example, a system can generate heart rate graphs with statistics, summarizing many values. If the user hovers over a data point in the graph, the exact value is shown. Printed tables and graphs showing

the same data lack this interaction. Also, paper records demand more effort from clinicians as data is often spread across multiple dossiers.

A paper-based medical dossier can store for example medical images, such as x-rays. Compared to a digital image, paper loses a lot of detail. As such, most multimedia types can only be stored digitally and not on paper. Tools to interact with these file types are provided by an EHRS.

In terms of data input, an EHRS can detect and prevent false data input. The system ensures that all necessary data fields are filled in and in the correct format. This results in more complete and accurate data gathering with fewer errors, increasing the information quality.

Information quality A summarizing paper noted that the use of EHRS leads to more complete, accurate, comprehensive, and reliable data compared to paper-based records [3]. As mentioned in the previous paragraph, a digital system can impose rules on data fields to avoid missing or entering the wrong data. In terms of comprehensibility, poor handwriting leads to wrong or loss of information, which digital systems avoid. Medical instruments can save measurements immediately into system, avoiding copying by hand.

Accessibility Paper records are difficult to access because most of the time only one copy exists. Therefore, transferring these records to other branches or institutions is difficult as the record needs to be located in the often large medical dossier. Also, misfiling, flooding or fires lead to irrecoverable loss of the data. Creating backups of the digital records avoids the last issue, but difficulties surrounding data transfer are still present. Most institutions have their own database structures which hinders the transfer of raw medical data. To remedy this, interfaces are created by the IT staff to interpret and store the data. Sending for example PDF-reports via email is a convenient alternative, although limited in functionality comparable to an actual paper record. Therefore, integration capabilities and the use of standards (section 2.5.1) are important for an EHRS.

Moving data from paper to a digital format requires a lot of manual work. While automation is possible, such as scanning and processing the paper forms with software, verification is still necessary. All the data fields from the documents need to find a place in the system, which is difficult to achieve. Extra diagnostic information scribbled all over the the document, will be lost during this process. Also, what do we do with unreadable data due to poor handwriting? What happens with two forms which contain partly the same information? Do we save the information twice or do we add extra checks to prevent duplication? Because of these reasons, adopting an EHRS is a significant undertaking for an institution. As such, the benefits are not immediately apparent.

Because digital storage increases accessibility significantly, security measurements have to be taken. If not, data can be stolen, deleted or even altered, which means a breach of privacy (section 2.5.1). This also adds to the complexity of developing EHRS.

Efficiency An important task of an EHRS is to facilitate effective care. An early study saw a 6% increase in productivity when EHRS were deployed in health care institutions [4]. However, other factors are at play, such as the adoption rate. The transition from paper-based documenting to digital is accompanied by a learning curve, which can be steep. After the switch, the productivity will most likely be lower at the start. As the users become more experienced with the software, it will increase. Today most institutions have already made the transition to an EHRS, so this has become less of an issue.

2.1.2 Components of an EHRS

As mentioned before, EHRS do not simply store patient records. They consist of many components which ultimately defines how well they perform in health care. Literature defines many lists of essential components. However, we define the following five components as key [1]:

Integrated view of patient data An EHR must allow storage of a wide range of data types. This can be text, numbers, images, video, and others. Some data can still be on paper due to lacking support of the EHRS, as mentioned in section 2.1.1. To display more complex data types, such as x-ray images, standards are used. A brief overview of medical standards is described in section 2.5.2.

Clinician order entry The point at which the clinician enters treatment instructions is called order entry. An order entry system assists the clinician during the decision-making process to ensure that the instructions are correct. It also reduces errors and costs compared to paper order entry for the same reasons already mentioned in section 2.1.1.

Clinical decision support A decision support system embed into an EHRS aids the clinician by suggesting certain actions when certain situations occur. If for example, a patient is due for vaccination, the system notifies the clinician by presenting a constructed order which needs to be confirmed or denied. The system can do this for a bulk of patients, so manual checkups are not required, thus saving time.

Access to knowledge resources During the writing of notes or orders for a patient, clinical questions often arise. Instead of asking colleagues or searching through multiple manuals, the EHRS searches for relevant literature to address the question. Due to the internet, a very large source of information is readily available.

Integrated communication and reporting support Communication is an important part of health care. Often clinicians spread across multiple institutions provide care for the same patient. Communication, therefore, directly

affects the effectiveness of patient care. Tools that simplify this process are essential for a health system.

Most institutions are bound by their own EHRS. If data resides in another institution's health system, then access has to be requested. Health Information Exchange removes the need to manually ask for data access as institutions are able to reach beyond their own system. This, however, needs to be supported by the institution.

Throughout the years, many EHRS have been developed which may or may not integrate all of the above components. Should an essential component be missing, an institution may add another software system. This has its own benefits and drawbacks.

2.1.3 Monolithic vs. multiple EHRS

Health care institutions can opt for a monolithic EHRS or combine multiple EHRS to achieve all required functionality. There advantages and drawbacks to both [5]. Elements that influence this choice include IT infrastructure, safety risks, the volume of care, and frequency with which patients move facilities. It is possible that a single EHRS does not satisfy the requirements of an institution. A reason for this is that certain branches require specially tailored software for their clinical practice, which the current EHRS in use lacks.

Functionality wise, a monolithic EHRS tends to appeal to more general types of care, whereas it lacks in very specific ones. Also, vendors that offer these all-in-one solutions tend to have less experience with these special types of care which reduces the chance it will be added to the system. Vendors of specific EHRS do have this expertise and can tailor the system to the needs of the customer. In this case, combining a system that supports general care with special care systems seems like the best choice. However, other factors have to be considered.

The advantage of a monolithic system is that the data it uses is centralized. This ensures that all data is accessible anywhere throughout the system and is easier to maintain. When multiple systems are in place, data has to be exchanged between them. As a result, searching for data is more difficult. As mentioned in section 2.1.1, due to different data structures, data exchange is difficult. To solve this issue, an intermediate EHRS can be developed. This system serves solely for the purpose of data collection and transformation. All other systems search for data in this intermediate system. However, this leads to another system, requiring additional costs, development effort, and maintenance.

2.2 Chronic diseases

In order for the dashboard to be effective for chronic disease management, it is important to know what a chronic disease is, together with its prevention and

management strategies. A chronic disease can be defined as a long-lasting human health condition. If the length of the condition is at least three months, the term chronic is used. The World Health Organization defines four major categories of chronic diseases: cardiovascular diseases, cancer, diabetes and chronic obstructive pulmonary disease [6]. Cardiovascular disease is the leading cause of death worldwide [7]. Combined with cancer, they accounted for 46% of all deaths in the United States in 2015 [8]. Also, 86% of all health care spending in 2014 was for patients with one or more chronic conditions [9].

We now define each disease category. All diseases that involve the heart or blood vessels belong to the cardiovascular disease group. Typical examples of such diseases are stroke and coronary artery disease. Cancer has many variants, but all of them share the same behavior. This is when some cells of the human body divide without stopping and that it spreads to surrounding tissues [10]. Diabetes is a disease group in which there are high blood sugar levels present for a prolonged period. There are three types of diabetes: type 1, type 2, and gestational diabetes. Last, chronic obstructive pulmonary disease (COPD) is a chronic lung disease that is characterized by persistent respiratory symptoms and airflow problems [11]. Examples of these symptoms are shortness of breath and coughing. An acute exacerbation means that these symptoms periodically worsen.

2.2.1 Prevention

The factors to lower the incidence of chronic diseases mainly involve lifestyle and diet changes [12]. Now, we summarize the recommended changes:

- Avoid smoking: prevents cardiovascular disease, diabetes type 2, cancer, and COPD. Secondhand smoking also contributes significantly to the risk of chronic diseases [13].
- Maintain a healthy weight: prevents cardiovascular disease, diabetes type 2, and cancer. One should aim for a BMI between 18.5 to 25, ideally less than 23. The next two items help in achieving this.
- Maintain daily physical activity and limit time spent sedentary: prevents cardiovascular disease, diabetes type 2, and colon and breast cancer. A healthy goal is to perform physical activity for at least half an hour a day. Sedentary activities such as watching television should be limited to a maximum of two hours a day.
- Eat a healthy diet: prevents cardiovascular disease, diabetes type 2, and some types of cancer. The following factors attribute to a healthy diet: consume healthy fats, eat plenty of fruit and vegetables, replace refined grains with whole grains, limit sugar intake, limit excessive calories, and limit sodium intake.

The aforementioned lifestyle and dietary changes are closely coupled as they affect one another. Eating healthy is an important step of losing weight. How-

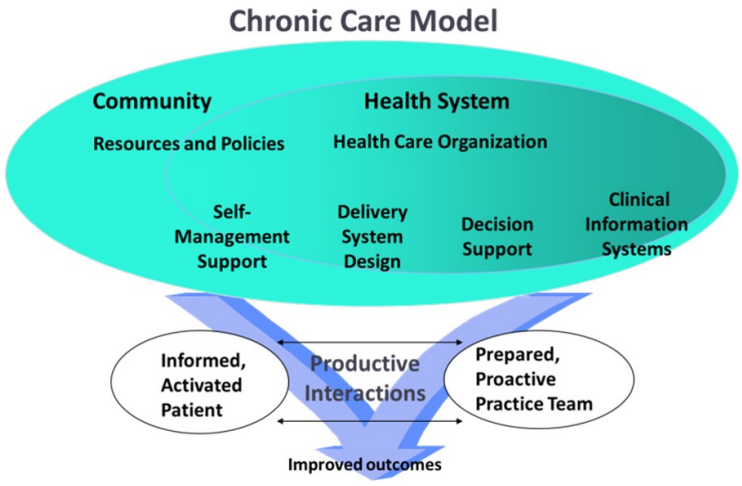

Figure 1: The Chronic Care Model.

ever, solely changing diet without exercising hinders the speed at which weight is lost.

Although one can argue that the incidence of chronic diseases largely depends on genetics, it was estimated that 90% of cardiovascular disease can be prevented [14]. This calls for the need to make the public aware of implementing these lifestyle changes. Programs that promote cycling or walking to school or the workplace are, for example, a manner to engage in physical activity.

2.2.2 Management

The same lifestyle and dietary changes mentioned in the previous section are applied by the patients during the management phase, paired with a medication plan. On a broader scale, the Chronic Care Model was designed to improve the quality of chronic disease management for primary care [15]. The model recognizes six essential elements in chronic care (figure 1):

- Community resources and policies: mobilize community resources to meet the needs of patients. Education classes, self-help groups, and exercise programs are examples of such resources.
- Health care organization: create an organization that provides safe and high quality care.
- Self-management support: patients need to be trained and given the necessary tools to effectively self-manage their chronic disease. This is an important element of telemonitoring (section 2.3).
- Delivery system design: alter the structure of the medical practice to ensure effective care and self-management support.

- Decision support: promote care consistent with scientific data and patient preferences.
- Clinical information system: organize data to facilitate effective care.

Improvements in chronic disease management and reduced health care costs have been observed after implementing this model [16]. Of these elements, most studies focus on self-management support interventions [17]. Here, a key aspect is patient education as it affects the patient's perception of the effectiveness of the intervention [18]. Self-management is heavily facilitated through telemonitoring, which the next section covers.

2.3 Telemonitoring

Monitoring patients at a distance through the use of information technology is called telemonitoring [19]. Here, the patient uses special monitoring devices to gather data which in turn is sent to the health care provider. While many health conditions can be monitored, telemonitoring is most prevalent in chronic disease management discussed in the previous section. A study showed that the complications of chronic diseases and costs can be reduced by the use of telemonitoring. Also, care can be provided without the use of hospital beds and the time spent managing the disease is lower [20].

To get a better understanding of telemonitoring and which parameters are measured, an overview of its application to chronic disease management is given. Second, the difficulties surrounding telemonitoring are mentioned.

2.3.1 Chronic disease management

A lot of studies have been conducted to test the effect of telemonitoring on chronic disease management. For all but one of the four disease categories mentioned in section 2.2, general literature was found: the literature found on cancer focussed on very specific types of the disease and not in the general sense. Consequently, only the application of telemonitoring on the other categories is discussed.

Cardiovascular diseases One study recruited 390 patients to test the effect of telemonitoring on reducing hospital days and total costs [21]. 193 individuals in the intervention group were to take daily measurements using special hardware. The hardware recorded the weight, blood pressure, heart rate and blood oxygen levels. Diabetes patients received an additional blood glucose meter. Patients were also required to fill in a predetermined set of questions every week. On the clinician's end, nurses were shown a summarizing table indicating if the sent results were good, bad or missing. Based on these results, the nurses decided to contact the patient if extra care was required. The study concluded that telemonitoring appears to facilitate more efficient outpatient care and potentially reduce costs. The total numbers of hospital days for the patients in

the intervention group showed a tendency to be lower and fewer emergency department visits. However, there were more urgent care and primary care visits.

Another research paper reviewed 30 studies and noted that most patients had a positive experience and little difficulty with telemonitoring technology [22]. In total there were more than 9500 patients in the studies. The summary concluded that there were fewer deaths, fewer hospital admissions and improved quality of life. There were also hints of reduced cost, but most studies did not report on this topic. None of the studies reported on the optimal duration of for telemonitoring.

However, the effect of telemonitoring treatment on cardiovascular patients is still unclear. A paper criticized the aforementioned one, stating that the conclusions made were only valid for small studies. For large trail studies no significant effect was found [23].

Diabetes Due to the majority of research reporting on telemonitoring for diabetes type 2, literature concerning type 1 was omitted. Since diabetes is a long-term disorder, combining the management of diabetes with telemonitoring is well documented in research literature.

One study required 32 patients to take blood glucose measurements for nine months using a Bluetooth capable glucose meter. The meter sent the measurements to a mobile phone which in turn forwarded them to a hospital server. Based on the measurements clinicians would review them and send recommendations to the patient and their general practitioner. The results showed a decrease in HbA_{1C} from 8.40% to 7.76%. Higher values indicate poorer blood glucose level control. Due to the low amount of patients, the paper concluded that the benefit of telemonitoring would likely be small [24].

Diabetes also has an effect on blood pressure, as approximately 50 to 80% of diabetes type 2 patients suffer from hypertension which in turn can lead to cardiovascular disease. A target blood pressure level for diabetes patients is 130/80 [25]. One study kept this goal in mind and developed a home telemonitoring program [26]. It is labeled as 'home' due to the fact that a smartphone generates self-care messages for the patient, based on multiple previous readings. The results, however, would still be sent towards the clinician. Careful reviewing the data and corresponding with the patient would not be necessary thanks to the generated messages, ultimately saving the clinician time. There were 55 patients in the telemonitoring group and as well in the control group. Both groups were required to measure their blood pressure using a device, but only the telemonitoring group had the device connected to a smartphone, generating the self-care messages. The results showed that the blood pressure only decreased significantly in the telemonitoring group. Also, the percentage that reached the 130/80 target blood pressure was 51% for the telemonitoring group and only 31% for the control group. Last, it was suggested that the self-caring nature of the program might have negative psychological effects, requiring future study.

Weight loss also influences blood sugar. The Active Body Control Program

reported the effects of telemonitoring in combination with a dietary program [27]. The subjects were obese with diabetes type 2. A device gathered measurements from a weighing scale and an accelerometer. This device regularly sent these measurements to the hospital server. After six months significant improvements in blood sugar and weight loss were observed. No relevant changes were observed in the control group.

Lastly, medication management is also linked to telemonitoring and diabetes. A study asked 64 patients (the telemonitoring group) to measure their blood glucose, blood pressure, and weight on a daily basis [28]. A nurse would receive these measurements and adjust the medication intake five times a week. The control group consisting of 73 patients also took measurements of the same parameters, but they only received a phone call once a month to have their medication schedule adjusted. The results noted a significant decrease in HbA_{1C} for the telemonitoring group after three and six months, compared to the control group.

Chronic obstructive pulmonary disease A 2003 study investigated the feasibility of telemonitoring for patients with COPD [29]. In the first phase of the study 23 patients were observed for 12 months during their medical visits. For the second phase, patients were given a device which uses a finger clip to measure the oxygen saturation and heart rate. This device sends the data towards the caregivers. The caregivers analyzed the measurements and communicated comments on the symptoms and prescription changes to the patients. Results noted decreases in the numbers of hospital admissions (50%) and acute home exacerbations (55%). The hospitalization costs were 17% lower compared to the first phase and 96% of the patients were satisfied with the telemonitoring equipment. However, it should be noted that the sample size was small.

Another study observed the quality of life of telemonitoring compared to usual care [30]. The control group received standard care for 52 weeks. The telemonitoring group received devices asking them questions about their symptoms twice a day. Also, the temperature and oxygen saturation were recorded by the device. If two of these measurements crossed a threshold, the respiratory nurses received an alert and called the patient. Between the two groups there were no significant differences seen in the quality of life, meaning that telemonitoring had no effect on this outcome.

2.3.2 Issues

Many issues still surround the research base of telemonitoring [19]. Due to the small sample sizes present in a lot of studies, results often are inconclusive or not significant. On top of that, some report positive impacts while others seem to observe the opposite. A possible cause of these inconsistencies is the fact that these studies have to deal with many variables. Characteristics such as age, technology experience, and the stage of the disease can all influence the observed results. It is important that these variables are carefully noted and considered when evaluating the results.

The effect of telemonitoring ultimately depends on the adherence of the patient. Many studies report that the adherence of the intervention group was great, but others reported that it decreased over a long period of time. Due to chronic diseases requiring long follow-up, long-term adherence is key for telemonitoring to be successful.

Few studies report on the effects on health care costs. A couple of studies report on the daily costs of a system or program, but long-term analysis was never mentioned. This is an important factor, since telemonitoring needs to be economically viable before health care providers will adopt it. Also, the impact of telemonitoring on the utilization of health services needs to be investigated further. These services include emergency room and clinic visits, hospitalizations, and lengths of stay.

Last, studies widely reported that telemonitoring was well received and accepted by the patients. As mentioned before, variables such as age can influence this. Also, the measuring devices reported accurate data and few technical issues occurred. However, these factors are still important to keep track of.

2.4 Dashboard design

Dashboards are used in many contexts. The most common examples include a car speedometer and stock analysis. In computer networking, dashboards are used to quickly interpret network traffic. In our case, we want to gather all telemonitoring data for a specific patient. To create an effective dashboard, multiple design principles should be kept in mind.

While some dashboards provide fancy graphics, many miss the key point of what they are supposed to accomplish. The primary goal of a dashboard is clear communication, which is achieved through effective design. A formal definition is as follows: "A dashboard is a visual display of the most important information need to achieve one or more objectives; consolidated and arranged on a single screen so information can be monitored at a glance." [31]. This definition contains four key characteristics:

Dashboards are visual displays. It is important that data is represented visually. Displaying charts and other graphics allows more efficient communication compared to textual information. For example, trends and outliers are easier to spot in a line chart compared to a table with the same values. Instead of using a different visualization for the sake of variety, one should always use the visualization which is best suited to effectively communicate the data [32].

Dashboards display information needed to achieve specific objectives. The data that is shown, must be of use for the job that needs to be done. It is possible that the data needs to be gathered from multiple sources and tailored to the specific context so an efficient visualization can be created.

A dashboard fits on a single computer screen. In order to see all information at a glance, scrolling must be prevented. If multiple screens are present, then it is no longer a single dashboard. This leads to another question: what type of display is the information shown on? Due to the wide variety of screen sizes and aspect ratios, a responsive dashboard would be most beneficial.

Dashboards are used to monitor information at a glance. Important data should be immediately noticed, whereas very specific details should be hidden. This means data must be summarized or aggregated in order to be effectively shown. However, if the user wishes to view the detailed data, the dashboard should provide means to do so.

To create an effective dashboard, the user-base must be well known and understood. What type of users are we dealing with? What are their characteristics? These are important questions to ask, as one user may not comprehend one visualization while another one can. This means that the focus should be put on the user [33]. We can ask the following question to better understand the user's needs:

- What metrics does the user need to see?
- What context does each metric require to make it meaningful? Do we need to visualize the variance, target to reach, trend...?
- What visualization best communicates the metric?

On that end, sketches and mockups can significantly help the design process. Multiple iterations, each incorporating feedback from the users, ensure that the end result is satisfactory. In section 3 created paper mockups are showcased together with an explanation on the design choices.

2.5 Data privacy & standards

Although out of the scope of this thesis, data privacy and the use of health standards are important components of health software. Therefore, it is important to mention them. From the onset of the design phase, data security and standard choices need to be kept in mind. When purchasing a commercial system, these aspects should also not be forgotten.

2.5.1 Privacy

Privacy refers to the desire of a person to control the disclosure of personal health and other information [1]. On the other hand, confidentiality is the ability to control the release of personal health information to a care provider under the agreement that the information will not be spread or used further. Security is the protection of privacy and security, which is achieved through policies, procedures, and safeguards.

We can ask several questions related to health information data access and storage: who owns the data? Is it the health provider or the patient? What type and how much of data needs to be stored? Who can read the data? Who can write to the data? Can someone access specific health information without consent? In order to deal with the some of the challenges concerning data access and storage, these questions need to be answered [34].

An important issue surrounding privacy is medical data access. For example, researching the medical data of a large population can uncover looming threats or an epidemic. Also, disease incidence and intervention analysis benefits from data pooling. However, medical data should not be too accessible. As more and more people gain access to health data of the population, the chance that data is misused or confidentiality is broken increases. Therefore, striking a balance between free information access and protection of privacy and confidentiality is difficult. An argument that promotes free access is to anonymize the data by removing identifiers such as the person's name. However, the individual pieces of information can still reveal the their identity.

Compared to paper records, electronic records are easily accessible, such as through the internet. This calls for extra security measures to prevent data breaches. If someone wants to access a medical record, this person must first authenticate himself. In turn, the system checks if the person is allowed to access this information. Regardless of the outcome, the audit system logs the access attempt. Such systems need clear rules and policies defined by the institution on who can access what. Another obvious security measure is data encryption.

To further prevent malicious use of health data, medical staff should follow education and training programs to make them aware of the privacy rules and policies that are in place. In case of privacy violation, the person in question should be punished appropriately.

2.5.2 Standards

When excessive diversity creates inefficiencies and affects effectiveness standards are required [1]. A hospital contains many independent units spread across primary, secondary, and tertiary care. These units use software best fit for their practice and all record different types of data. For example, the hospital admissions system records patient diagnosis, the pharmacy records prescriptions that were handed out, and the laboratory system records test results. Inevitably, transfer of data between these units is required.

To coordinate multiple systems, data transfer is essential. Nowadays too many different systems exist to create point-to-point interfaces for. Standards try to resolve this problem by defining guidelines software system should follow in order to communicate data. An efficient standard requires that data can be easily stored and presented towards the users of an EHRS. Also, security measures, such as authentication and access control, need to be interwoven with these standards.

The use of standards in an EHRS leads to easier integration which in turn leads to lower development costs. Integration of many proprietary data struc-

tures leads to difficult to maintain software and a higher chance of components breaking. New medical devices and software that comply to these standards prevent this.

One of the most used and implemented standard set is Health Level 7. These messaging-based standards are created to exchange clinical and administrative data. Another standard such as DICOM is used for the communication and management of medical imaging information [35]. Medical device interface standards are created by IEEE. Many more standards exist, but these are the more prominent ones.

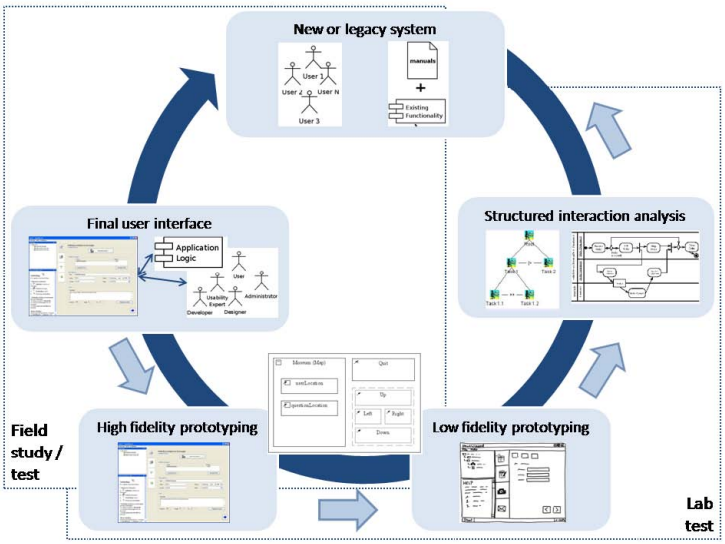

Figure 2: The MuiCSer process.

3 Design

After the literature study, the design phase started. First, the MuiCSer framework is briefly explained as this was used throughout development of the prototype. Next, the dashboard solution was proposed following the literature study. Hereafter, personas and scenarios hint at the functionality and usage of this dashboard. Last, the components of the system, together with design choices, are described and visualized with paper-mockups.

3.1 MuiCSer

This section describes the MuiCSer framework, made for user-centered software engineering processes in a multidisciplinary context [36], which was used to create the dashboard prototype. This framework focuses on optimizing the user experience during the entire software engineering cycle to ensure that the needs of the end user are fulfilled. By combining user-centered design methodologies and software engineering principles, the user experience of the final product can be improved substantially.

3.1.1 Process

The MuiCSer process is summarized in figure 2. After each phase, the result is evaluated, verified and validated to ensure that the required functionality is present. In turn, the received feedback can be used to reiterate over the previous phase. On the figure, this is denoted with the light arrows, while the dark one

represents the overall process direction. What follows is a brief description of each phase:

New or legacy system At the start of the MuiCSer process an existing system in need of improvement is either evaluated or a new one will be designed. This requires an analysis of the tasks and needs of the user, as well as the objects and resources required to perform these tasks. Personas and scenarios are the resulting artifacts of this phase. First, personas describe the personalities of the potential end users including hobbies, skills and the environment they surround themselves in [37]. Its goal is to uncover behavior patterns which can be of use when designing a user interface. Second, scenarios tell stories describing the use of a fictitious system from the point of view of a persona [37]. It tries to sketch the usage of the system for which a design must be made. These artifacts are found in section 3.3.

Structured interaction analysis During this phase, the results of the analysis are used to create task models. These models specify concrete tasks and goals which can be dissected into specific actions or steps the user has to take. These artifacts lay the foundation for designing a user interface which supports these tasks and goals. Task models were not created. However, each component of the dashboard tries to support certain tasks. These are described in section 3.4.

Low fidelity prototyping When the actions have been specified using the task models, low fidelity prototypes are designed. Paper sketches and mockups are such examples and are ideal for visualizing the layout of the user interface. Without spending too much time and resources, presenting such prototypes can yield valuable feedback from the end-user or customer. However, there is no interaction present. Typically multiple versions of these prototypes are created until the customer is satisfied, after which high fidelity prototypes can be developed. For each component description in section 3.4, low fidelity prototypes are presented. If some of these are too similar, then they are found in appendix A.1.

High fidelity prototyping Creating high fidelity prototypes requires a lot more effort compared to low fidelity prototypes, as they offer functionality closely resembling the final product. However, the feedback will be much more valuable as not only design, but also functionality is tested. Section 4 describes the process of creating the prototype featured in this thesis. Choice of frameworks and design changes compared to the low fidelity prototypes are also mentioned in this section.

Final user interface When the latest iteration of the high fidelity prototype satisfies all user requirements, the final user interface can be created. It would be beneficial to reuse the code from the prototype in order to save time and

resources. As a final step, the task models are checked against the interface to check if all required functionality is present. In this case, the high fidelity prototype is the final product of this thesis. As such, this phase was not reached.

3.2 Proposal

The proposal features four core concepts which will be combined to form one cohesive solution: a dashboard, telemonitoring of chronic diseases, customization, and integration. As mentioned in the introduction, multiple applications and integrating them pose several difficulties: data is spread and stored multiple times. Interaction between these applications is initially not present and integration with existing systems requires a significant amount of work from the IT staff of the institution. As more applications are added, this becomes increasingly difficult. To remedy this problem, each application becomes a module which can be plugged into the dashboard system.

A module-based approach towards the dashboard allows easy customization and integration. Each module will serve its own purpose by showing data for one particular health parameter. As mentioned in section 2.3, parameters such as heart rate and blood pressure are often measured to monitor multiple chronic diseases. Integration is handled by the fact that each module is independent and does not require other modules to work properly. This also helps with customization as a clinician can choose which modules are relevant for the patient in question.

The modules that will be developed are for use in chronic disease monitoring. However, modules can be developed for purposes more fit for in-patient care. Examples are real-time monitoring, documenting diagnoses, and viewing of lab results. Which modules will be implemented and what each one of them tries to achieve is described in section 3.4. The implementation section will provide more insight on how this independence is achieved and how future modules can be easily integrated.

This combination allows the clinician to customize the dashboard according to the needs of each patient, facilitating effective care. Data is found rapidly, albeit summarized or detailed, and not hidden away behind multiple screens. Developers should be able to create modules that will fit into the dashboard, negating the use of multiple applications. The following section describes how the system will work and how the users interact with it.

3.3 Personas & scenarios

Based on the proposal mentioned in the previous section, personas and scenarios have been created to get a better understanding of its users and how the system will work. Each of these tries to highlight the problems the users face and how the new system tries to solve them.

3.3.1 Jake

Persona Jake is a 25-year-old male who currently works as a nurse at the hospital in his city. He has been working there for four years and lives alone in his apartment. Still being a young adult, Jake grew up with technology. As a result, he experiences no difficulties when using new software on his computer or smartphone. His hobbies include music, playing the guitar, and video gaming.

Currently, the workflow at the hospital is dated. A new health information system was introduced to summarize and gather all medical data in a singular space. Because this is an all-in-one system, a lot of features that Jake does not need still clutter the screen. Navigating the system is difficult and customization is not present. Jake wishes to only see the features he uses most while hiding the features he does not need.

Scenario Jake starts his first day using the new system by reading the manual that is accompanied with it. He starts the application and for each patient he is presented with a list of modules that can be added to the dashboard. These modules can help with the following: cardiovascular disease, respiratory disease, diabetes, and others.

After selecting a few modules, Jake is presented with a dashboard. The dashboard contains all the wanted functionality of the system, where each "block" represents a module that was added. Jake moves a few blocks around until he is satisfied with the layout. He noticed a module he does not need anymore and promptly deletes it by selecting the option on the module. After a while, Jake got transferred to another department and wants to add a new module to support his new workflow. He opens the module list and selects the he wants, which is then added to the dashboard.

By examining the new module Jake quickly notices that on the dashboard a summary is given. But when Jake clicks on the enlarge button, the module expands to fill a larger area where detailed information is shown. Jake feels he is in control of the system and that it will improve his productivity.

This scenario highlights the benefits of customization. Hiding unwanted components, while adding the useful ones allow for a clear dashboard to be displayed. The user is in control and is able to quickly view data of importance. It is still possible to view detailed data.

3.3.2 Dan

Persona Dan is a general practitioner since he graduated from university. He is on the job for 21 years and he is the preferred doctor in his town. Throughout the years Dan has used a multitude of systems and he always tries new ones to improve his workflow. Because he has been a general practitioner for such a long time, he has 500 patients that visit him at least twice a year. Some patients, especially the elderly, visit as much as once a month.

Most of Dan's patients visit for illnesses such as fever and a cold. To diagnose these illnesses no data is necessarily needed, just a description of the patient's symptoms will suffice most of the time. If Dan performs such a diagnosis, he promptly adds it to the information system and to the electronic health record of the patient.

However, if a patient visits that has a lot of problems regarding his blood pressure, then Dan has to perform a more complex diagnosis using historical data. In the current system that Dan uses, it is very difficult to search for this data. But what bugs Dan the most is that he has to do it every time this patient visits.

Scenario Because of Dan's ongoing curiosity, he tries a new health information system which allows customization for each specific patient. Because the diagnoses of most patients can be very different from time to time, Dan creates a default module group which is displayed for each patient, unless that patient has a specific module configuration. The default module group includes past diagnoses, known allergies, patient information (blood type, last weight, height...), and a current medication list.

The first patient of the day describes what sounds like a fever. Dan confirms this and hands the patient a prescription. The diagnosis is added to the 'past diagnoses' module and the prescribed medication to the 'current medication list' module. Dan did not need other health data to perform the diagnosis. Therefore, Dan does not change the configuration of this patient.

The next patient, an elderly woman, came for her second visit of this month. Dan knows from the past that it will probably be a heart related problem. Dan searches for a 'heart' module and adds it to the configuration of the elderly woman. Now both the default module group and a heart rate module are present, which is unnecessary according to Dan. He removes some modules of the default module group. Now, the next time the elderly woman visits, that configuration will be automatically loaded.

One specific patient had broken his arm three times in less than a year. When the patient came for a routine visit, Dan immediately added a module to easily view x-ray photos and view them in a timeline.

Again, the benefits of system customization for each patient are obvious. Initially, it will take some time to configure each dashboard for all the patients, but the system will try to make this process quick to complete. Once the configuration is done, the time spent during a consultation will decrease.

3.3.3 Emily

Persona Just like Dan, Emily has been a successful general practitioner for quite some time. However, she has different needs of an EHRS. Between each patient visit, there is a period of time in which Emily does not know what happens to patients regarding certain parameters. For example, if a person has to regularly measure his heart rate and blood pressure because of cardiovascular

disease, it is imperative that the doctor is made aware of these values. If Emily sees that these values are not looking well, she can contact the patient to come in for a checkup.

If a patient has sleep issues, Emily wishes to not see detailed measurement values, but regular descriptions of the nights sleep. This includes the hours of sleep, amount of wake-ups, subjective feeling of tiredness. Currently, Emily has no way of regularly receiving this information without having the patient visit.

Scenario Emily recently received notice of a new platform that includes telemonitoring support. Several mobile applications are developed that can send data using an API to the platform, which in turn processes the values and can notify Emily of any anomalies. It includes customization for certain parameters in which Emily can individually assign thresholds for each patient.

A patient who recently had a cardiac arrest is continuing rehabilitation at home. However, the patient has chest pains and pays Emily a visit. She tells the patient to regularly measure his blood pressure and heart rate, and to take note of these values in a mobile application. This application also allows taking general notes, such as feeling pain or having caught a cold. After they have scheduled the next visit, the patient is sent home.

As the next few weeks pass by, Emily is notified that this patient has crossed a threshold regarding his blood pressure. Immediately Emily checks the measured data and sees a graph of all measured values. This dataset delivers insight into the history of the patient and Emily sees that there is currently no need to panic. She decides to not take action and configures a weekly reminder to keep monitoring the blood pressure.

The next week, Emily receives a notification that the patient has made a note. It reads that he experienced chest pains. Again, Emily takes a look at the blood pressure data and sees a worsening trend leading to hypertension. Emily decides to call the patient to schedule an early visit. The system helped Emily to intervene as soon as the situation seemed to worsen while avoiding having the patient visit too early which in turn saves Emily time.

Another patient has sleep issues. Emily encourages the patient to use a sleep monitor, which is a wristband. This device is connected to a smartphone which communicates the quantitative data to the new platform. The application on the smartphone allows the patient to take note of qualitative data such as a general description of the night or what food he ate. One night the patient slept only three hours and took note that he went out and drank a lot of alcohol. This could explain for example the bad sleeping rhythm for the next few days. Emily wishes to keep track of this patient on a weekly basis and configures the platform to notify her.

Here, the effects of telemonitoring are highlighted. By generating reports and alerts, the caregiver can intervene when necessary. Combine this with the aforementioned dashboard platform, greater efficiency of care can be accomplished.

3.3.4 Anna

Persona Anna is a software engineer working at the local hospital. From the early 2000's, she was responsible for the integration of health software systems. As such, she has experienced the continuous change associated with health software. Each unit of the hospital uses different software, while they all share a general system to view patient records. The interaction between these applications is one of Anna's responsibilities.

As time goes on, new software and medical instruments that improve workflow are purchased. Sometimes these replace old systems, otherwise they are added to work alongside them. Both are becoming increasingly difficult to do, as the codebase of the EHRS grows. When an older system gets replaced, Anna has to weed through the code and delete little pieces, hoping other parts wont break. It is a frustrating task as often band-aid solutions are applied.

Scenario A new EHRS was installed that should improve integration of other applications. Anna skims through the documentation and starts the transition process. She quickly notices that for each device or application she needs to create a back end data structure and a module that represents it in the EHRS. The new EHRS is designed that these modules can serve as complete standalone applications or be part of a larger set of modules.

After the initial transition period, everything is put into place. Not soon after, new bedside monitors with accompanying software were purchased. These replace the old bedside monitors and consequently, the software. Thanks to the new EHRS, Anna deletes the old module without much hassle. Also, she reworked the back end data structure to work with the readings from the new device. The data of the old device is ported to the new data structure, so no data is lost. Anna designed the new module to be similar in appearance and functionality. After implementing the module, the staff barely notices any change and workflow resumes with minor disruption.

This persona and scenario highlight the importance of integration. Significant time and effort can be saved if this process is simplified. Also, maintenance is easier due to looser coupling of the modules.

3.4 Modules

As mentioned in the proposal, the dashboard will feature a module-based design. This section describes the modules that will be available in the prototype. For each module, the reasons behind the design choices are given, as well as a low-fidelity paper mockup. Keep in mind that all modules are primarily chosen to aid with chronic disease management, due to the customization possibilities. Each module has a design which is in line with the principles mentioned in section 2.4. For each module we ask the following questions:

- What functionality does the module offer?

- What data will it represent?
- How do we show as much data as possible at a glance?
- How are summaries generated?
- How are the detailed values shown?

Some modules share common functionality. These include resizing the module to a larger or smaller version with varying amounts of detail, increasing the customization options. The smaller the module, the more data is aggregated to a summary. Each module can be freely placed anywhere on the dashboard. Also, all modules can navigate through historical data by either three days, a week, or a month (30 days). For each patient it is possible to define threshold values as this feature was mentioned many times in telemonitoring research (section 2.3.1). This threshold feature is available for all modules in order to generate warnings.

Based on the telemonitoring literature review from section 2.3, the following modules will be created:

- Heart rate: heart rate in rest (BPM). Relevant to cardiovascular disease, diabetes, and COPD.
- Blood pressure: systolic and diastolic (mmHg). Relevant to cardiovascular disease and diabetes.
- Blood sugar: blood glucose level (mmol/L). Relevant to diabetes.
- Weight: kilograms. Relevant to cardiovascular disease and diabetes.
- Oxygen saturation: blood oxygen level (%SpO2). Relevant to cardiovascular disease and COPD.
- Medication: adherence percentage. Relevant to cardiovascular disease and diabetes.

Each module will be described and visualized with a low-fidelity prototype. Charts in these prototypes are very basic to give an idea of its size and placement. The styling of the chart is described in the respective paragraph. Modules that are too similar in design are moved to appendix A.1.

3.4.1 Heart rate

The small summarizing module (figure 3) tries to indicate how the last three, seven, or 30 days went. Three buttons are present to choose the time period. Navigating the time period is done by pressing the arrow buttons. Based on the set thresholds, a count is generated of the measurements: good values lay between the recommended values, warning values lay between the danger and recommended values, and dangerous values have crossed the danger threshold. These are color coded as green, yellow, and red respectively. Below the count,

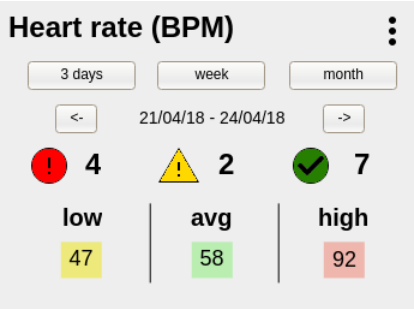

Figure 3: The small heart rate module.

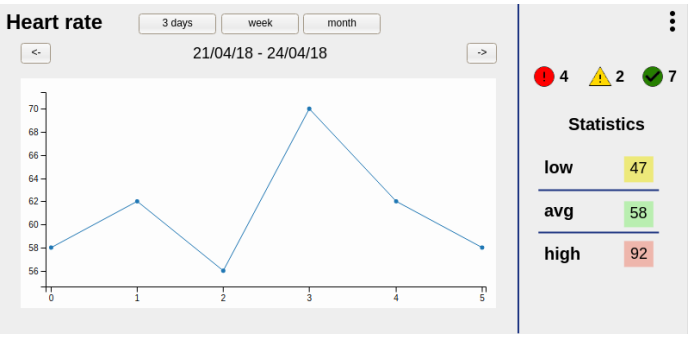

Figure 4: The large heart rate module.

the lowest, average, and highest values for the selected time period are found. These values are again color coded to situate them in the threshold zones. A settings button shows a menu wherein the user can change the thresholds for the current patient. Also, the options to remove or enlarge the module are found here.

A larger module (figure 4) provides more detail. The left hand side features a line chart which plots the heart rate measurements over time. Areas on the chart are highlighted green, yellow, or red to indicate the thresholds. Just as the small module, time periods can be selected. To the right of the chart area, the same count and summarizing values are found as in the small module. A simple list of the values was avoided since all the information is present in the graph, although such a list can be added in the future.

3.4.2 Blood pressure

The functionality and design of both the small and large modules are the same compared to the heart rate module. The only difference is that the chart of the large module (figure 5) displays two lines: one for the systolic blood pressure (BP) and one for the diastolic blood pressure. The threshold zones are also

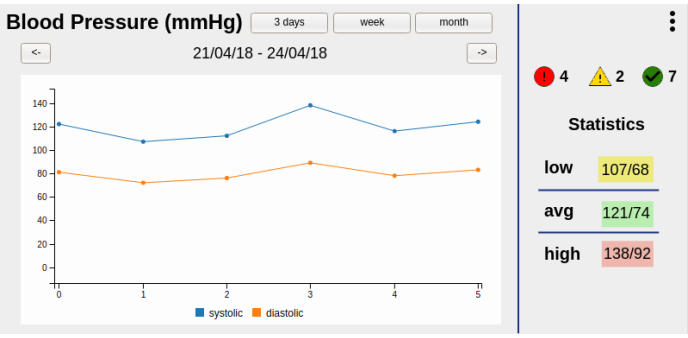

Figure 5: The large blood pressure module.

different. Warnings are shown if either the systolic BP is too high or if the diastolic BP is too low. Since both systolic and diastolic BP change in the same direction, this method should suffice. For example, in a hypertensive state both the systolic and diastolic are too high, not just one. If both BP values are between the systolic and diastolic warning zones, the value is flagged as okay.

3.4.3 Blood sugar

The blood sugar modules are composed the same as the heart rate modules. An issue surrounding blood sugar is the time it was measured: in a fasted state or not. Blood sugar levels differ according to this state. We assume that all blood sugar measurements are taken in a more or less similar setting. It is possible for one patient to measure blood sugar in a fasted state and another patient after a meal, since individual thresholds can be customized.

3.4.4 Weight

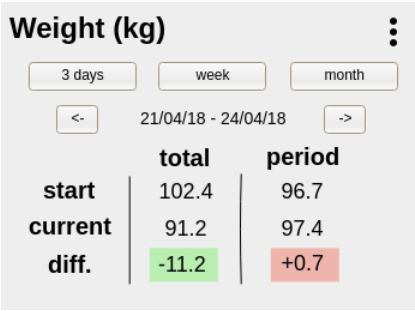

Figure 6: The small weight module.

The weight module shows different statistics. To start, there are no thresholds present, only a goal weight can be set. Progress is visualized based on this value.

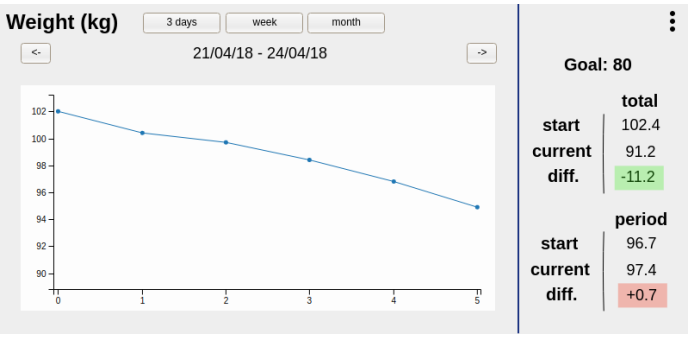

Figure 7: The large weight module.

The small module (figure 6) shows the starting weight (the first measurement ever registered) and the current weight of a patient today (last registered measurement). The difference of these two values is calculated and marked either green or red. If the person is moving closer towards the goal weight, the color will be green and otherwise red. A second column shows the same information, but for values from the selected time period. This allows the clinician to see, for example, the progress made weekly. Similar values are shown in the larger module (figure 7). It features the same layout as the previous large modules. The chart does not show any colors since there are no thresholds. Instead, a line is shown which represents the goal weight. Ideally, the measured values close in towards this line over time.

3.4.5 Oxygen saturation

This module group is also very similar to the heart rate ones. The only small difference is present in the thresholds. Blood oxygen levels are generally very high and only if this value is lower than usual, warnings should be issued. Therefore, there are no thresholds when the blood oxygen level is too high, since higher is better.

3.4.6 Medication

Medication intake is tracked in a very different manner compared to the previous parameters. It is possible that a patient needs to take in multiple medicines in changing amounts over time. The small module (figure 8) shows all the medicines the patient had to take for the selected time period. Precisely, the actual amount the patient took in, the prescribed amount, and the ratio of these two values. Depending on the thresholds, the background will be color coded red, yellow, or green. These thresholds indicate the minimum ratios the patient must abide by.

The chart in the large module (figure 9) shows a line for each medicine the patient has to take in. The y-values are the ratios previously discussed and

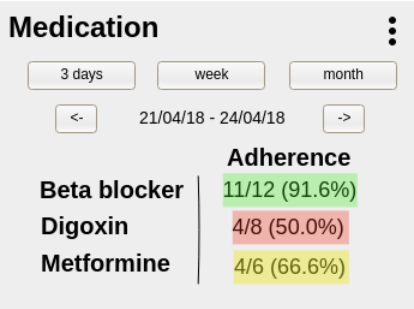

Figure 8: The small medication module.

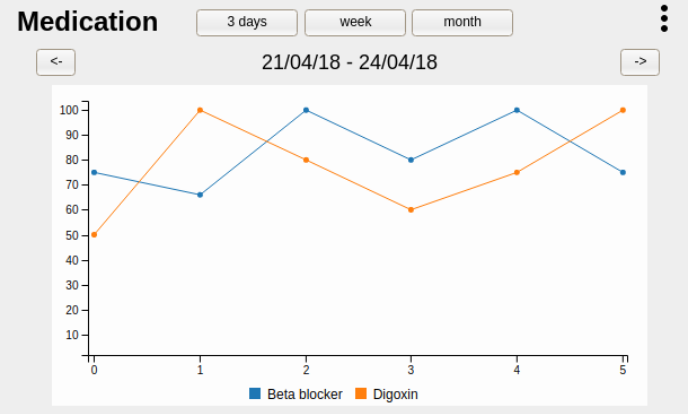

Figure 9: The large medication module.

the threshold regions are colored on the chart. The aggregated values from the small module are not shown as the intake trend is visualized on the chart, making the average intake irrelevant. A legend indicates which line represents which medicine. Hovering over a data point indicates this as well.

4 Implementation

This section describes the implementation process. The design highlighted the usage of the dashboard through its personas and scenarios, while the low fidelity mockups gave an early glance at its appearance. However, at the heart of this application lies the ability to customize and easily integrate new components. To provide these features, a lot of thought went into choosing the best libraries fit for the job.

First, an overview of the global scope of the application is given, after which we go into more detail. The dashboard consists of two major parts common to web development. The back end is described first. This section covers topics such as the chosen frameworks, the REST API, and data storage. For the front end part, the framework which supports our module-based design and its underlying principles are explained. The contribution of both parts towards a customizable dashboard which allows simple integration, is explained further in their respective sections. It is without question that during the implementation small changes were made to the design. These changes are also documented for both parts. During the entire development process, no effort was put towards privacy safeguards and standard support as they are out of the scope of this thesis.

4.1 Overview

The first major choice that was made, was the type of application to develop: a native application or a web application. It was a relative simple choice, but nonetheless an important one. Native applications are developed, in most cases, for one type of operating system. If a developer wants to support both Android and iOS devices, separate applications need to be developed for each of them. Maintaining multiple applications requires a lot more development effort. Java, for example, has the benefit that it is cross-platform thanks to its virtual machine. However, regardless of platform, platform specific changes need to be made to tackle bugs. One of the largest drawbacks of native applications is updating them. This process often requires the reinstallation of the application. As a result, rolling out software updates in hospitals is a difficult task.

Web applications saw a recent surge in popularity. Web applications today are sometimes very similar in user interface and functionality. An example is Office 365 Web Apps, which shares a lot of functionality from its native counterpart. While these web applications also require updates, these happen on the server side. When the user refreshes the web page, the latest version of the application is automatically loaded. This simplifies the roll-out process significantly. Also, all operating systems that support web browsers, are capable of running this type of applications. As a result, developers only have to develop one version. Mobile devices, therefore, are capable of viewing these applications. However, to fit their small screen, developers need to support a flexible layout.

With this in mind, the choice to create a web application was evident. Users can view the dashboard regardless of the device they use, without sacrificing

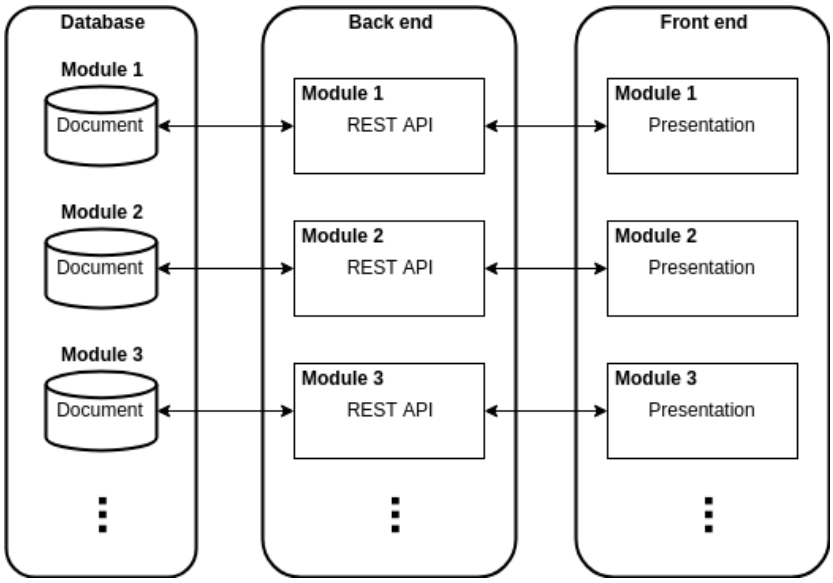

Figure 10: The application structure.

functionality. It should be noted that complex image processing applications will not fit easily into the dashboard, due to their computational complexity. Web applications are limited in this regard. Transferring the computational burden to the server side also has the drawback of latency, which the user can experience as slow.

A typical web application consists of two parts: the back end and the front end. In software engineering principles, these refer to the separation of the data access layer and the presentation layer of the web application. For example, the back end fetches data requested by the front end. In turn, the front end presents the retrieved data towards the user. Due to the modular design of the dashboard, each module needs to have its own back end (data retrieval and database models) and front end functionality as summarized in figure 10. As seen in the figure, there is no interaction between the modules. Furthermore, the front end has a general presentation in which the modules reside. The next sections describe the back end and front end in detail. Also, before the implementation began, I only had basic knowledge of HTML, CSS, and JavaScript. As a result, the learning curve was personally quite steep.

4.2 Back end

The back end consists of a REST API which handles the requests sent by the front end. Each request that the back end supports performs an operation on the database. Examples of such requests are fetching, posting, and deleting data. In this section we describe the REST API, its structure, and the operations it

allows. Hereafter, we mention how the database program was chosen, its models and their relations.

4.2.1 REST API

Node.js was used to create the back end server. This server environment was chosen due its large user base and available libraries. The REST API itself was created with the Express framework and tested with Postman. The documentation of the REST API present in the prototype can be found in appendix A.2. We now describe the general structure of the API and the operations it supports.

General structure The API structure starts at the user. Each user has a unique identifier which is passed on to the subroutes. To get the information of the user, we send the following request:

```
GET /user/123456
```

If there is a user present with the id '123456', the API sends a response containing the user information. Modules that want to use data from a certain user, need to prepend their request with `/user/id_value`, as this is the main route. For example, both a heart rate and a blood pressure module want to retrieve their respective values of a certain user whose id is '1a2b3c'. The modules each send the following request:

```
GET /user/1a2b3c/heart
GET /user/1a2b3c/bp
```

Each module is responsible for its own subroutes and operations it supports. As an example, the threshold values can be retrieved differently for each module, depending on the developer's choice:

```
GET /user/1a2b3c/heart/thresholds
GET /user/1a2b3c/bp/misc_values/thresholds
```

The subroutes are simple to integrate thanks to the Express framework. All the requests a subroute handles are stored in its own file. The main route imports this file so it can forward the subroute towards that module. Suppose we have a `route/heart.js` file that we want to integrate in the main route `user.js`. This is done as follows:

```
const heartRoutes = require('.routes/heart'); //import routes
router.use('/:userId/heart', heartRoutes); //forward routes
```

Only two lines are needed to import all the necessary requests from a module. The main route, and all subroutes of all modules present in the prototype are documented in appendix A.2.

Operations The modules can create new, retrieve, delete, and update data entries. Respectively, these are the following request types: POST, GET, DELETE, and PATCH. In the prototype however, data deletion is not possible because it concerns valuable medical data. For threshold values, deletion is also not possible as every patient is required to have these. Creation of the thresholds for a patient is only possible there is currently no entry present for

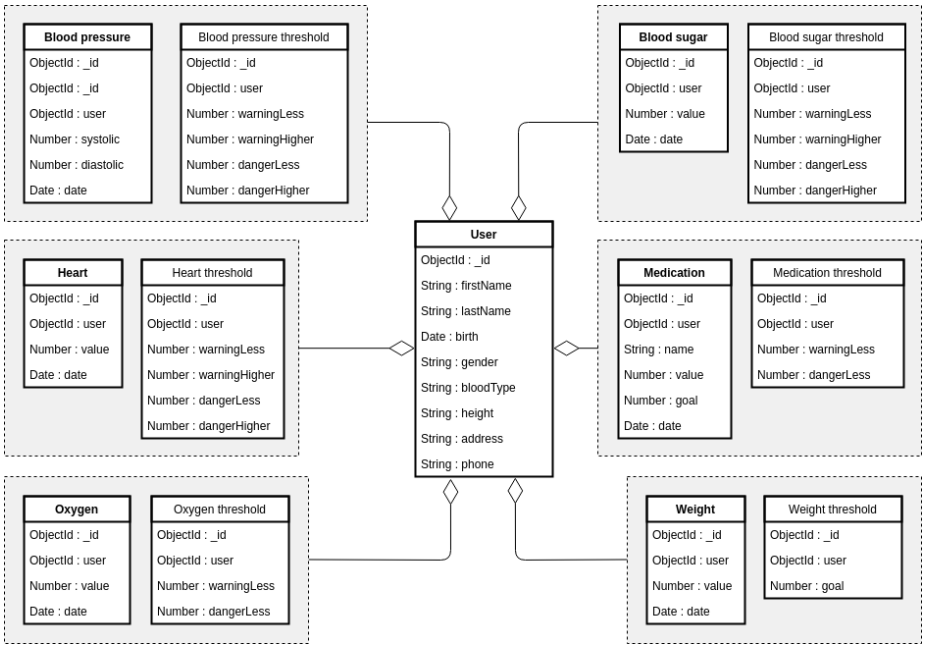

Figure 11: The database structure.

that patient. Again, appendix A.2 documents all implemented operations for all modules present in the prototype.

4.2.2 Database

At the very beginning, SQL was used in the back end, but it was quickly switched out for MongoDB. MongoDB uses JSON-like documents to store data. Groups of these documents are stored in collections. A feature of MongoDB is that the documents inside a collection do not need the same data fields. Suppose we update our patient information page to also show their phone number. In SQL, tables would need to be redefined and the existing rows updated to have a default value for this attribute. In MongoDB this is not necessary as patients with this new attribute can be added without a problem to the existing collection. These flexible documents facilitate integration as no schema rules need to be defined. This allows developers to create their own document schemas and pair them to existing users by solely referring to their id.

The prototype runs a local MongoDB database server where all dashboard data is stored. Each module is responsible for its own documents. As such, every module has a value collection, which contains all values of all users, and a threshold collection, where one threshold entry exists for each user in the dashboard system. No module accesses data from another module. The database is managed via the Mongoose modeling library for Node.js. By defining models,

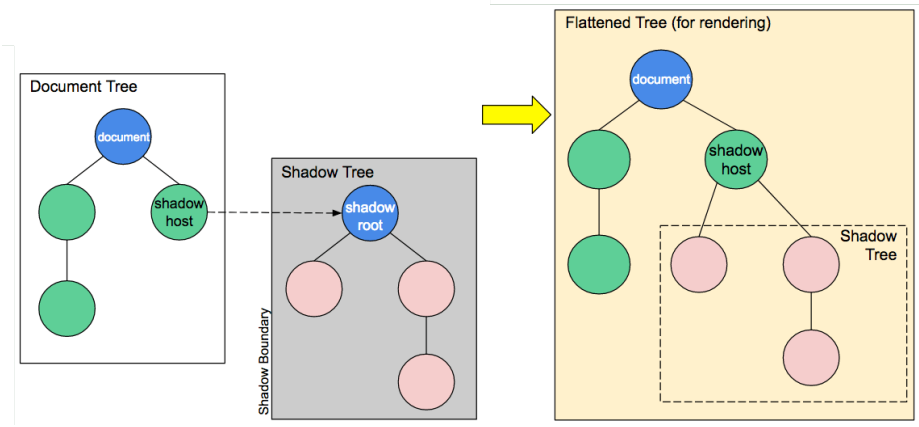

Figure 12: The shadow DOM (src: `https://developer.mozilla.org/en-US/docs/Web/Web_Components/Using_shadow_DOM`).

Mongoose helps us by managing relationships between data. Also, it translates source code objects to their representation in the database.

The models present in this prototype are straightforward, as shown in figure 11. It is possible that one blood pressure entry can have an additional field, such as whether it was seen or not by the clinician. However, the current implementation of each module, ignores extra data fields. If in the future data fields are added, the front end part of the module must adapted in order to present these to the clinician.

4.3 Front end

Ultimately, the front end is what the user sees. From the onset of the implementation process, the search for a library which supports a modular approach began. It was clear that Web Components would be used to create the individual modules, which will be explained in the next section. There are several libraries that support the creation of Web Components. Google's Polymer library was eventually chosen, due to its extensive documentation. The following sections describe the Web Component principles and the structure of the front end. For the latter, the customization aspect and the library used to achieve this are explained.

4.3.1 Web Components

HTML provides us with standard elements such as `h1` and `img`. Web Components allows the developer to create custom elements which can be used in the same manner as the ones HTML provides. Suppose a calendar Web Component was created, a developer could add the element to any web page as follows: `<calendar></calendar>`. The actual definition of the calendar module resides

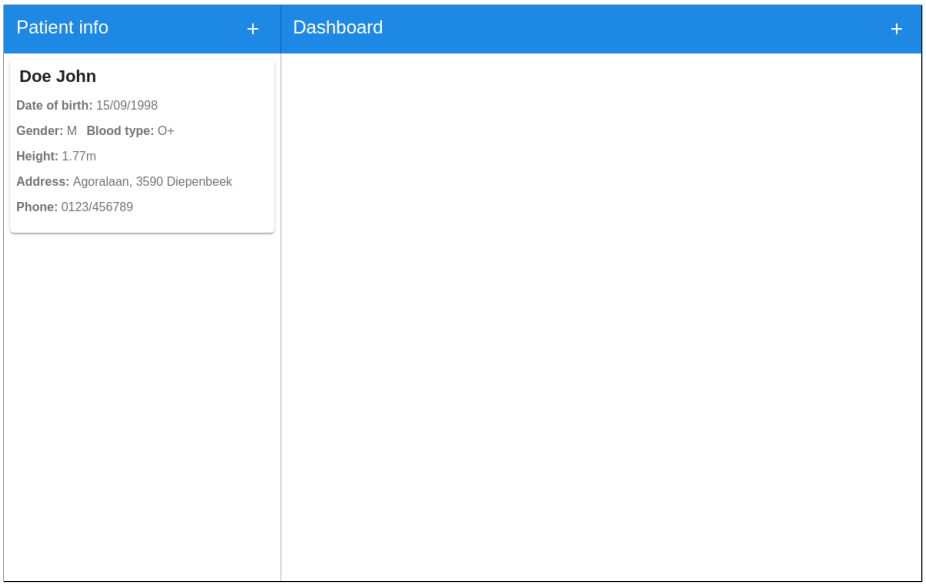

Figure 13: The empty dashboard.

in another file which is imported on the page where it will be used. This approach supports reusability, and above all, easy integration which is a primary goal of our dashboard application. At the heart of Web Components is the shadow DOM.

Shadow DOM We assume the reader is familiar with the DOM concept. The shadow DOM can be seen as a scoped subtree inside a custom element. The root of this subtree is called the shadow root. The structure, styling, and behavior of the children within the shadow DOM do not affect any elements outside of it. Therefore, the appearance and functionality of the custom element is encapsulated and hidden at the document level. Only events fired by elements in the shadow DOM can be pickup outside the shadow DOM boundary. Figure 12 visualizes this concept. This idea of encapsulation is important for our approach towards the dashboard, as each module is responsible for its own appearance and functionality. Developers of custom modules should not need to worry about what happens outside of the shadow DOM.

4.3.2 Structure

The dashboard features two panels. On the left, only small modules can be pinned, while both the small and large modules can be placed on the right panel. The left panel is thin and is automatically hidden in case of small screen estate. Should too many modules be present on the right panel, a scroll bar appears. The left panel will stay in place if scrolling happens in the right panel

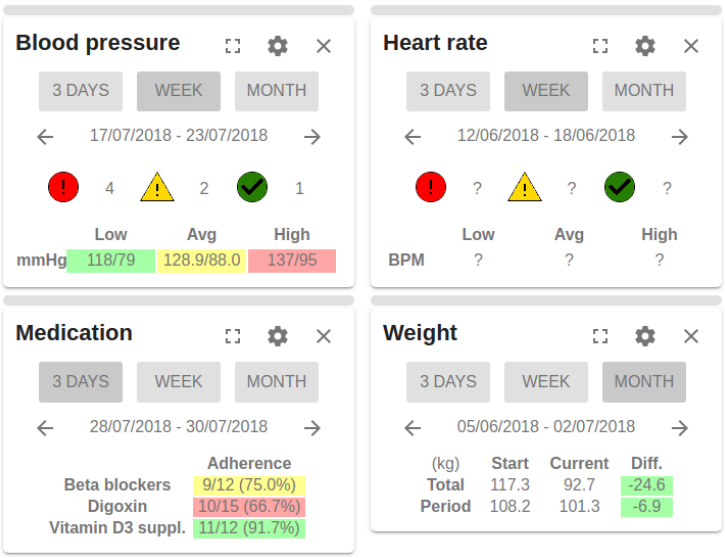

Figure 14: The small modules: blood pressure, heart rate, medication, and weight.

to show the information of the small modules that reside there. Figure 13 shows the empty dashboard.

The right panel allows the ordering of the modules via drag-and-dropping them into place. Several libraries were tried out to achieve this functionality, but many of them did not support dragging Web Components. As a result, the front end was rebuilt several times. Eventually, Packery.js was used. This library allows free placement of the modules and tries to arrange them to maximize space efficiency. In case of resizing, the modules are rearranged automatically. Modules are added via a dialog which can be accessed by pressing the plus-sign button in the top-right corner. The same applies for the smaller panel on the left side. However, the modules in this panel can only be realigned vertically.

4.4 Result

Throughout this section multiple screenshots show the final prototype in action. Extra screenshots can be found in appendix A.3, as some modules are very similar in design and functionality. The threshold settings menu is found at the start of that appendix. Changes visible in a screenshot compared its low fidelity counterpart, are explained in the paragraph where it is referred to. First, the small and large modules are shown. Hereafter, multiple complete dashboards are presented to show the different possibilities in terms of layout.

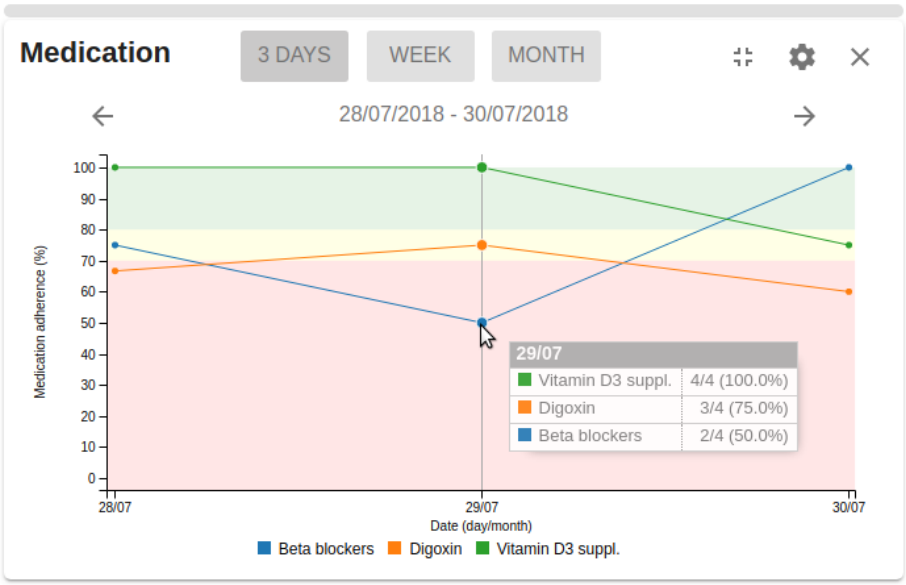

Figure 15: The large medication module.

4.4.1 Small modules

Figure 14 shows four small modules, with each showing data of different time periods. For easy arrangement, all small modules have the same width. Note that there is no heart rate data for the time period selected in that module. The only changes present in the small modules are also applied to the large modules. One of these changes was the addition of a small handle bar at the top of each module. This allows safe dragging and dropping. Previously, the entire module was draggable, which resulted in unexpected button behavior, but could also lead to accidental dragging.

The second and last change, was the replacement of the menu button in the top-right corner. This menu was replaced by buttons for each action it would contain: resize, update thresholds, and delete. As a result, a click is saved. The unit of measurement, which was originally in the title, had to be moved to the tables below to make room for the buttons.

4.4.2 Large modules

The larger modules saw additional changes on top of the two mentioned in the previous section. When the large modules were completed, the right-hand side had a lot of whitespace. To remedy this, the statistics were placed beneath the chart. This had the added bonus that the modules became narrower. All large modules have a fixed width of 716 pixels, which is the minimum screen resolution width without needing to scroll horizontally.

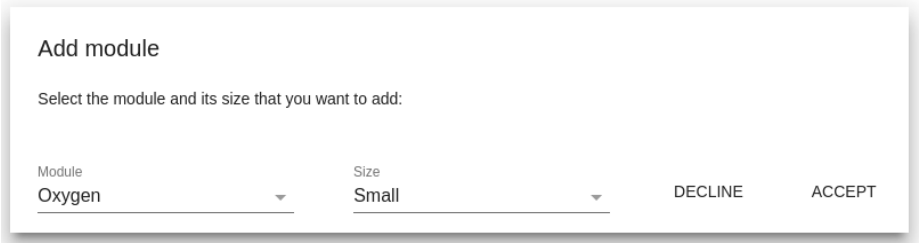

Figure 16: The dialog to add a module to the dashboard.

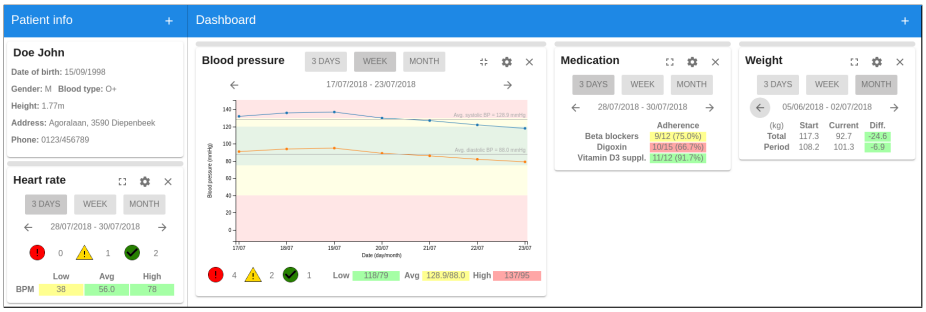

Figure 17: A very wide dashboard.

The charts were generated using the C3.js chart library. Not mentioned in the design, most charts feature a line showing the average of the values for the selected time period. Indicating the lowest and highest values is difficult to do in C3 and was left out of the prototype. Hovering over the plotted data points shows the exact values, as shown in figure 15 for medication. Labelling the axes and indicating the measurement units was overlooked and added later for all charts. All charts are simple to understand and no "chart junk" is present.

4.4.3 Dashboard

As mentioned in section 4.3.2, different screen sizes are supported, and the left panel is hidden if the available screen space is too narrow. This section shows several examples of dashboards on smaller and larger screens. The empty dashboard was already displayed in figure 13. The plus-signs on top open the menu seen in figure 16. Only small modules can be added to the left panel. Figure 17 shows a very wide dashboard, which was automatically resized on a smaller screen to the one shown in figure 18. If the screen is even smaller, the left panel is hidden as seen in figure 19. Pressing the button in the top left corner shows the left panel on top the other modules (figure 20). Extra dashboard layouts are found in appendix A.3.

Figure 18: A narrower version of the dashboard shown in figure 17.

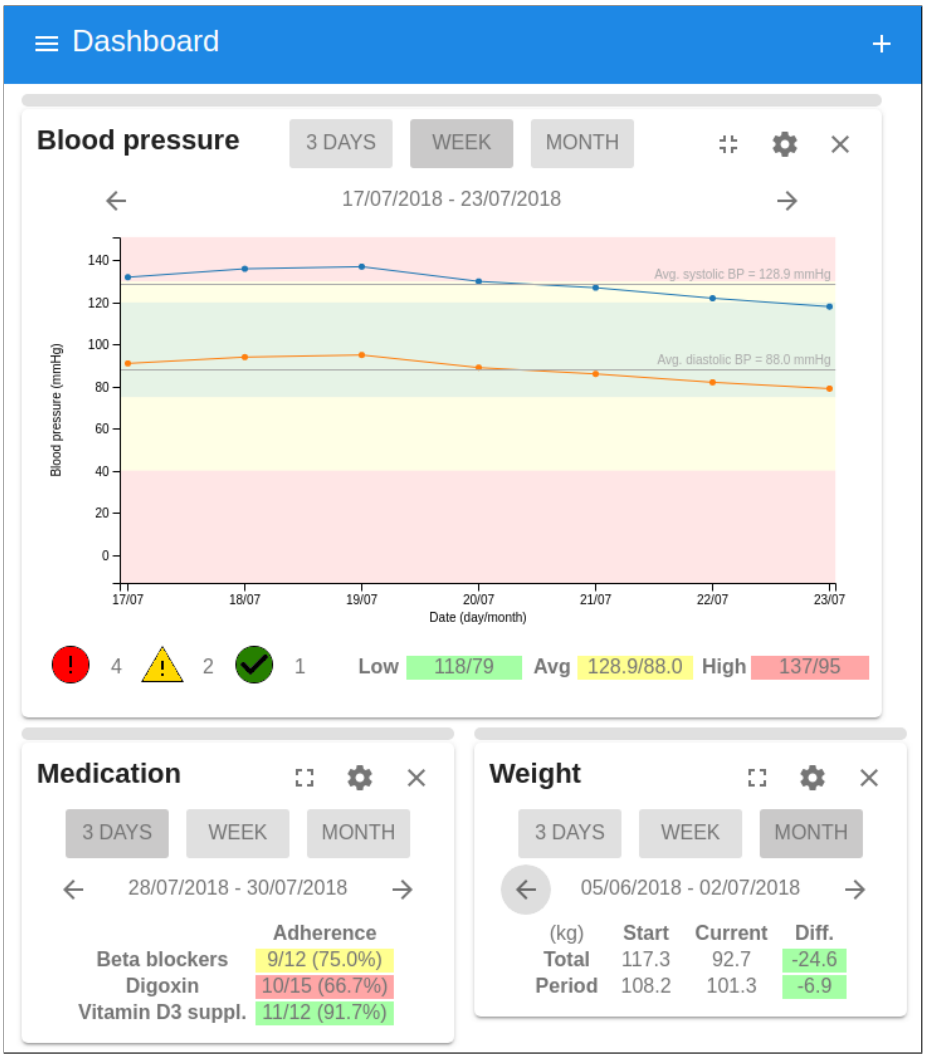

Figure 19: A narrow dashboard with the left panel hidden.

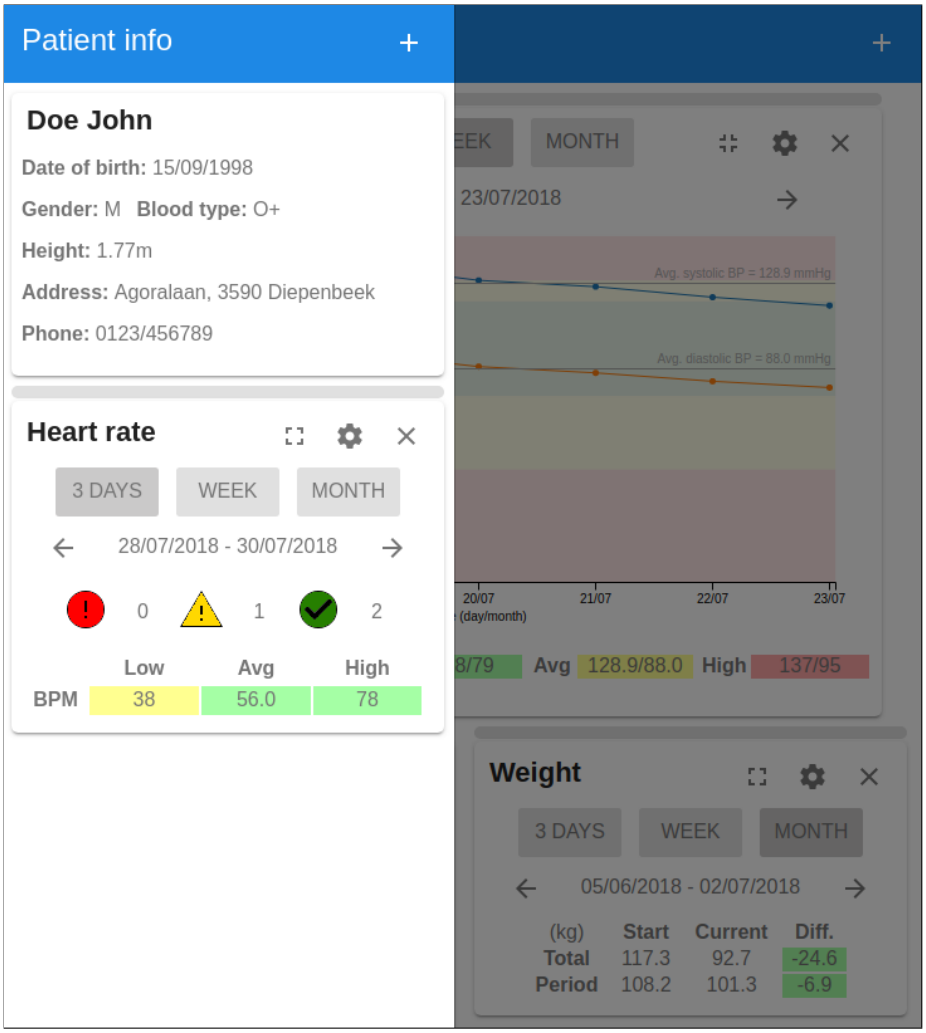

Figure 20: A narrow dashboard with the left panel shown.

5 Usability test

A usability test has been conducted to test the feasibility of the high fidelity prototype. Specifically, we want to know whether our dashboard solution has the potential to solve the issues surrounding the customization and integration of health software. This is merely a proof-of-concept, which demonstrates the solution at a very basic level. Health software is very complex and developing such a modular system for use in this sector is very resource intensive. Therefore, a usability test is absolutely necessary. The results of this test can indicate if further research on this topic is worthwhile.

Customization and integration target two different groups. On one end, clinicians can customize their software environment to increase efficiency. On the other end, developers save time if new pieces of health software can be integrated through a simple process. As a result, two tests need to be designed, one targeting each group. The next sections describe the steps of the test procedure, the test setup, and the data gathering techniques for each topic.

5.1 Customization

To test the customization of the dashboard, qualitative data is gathered via interviews and observation. The actual functionality and appearance of the modules are not evaluated, only the customization aspect.

5.2 Integration

6 Discussion

TODO

7 Conclusion

todo

A Appendix

A.1 Low fidelity prototypes

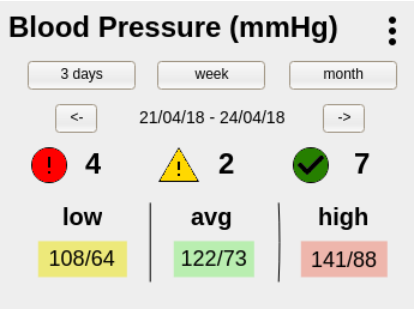

Figure 21: The small blood pressure module.

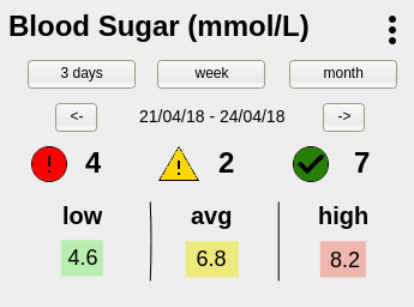

Figure 22: The small blood sugar module.

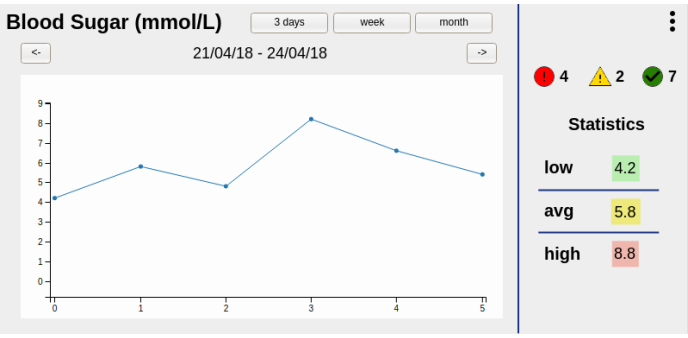

Figure 23: The large blood sugar module.

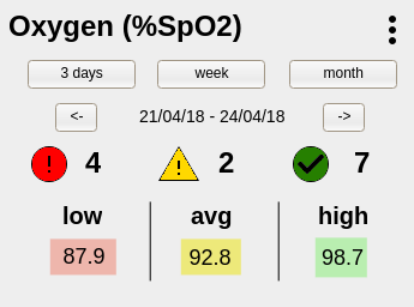

Figure 24: The small oxygen module.

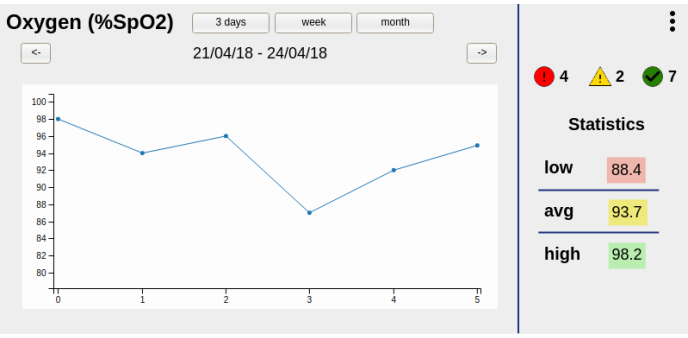

Figure 25: The large oxygen module.

A.2 REST API documentation

A.2.1 User

The user module was created to show basic information and does not offer much functionality.

Create user Creates a new user in the database.

- **URL**: /user
- **Method**: `POST`
- **Data Params**:

```
{
    "firstName": "John",
    "lastName": "Doe",
    "birth": "1998-09-15T15:53:00",
    "gender": "M",
    "bloodType": "O+",
    "height": "1.77m",
    "address": "Agoralaan, 3590 Diepenbeek",
    "phone": "0123/456789"
}
```

Get user Get the information of a user according to the supplied id.

- **URL**: /user/:id
- **Method**: `GET`
- **URL Params**: `id=[string]`
- **Response**:

```
{
    "_id": "5b5c65e3ad30264506380dd1",
    "firstName": "John",
    "lastName": "Doe",
    "birth": "2008-09-15T15:53:00",
    "gender": "M","bloodType": "O+",
    "height": "1.77m",
    "address": "Agoralaan, 3590 Diepenbeek",
    "phone": "0123/456789"
}
```

A.2.2 Blood pressure

Get blood pressure values for large module Get blood pressure values within given period.

- **URL**: /bp/:start&:end
- **Method**: `GET`
- **URL Params**: `start=[integer], end=[integer]`
- **Response**:

```
{
    "thresholds": Object,
    "values": Array,
    "avgLine": Array // contains info to draw
}                    // average lines on chart
```

Get blood pressure statistics for small module Get blood pressure statistics within given period.

- **URL**: /bp/small/:start&:end
- **Method**: `GET`
- **URL Params**: `start=[integer], end=[integer]`
- **Response**:

```
{
    "low": "110/68",
    "high": "131/88",
    "avg": "119.7/78.0",
    "dangerVals": 1,
    "warningVals": 1,
    "okVals": 1,
    "lowCol": "yellow",
    "avgCol": "green",
    "highCol": "red",
    "thresholds": Object
}
```

Create blood pressure entry Creates a new blood pressure entry in the database.

- **URL**: /bp
- **Method**: `POST`
- **Data Params**:

```
{
    "systolic": "118",
    "diastolic": "78",
    "date": "2018-07-30T16:39:12Z"
}
```

Create thresholds Creates default thresholds for user.

- **URL**: /bp/threshold
- **Method**: `POST`

Update thresholds Updates thresholds for user.

- **URL**: /bp/threshold
- **Method**: `PATCH`
- **Data Params**:

```
{
    "warningLess": "40",
    "warningHigher": "41",
    "dangerLess": "42",
    "dangerHigher": "43"
}
```

Get threshold values Get threshold values for user.

- **URL**: /bp/threshold
- **Method**: `GET`
- **Response**:

```
{
    "_id": "5b684666c0e0e", // id from main route
    "warningLess": 75,
    "warningHigher": 120,
    "dangerLess": 40,
    "dangerHigher": 130
}
```

A.2.3 Blood sugar

Get blood sugar values for large module Get blood sugar values within given period.

- **URL**: /bs/:start&:end
- **Method**: `GET`
- **URL Params**: `start=[integer], end=[integer]`
- **Response**:

```
{
    "thresholds": Object,
    "values": Array,
    "avgLine": Array // contains info to draw
}                    // average lines on chart
```

Get blood sugar statistics for small module Get blood sugar statistics within given period.

- **URL**: /bs/small/:start&:end
- **Method**: `GET`
- **URL Params**: `start=[integer], end=[integer]`
- **Response**:

```
{
    "low": "3.8",
    "high": "6.7",
    "avg": "5.2",
    "dangerVals": 0,
    "warningVals": 2,
    "okVals": 1,
    "lowCol": "yellow",
    "avgCol": "green",
```

```
    "highCol": "yellow",
    "thresholds": Object
}
```

Create blood sugar entry Creates a new blood sugar entry in the database.

- **URL**: /bs
- **Method**: `POST`
- **Data Params**:

```
{
    "value": "6.7",
    "date": "2018-07-30T16:39:12Z"
}
```

Create thresholds Creates default thresholds for user.

- **URL**: /bs/threshold
- **Method**: `POST`

Update thresholds Updates thresholds for user.

- **URL**: /bs/threshold
- **Method**: `PATCH`
- **Data Params**:

```
{
    "warningLess": "4",
    "warningHigher": "6",
    "dangerLess": "3",
    "dangerHigher": "8"
}
```

Get threshold values Get threshold values for user.

- **URL**: /bs/threshold
- **Method**: `GET`
- **Response**:

```
{
    "_id": "5b684666c0e0e", // id from main route
    "warningLess": 4,
    "warningHigher": 6,
    "dangerLess": 3,
    "dangerHigher": 8
}
```

A.2.4 Heart rate

Get heart rate values for large module Get heart rate values within given period.

- **URL**: /heart/:start&:end
- **Method**: `GET`
- **URL Params**: `start=[integer], end=[integer]`
- **Response**:

```
{
    "thresholds": Object,
    "values": Array,
    "avgLine": Array // contains info to draw
}                    // average lines on chart
```

Get heart rate statistics for small module Get heart rate statistics within given period.

- **URL**: /heart/small/:start&:end
- **Method**: `GET`
- **URL Params**: `start=[integer], end=[integer]`
- **Response**:

```
{
    "low": "38",
    "high": "78",
    "avg": "56.0",
    "dangerVals": 0,
    "warningVals": 1,
    "okVals": 2,
    "lowCol": "yellow",
    "avgCol": "green",
```

```
    "highCol": "green",
    "thresholds": Object
}
```

Create heart rate entry Creates a new heart rate entry in the database.

- **URL**: /heart
- **Method**: `POST`
- **Data Params**:

```
{
    "value": "62",
    "date": "2018-07-30T16:39:12Z"
}
```

Create thresholds Creates default thresholds for user.

- **URL**: /heart/threshold
- **Method**: `POST`

Update thresholds Updates thresholds for user.

- **URL**: /heart/threshold
- **Method**: `PATCH`
- **Data Params**:

```
{
    "warningLess": "40",
    "warningHigher": "80",
    "dangerLess": "30",
    "dangerHigher": "95"
}
```

Get threshold values Get threshold values for user.

- **URL**: /heart/threshold
- **Method**: `GET`
- **Response**:

```
{
    "_id": "5b684666c0e0e", // id from main route
    "warningLess": 40,
    "warningHigher": 80,
    "dangerLess": 30,
    "dangerHigher": 95
}
```

A.2.5 Medication

Get medication adherence for large module Get medication adherence within given period.

- **URL**: /medication/:start&:end
- **Method**: `GET`
- **URL Params**: `start=[integer], end=[integer]`
- **Response**:

```
{
    "thresholds": Object,
    "values": Array,
    "dates": Array, // dates for each value
    "xsDates": Array, // array to pair each medicine
                      // to line on chart
    "indivStrings": Array // contains adherence
}                         // fractures for tooltips
```

Get medication adherence averages for small module Get medication adherence averages within given period.

- **URL**: /medication/small/:start&:end
- **Method**: `GET`
- **URL Params**: `start=[integer], end=[integer]`
- **Response**:

```
{
    "thresholds": Object,
    "values": [
        {
            "name": "Beta blockers",
            "values": 9,
```

```
            "goal": 12,
            "percentage": "75.0",
            "color": "#ffff90"
        }
    ]
}
```

Create medication adherence entry Creates a medication adherence entry in the database.

- **URL**: /medication
- **Method**: `POST`
- **Data Params**:

```
{
    "name": "Beta blockers",
    "value": "3",
    "goal": "4",
    "date": "2018-07-30T16:39:12Z"
}
```

Create thresholds Creates default thresholds for user.

- **URL**: /medication/threshold
- **Method**: `POST`

Update thresholds Updates thresholds for user.

- **URL**: /medication/threshold
- **Method**: `PATCH`
- **Data Params**:

```
{
    "warningLess": "90",
    "dangerLess": "70"
}
```

Get threshold values Get threshold values for user.

- **URL**: /medication/threshold
- **Method**: `GET`
- **Response**:

```
{
    "_id": "5b684666c0e0e", // id from main route
    "warningLess": 90,
    "dangerLess": 70
}
```

A.2.6 Oxygen

Get oxygen values for large module Get oxygen values within given period.

- **URL**: /oxygen/:start&:end
- **Method**: `GET`
- **URL Params**: `start=[integer], end=[integer]`
- **Response**:

```
{
    "thresholds": Object,
    "values": Array,
    "avgLine": Array // contains info to draw
}                    // average lines on chart
```

Get oxygen statistics for small module Get oxygen statistics within given period.

- **URL**: /oxygen/small/:start&:end
- **Method**: `GET`
- **URL Params**: `start=[integer], end=[integer]`
- **Response**:

```
{
    "low": "92.1",
    "high": "98.4",
    "avg": "95.0",
```

```
    "dangerVals": 0,
    "warningVals": 2,
    "okVals": 1,
    "lowCol": "yellow",
    "avgCol": "yellow",
    "highCol": "green",
    "thresholds": Object
}
```

Create oxygen entry Creates an oxygen entry in the database.

- **URL**: /oxygen
- **Method**: `POST`
- **Data Params**:

```
{
    "value": "96.7",
    "date": "2018-07-30T16:39:12Z"
}
```

Create thresholds Creates default thresholds for user.

- **URL**: /oxygen/threshold
- **Method**: `POST`

Update thresholds Updates thresholds for user.

- **URL**: /oxygen/threshold
- **Method**: `PATCH`
- **Data Params**:

```
{
    "warningLess": "94",
    "dangerLess": "90"
}
```

Get threshold values Get threshold values for user.

- **URL**: /oxygen/threshold
- **Method**: `GET`
- **Response**:

```
{
    "_id": "5b684666c0e0e", // id from main route
    "warningLess": 94,
    "dangerLess": 90
}
```

A.2.7 Weight

Get weight values for large module Get weight values within given period.

- **URL**: /weight/:start&:end
- **Method**: `GET`
- **URL Params**: `start=[integer], end=[integer]`
- **Response**:

```
{
    "goal": 86,
    "values": Array
}
```

Get weight statistics for small module Get weight statistics within given period.

- **URL**: /weight/small/:start&:end
- **Method**: `GET`
- **URL Params**: `start=[integer], end=[integer]`
- **Response**:

```
{
    "startWeight": 102,
    "curWeight": 94,
    "difference": "-8.0",
    "startPeriod": 102,
    "endPeriod": 94,
```

```
    "periodDifference": "-8.0",
    "totalCol": "green",
    "periodCol": "green",
    "goal": 86
}
```

Create weight entry Creates a weight entry in the database.

- **URL**: /weight
- **Method**: `POST`
- **Data Params**:

```
{
    "weight": "84.3",
    "date": "2018-07-30T16:39:12Z"
}
```

Create thresholds Creates default thresholds for user.

- **URL**: /weight/threshold
- **Method**: `POST`

Update thresholds Updates thresholds for user.

- **URL**: /weight/threshold
- **Method**: `PATCH`
- **Data Params**:

```
{
    "goal": "80"
}
```

Get threshold values Get threshold values for user.

- **URL**: /weight/threshold
- **Method**: `GET`
- **Response**:

```
{
    "_id": "5b684666c0e0e", // id from main route
    "goal": 80
}
```

A.3 High fidelity prototype screenshots

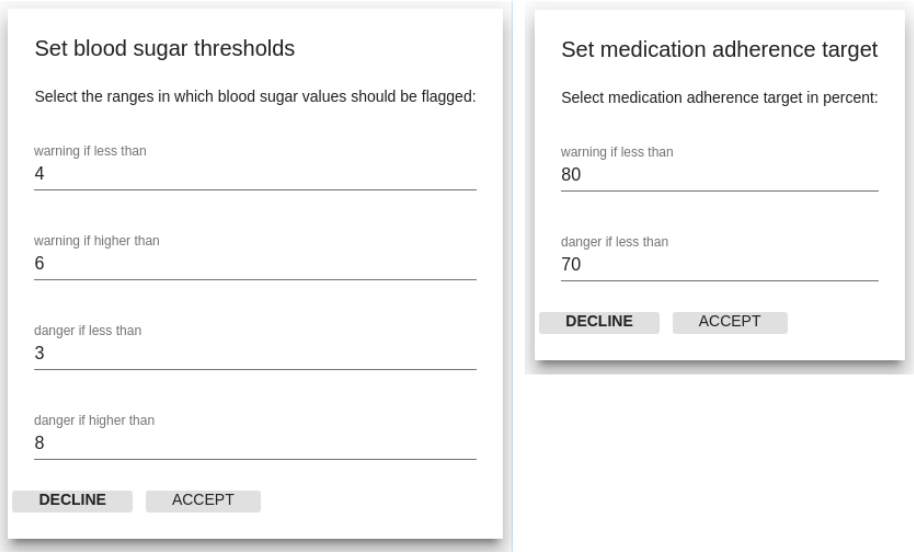

Figure 26: Examples of the blood sugar and medication thresholds dialog.

A.3.1 Small modules

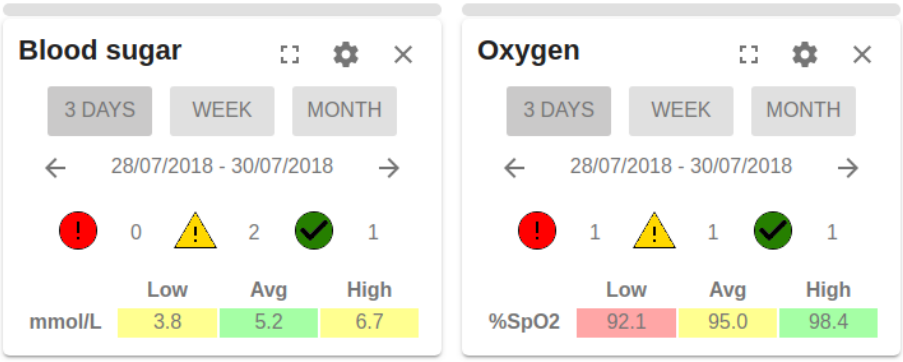

Figure 27: The small blood sugar and oxygen modules.

A.3.2 Large modules

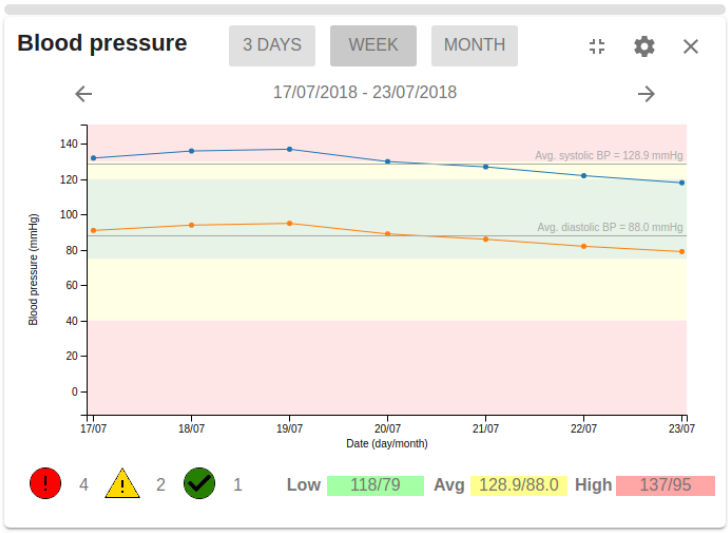

Figure 28: The large blood pressure module.

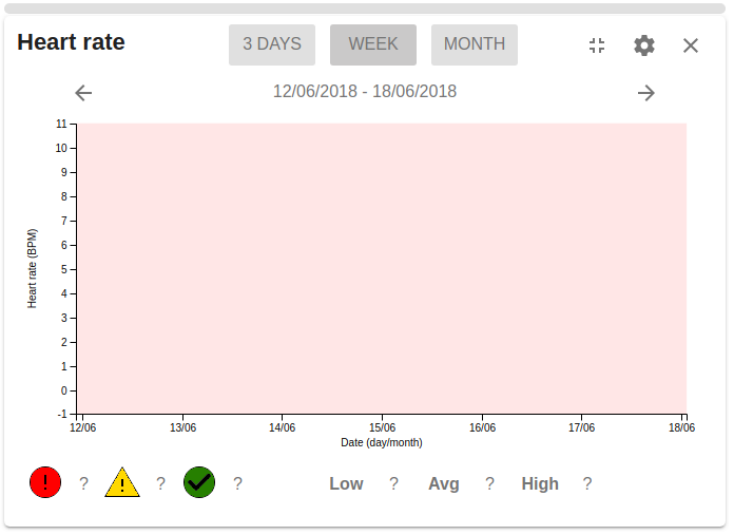

Figure 29: The large heart module, with no available data for the time period.

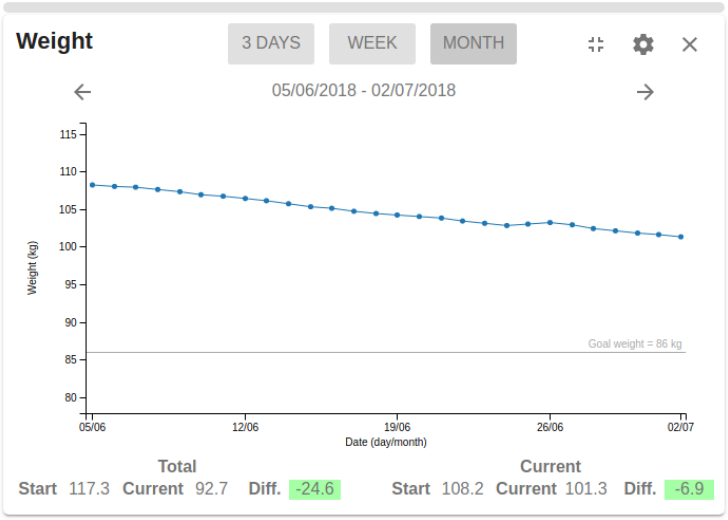

Figure 30: The large weight module.

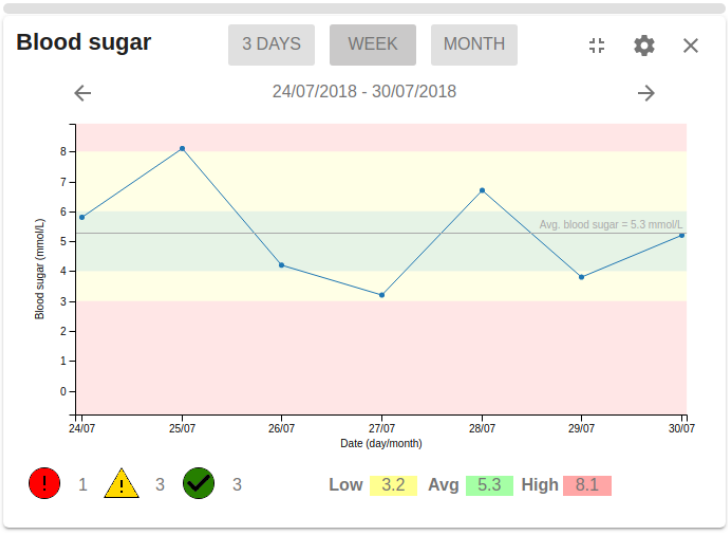

Figure 31: The large blood sugar module.

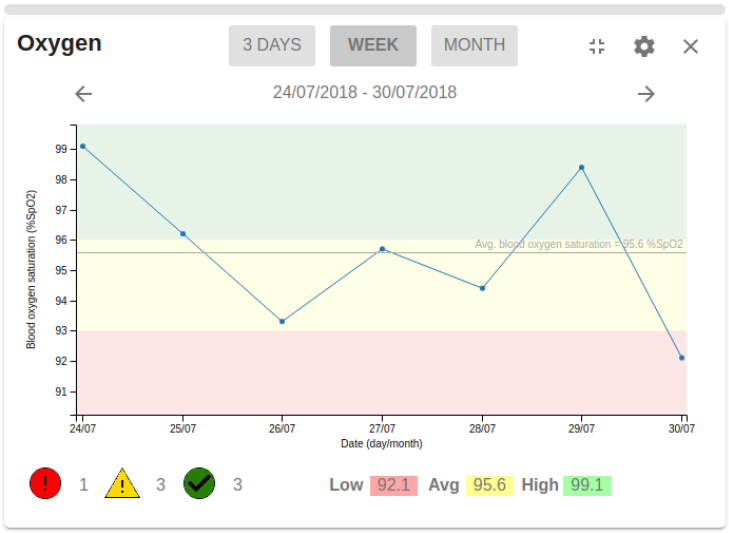

Figure 32: The large oxygen module.

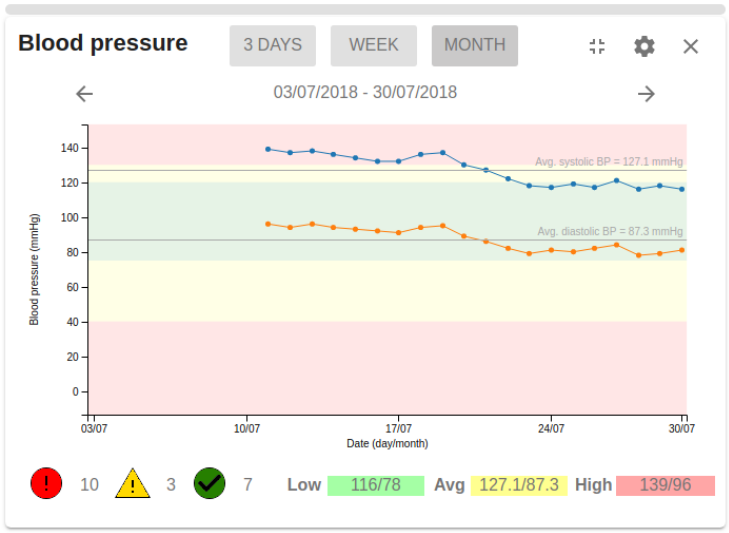

Figure 33: The large blood pressure module.

A.3.3 Dashboard examples

Figure 34: A very tall dashboard.

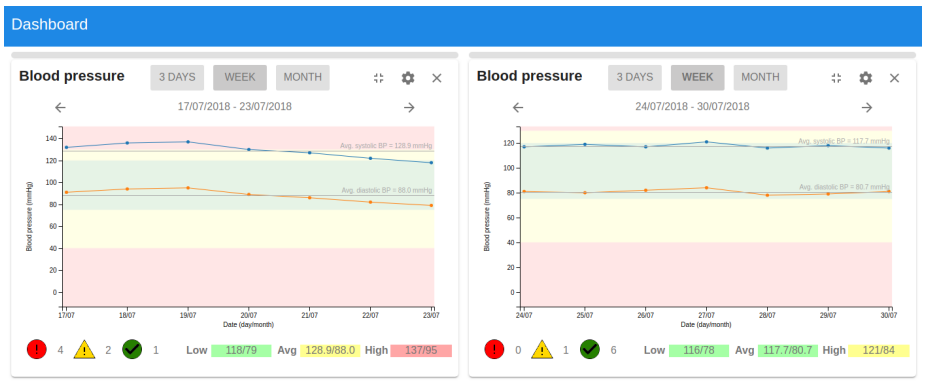

Figure 35: Comparing weeks by adding the same module.

A.4 Usability test documents

References

[1] Edward H Shortliffe and James J Cimino. *Biomedical informatics*. Springer, 2006.

[2] Beatriz Sainz de Abajo and Agustín Llamas Ballestero. Overview of the most important open source software: analysis of the benefits of openmrs, openemr, and vista. In *Telemedicine and E-Health Services, Policies, and Applications: Advancements and Developments*, pages 315–346. IGI Global, 2012.

[3] Kristiina Häyrinen, Kaija Saranto, and Pirkko Nykänen. Definition, structure, content, use and impacts of electronic health records: a review of the research literature. *International journal of medical informatics*, 77(5):291–304, 2008.

[4] Dwight C Evans, W Paul Nichol, and Jonathan B Perlin. Effect of the implementation of an enterprise-wide electronic health record on productivity in the veterans health administration. *Health Economics, Policy and Law*, 1(2):163–169, 2006.

[5] T Payne, J Fellner, C Dugowson, D Liebovitz, and G Fletcher. Use of more than one electronic medical record system within a single health care organization. *Applied clinical informatics*, 3(04):462–474, 2012.

[6] World Health Organization et al. Noncommunicable diseases: progress monitor 2017. 2017.

[7] Shanthi Mendis, Pekka Puska, Bo Norrving, et al. *Global atlas on cardiovascular disease prevention and control.* World Health Organization, 2011.

[8] National Center for Health Statistics (US et al. Health, united states, 2015: with special feature on racial and ethnic health disparities. 2016.

[9] Jessie Gerteis, David Izrael, Deborah Deitz, Lisa LeRoy, Richard Ricciardi, Therese Miller, and Jayasree Basu. Multiple chronic conditions chartbook. *Rockville, MD: Agency for Healthcare Research and Quality*, 2014.

[10] National Cancer Institute. What is cancer? 2018.

[11] Claus F Vogelmeier, Gerard J Criner, Fernando J Martinez, Antonio Anzueto, Peter J Barnes, Jean Bourbeau, Bartolome R Celli, Rongchang Chen, Marc Decramer, Leonardo M Fabbri, et al. Global strategy for the diagnosis, management, and prevention of chronic obstructive lung disease 2017 report. gold executive summary. *American journal of respiratory and critical care medicine*, 195(5):557–582, 2017.

[12] Walter C Willett, Jeffrey P Koplan, Rachel Nugent, Courtenay Dusenbury, Pekka Puska, and Thomas A Gaziano. Prevention of chronic disease by means of diet and lifestyle changes. 2006.

[13] US Department of Health, Human Services, et al. The health consequences of involuntary exposure to tobacco smoke: a report of the surgeon general. us department of health and human services, centers for disease control and prevention. *Coordinating Center for Health Promotion, National Center for Chronic Disease Prevention and Health Promotion, Office on Smoking and Health*, 2006.

[14] Henry C McGill, C Alex McMahan, and Samuel S Gidding. Preventing heart disease in the 21st century: implications of the pathobiological determinants of atherosclerosis in youth (pday) study. *Circulation*, 117(9):1216–1227, 2008.

[15] Thomas Bodenheimer, Edward H Wagner, and Kevin Grumbach. Improving primary care for patients with chronic illness. *Jama*, 288(14):1775–1779, 2002.

[16] Thomas Bodenheimer, Edward H Wagner, and Kevin Grumbach. Improving primary care for patients with chronic illness: the chronic care model, part 2. *Jama*, 288(15):1909–1914, 2002.

[17] Rebecca Reynolds, Sarah Dennis, Iqbal Hasan, Jan Slewa, Winnie Chen, David Tian, Sangeetha Bobba, and Nicholas Zwar. A systematic review of chronic disease management interventions in primary care. *BMC family practice*, 19(1):11, 2018.

[18] Andrea S Wallace, John R Carlson, Robb M Malone, James Joyner, and Darren A DeWalt. The influence of literacy on patient-reported experiences of diabetes self-management support. *Nursing research*, 59(5):356, 2010.

[19] Guy Paré, Mirou Jaana, and Claude Sicotte. Systematic review of home telemonitoring for chronic diseases: the evidence base. *Journal of the American Medical Informatics Association*, 14(3):269–277, 2007.

[20] Stephane Meystre. The current state of telemonitoring: a comment on the literature. *Telemedicine Journal & e-Health*, 11(1):63–69, 2005.

[21] Christopher Tompkins and John Orwat. A randomized trial of telemonitoring heart failure patients. *Journal of Healthcare Management*, 55(5), 2010.

[22] Sally Inglis. Structured telephone support or telemonitoring programmes for patients with chronic heart failure. *Journal of Evidence-Based Medicine*, 3(4):228–228, 2010.

[23] Sarwat I Chaudhry, Jennifer A Mattera, Jeptha P Curtis, John A Spertus, Jeph Herrin, Zhenqiu Lin, Christopher O Phillips, Beth V Hodshon, Lawton S Cooper, and Harlan M Krumholz. Telemonitoring in patients with heart failure. *New England Journal of Medicine*, 363(24):2301–2309, 2010.

[24] Robert SH Istepanian, Karima Zitouni, Diane Harry, Niva Moutosammy, Ala Sungoor, Bee Tang, and Kenneth A Earle. Evaluation of a mobile phone telemonitoring system for glycaemic control in patients with diabetes. 2009.

[25] Lewis Landsberg and Mark Molitch. Diabetes and hypertension: pathogenesis, prevention and treatment. *Clinical and Experimental Hypertension*, 26(7-8):621–628, 2004.

[26] Alexander G Logan, M Jane Irvine, Warren J McIsaac, Andras Tisler, Peter G Rossos, Anthony Easty, Denice S Feig, and Joseph A Cafazzo. Effect of home blood pressure telemonitoring with self-care support on uncontrolled systolic hypertension in diabetics. *Hypertension*, pages HYPERTENSIONAHA–111, 2012.

[27] Claus Luley, Alexandra Blaik, Kirsten Reschke, Silke Klose, and Sabine Westphal. Weight loss in obese patients with type 2 diabetes: Effects of telemonitoring plus a diet combination–the active body control (abc) program. *Diabetes research and clinical practice*, 91(3):286–292, 2011.

[28] Roslyn A Stone, R Harsha Rao, Mary Ann Sevick, Chunrong Cheng, Linda J Hough, David S Macpherson, Carol M Franko, Rebecca A Anglin, D Scott Obrosky, and Frederick R DeRubertis. Active care management supported by home telemonitoring in veterans with type 2 diabetes:(the diatel randomized controlled trial). *Diabetes care*, 2009.

[29] Carmela Maiolo, Ehab I Mohamed, Cesare M Fiorani, and Antonino De Lorenzo. Home telemonitoring for patients with severe respiratory illness: the italian experience. *Journal of Telemedicine and Telecare*, 9(2):67–71, 2003.

[30] Keir E Lewis, Joseph A Annandale, Daniel L Warm, Claire Hurlin, Michael J Lewis, and Leo Lewis. Home telemonitoring and quality of life in stable, optimised chronic obstructive pulmonary disease. *Journal of telemedicine and telecare*, 16(5):253–259, 2010.

[31] Stephen Few. Information dashboard design. 2006.

[32] Stephen Few. Intelligent dashboard design. *Information Management*, 15(9):12, 2005.

[33] Richard Brath and Michael Peters. Dashboard design: Why design is important. *DM Direct*, pages 1011285–1, 2004.

[34] Marci Meingast, Tanya Roosta, and Shankar Sastry. Security and privacy issues with health care information technology. In *Engineering in Medicine and Biology Society, 2006. EMBS'06. 28th Annual International Conference of the IEEE*, pages 5453–5458. IEEE, 2006.

[35] Peter Mildenberger, Marco Eichelberg, and Eric Martin. Introduction to the dicom standard. *European radiology*, 12(4):920–927, 2002.

[36] Mieke Haesen, Karin Coninx, Jan Van den Bergh, and Kris Luyten. Muicser: A process framework for multi-disciplinary user-centred software engineering processes. In *Engineering Interactive Systems*, pages 150–165. Springer, 2008.

[37] Sabine Madsen and Lene Nielsen. Exploring persona-scenarios-using storytelling to create design ideas. In *Human work interaction design: Usability in social, cultural and organizational contexts*, pages 57–66. Springer, 2010.