



Degree project in Technology
First cycle, 15 credits

Supporting CKD Patients on Home Hemodialysis with Digital Information and Communication

SANDRA RÖDLUND

MATILDA BYSTRÖM

This project was performed in collaboration with
Baxter medical AB
Karolinska Institutet
Supervisor at Karolinska Institutet: Jimmy Friebe

Supporting CKD Patients on Home Hemodialysis with Digital Information and Communication

Assistera CKD patienter med hemhemodialys genom
digital information och kommunikation

SANDRA RÖDLUND
MATILDA BYSTRÖM

Degree project in medical engineering
First level, 15 credits
Supervisor at KTH: Tobias Nyberg, Mattias Mårtensson
Examiner: Mats Nilsson

KTH Royal Institute of Technology
School of Engineering Sciences in Chemistry, Biotechnology and Health
Hälsovägen 11 C
SE-141 57 Huddinge, Sweden
<http://www.kth.se/cbh>

2022

Abstract

The lack of digital and easily accessible information for home hemodialysis patients leads to a disinclination of using the provided manuals. This problem could potentially be solved with a user-friendly app, where all information and communication with the healthcare providers could take place. Hence, leading to them feeling better prepared to take charge of their own treatment. The purpose of this thesis is to provide a minimum viable product of a mobile application to facilitate treatment for home hemodialysis patients. The app was developed using the programming language SwiftUI. Information was collected from patients with home hemodialysis treatment through interviews conducted in their homes.

The mobile application enables patients to access the manual and failure identification codes through a search bar. It also provides digital checklists, dialysis protocol and symptom follow-up. The healthcare providers can then access the dialysis protocol and symptom follow-up through remote patient monitoring. A chat function allows patients to contact nurses. The mobile application fulfills the clients' requests and with additional work it could be fully operational.

Keywords:

Hemodialysis, home hemodialysis, app development, SwiftUI, mobile application, remote patient monitoring.

Sammanfattning

Bristen på digital och lättåtkomlig information för patienter med hemhemodialys leder till en ovilja att använda manualer och felkoder. En lösning på detta problem skulle kunna vara att utveckla en användarvänlig app som innehåller all nödvändig information, samt ett kommunikationsverktyg mellan patient och vårdpersonal. Syftet med detta examensarbete är att skapa en minsta gångbar produkt i form av en app för att underlätta behandlingen för patienter med hemhemodialys. Appen utvecklades med programmeringsspråket SwiftUI. Information hämtades från patienter genom utförda hemintervjuer.

Den mobila applikationen ger patienter åtkomst till manualen samt felkoderna genom en sökruta. Den innehåller även digitala checklistor, dialysprotokoll och symptomuppföljning. Vårdpersonal kan genom appen få tillgång till patienters dialysprotokoll och symptomuppföljning genom fjärrövervakning. Via chatten kan patienter kontakta sjukvårdspersonal. Applikationen uppfyller uppdragsgivarens önskemål om funktionaliteter. Om fortsatt arbete genomförs kan appen användas för att ge patienterna stöd i sin egenvård.

Nyckelord:

Hemodialys, hemhemodialys, apputveckling, SwiftUI, mobil applikation, fjärrövervakning.

Acknowledgement

We would like to express our gratitude towards our supervisor Jimmy Friebe for his invaluable dedication and support to help us complete this thesis. We would also like to thank our coach Dr. Rita Nohra for her enthusiasm and guidance. A special thanks to Fernando Seoane Martinez and Märit Östling for making this project possible. Lastly, we would like to thank all patients that participated in the interviews, for your positive attitude and for showing us what home hemodialysis treatment looks like.

Contents

1	Introduction	1
1.1	Aim.....	1
1.2	Limitation	2
2	Background	3
2.1	Chronic kidney disease and kidney function	3
2.2	Hemodialysis	3
2.3	Home hemodialysis	4
2.4	Stakeholders.....	4
2.5	Mobile health and remote patient monitoring	4
3	Method.....	5
3.1	Theoretical phase	5
3.2	Development phase.....	5
4	Results	7
4.1	Interviews with patients	7
4.2	Compile the information that will be included in MVP	7
4.3	Establish a development plan for a technical solution.....	7
4.4	Finish implementation of MVP	7
5	Discussion	13
6	Conclusion.....	17
7	References	19
	Appendix 1: Frågor vid hembesök, version 1	
	Appendix 2: Frågor vid hembesök, version 2	
	Appendix 3: Development plan for technical solution	
	Appendix 4: Patient interview, patient 1, man 67 years	
	Appendix 5: Patient interview, patient 2, man 50 years	
	Appendix 6: Patient interview, patient 3, woman 73 years	
	Appendix 7: Programming code	

1 Introduction

Kidney failure is a life-threatening condition if left untreated, it always requires dialysis or a kidney transplant. One treatment method for kidney failure is hemodialysis and it can be done at the hospital or in the homes of the patients [1]. If the treatment takes place at home, it is called home hemodialysis (HHD) and it requires patients to monitor their own treatment [1]. This method enables flexibility in choosing their own treatment schedule, but it also leads to some difficulties [2]. Instead of nurses treating them, the patients are required to deal with all the steps of dialysis treatment [1]. Before they can do HHD, they are obliged to undergo a thorough training to make sure that they can perform dialysis on their own. If they encounter any challenges during treatment at home, they can receive help by reading the manuals regarding the machine or call the nurse on duty. Research shows that in the early months of HHD, patients felt scared when something unfamiliar happened and nearly all had forgotten some aspect of the process [3]. This issue results in fears and mistakes during treatment [3].

According to Jimmy Friebe, medical engineer at Rosenlund's hospital, there are currently 23 patients on HHD at Rosenlund's hospital and every patient must have dialysis at least three times a week. Hence, many opportunities to encounter challenges. When a problem arises, patients find it hard to dialyze while managing the manual with one hand [3]. The impracticality of reading a book in a stressful situation may make it difficult to solve the problem.

Patients on HHD from Rosenlund's hospital receive paper checklists and logs. The checklists are used to check off the important steps they must do before, during and after treatment to make sure they do not forget anything. In the logs, they record, among other things, heart rate, blood pressure and weight. Considering the logs are on paper, monitoring their vital signs and follow-ups by healthcare providers (HCPs) is not performed on a regular basis and needs to be postponed until the upcoming scheduled visit to the clinic. This could sometimes be delayed for several weeks up to a couple of months. The logs can also be lost, or patients can forget to bring them to regular checkups at the hospital. Research suggests one improvement that could be beneficial is remote patient monitoring (RPM) and the ability for clinicians to send messages to the patient [3]. A study from 2010 shows that one barrier for patients to choose HHD treatment is the fear of not being monitored by professionals [4]. If they are monitored by professionals for deviations, a sense of security can be felt by them. It could also possibly lead to better care, as continuous modifications regarding their care can be done.

A possible solution to this problem was suggested by the medical engineers at the dialysis unit at Rosenlund's hospital. The solution included a service accessed through a mobile application where educational resources and a communication tool are provided. This proposition was presented to Baxter Medical AB, and they decided to support the project.

This thesis provides a minimum viable product (MVP) of a mobile application for HHD patients. This leads to information being easily accessible, a communication tool for patients and HCPs and RPM of patients' vital signs.

1.1 Aim

The purpose of this project is to develop an MVP of a mobile application to compile information related to the usage of a HD machine. This MVP includes:

- Instructions of use.
- Failure identification codes.
- Chat function between patients and HCPs.
- A symptom follow-up function where the patient will track their status and HCPs can adjust treatment if deviation in symptoms occurs.

Furthermore, patient interviews are conducted to find out further needs regarding the MVP. To achieve this, the following sub goals are conducted:

Sub goal 1: Conducting patient interviews.

Sub goal 2: Compile the information that will be included in MVP.

Sub goal 3: Establish a development plan for a technical solution.

Sub goal 4: Finish implementation of MVP.

1.2 Limitation

- The technical implementation platform used is iOS. This limits the mobile application to only being available for Apple units.
- The project is an MVP and therefore the focus is on implementing functionalities. All information provided by Baxter is not included in the app, just enough to show the functionalities.
- Several technical functionalities are being discussed but only a few are implemented in the app, the rest is presented at the last meeting for further development of the mobile application.
- The app is available in Swedish. This limits the mobile application to only being available for Swedish speaking.

2 Background

In the following section an underlying background and literature study is presented. This provides the reader with necessary background information regarding HHD and what causes the need for treatment, as well as the desideratum of the mobile application.

2.1 Chronic kidney disease and kidney function

Kidneys are vital organs that filter waste from the blood and help control blood pressure. They also remove extra fluid and regulate certain electrolytes in the blood such as potassium and sodium [5].

Chronic kidney disease (CKD) implies that there is irreversible damage to the kidneys. The main causes of CKD are diabetes and high blood pressure [1]. If the damage is severe, kidney failure may occur and a retention of water and uremic toxins (also known as waste products) are accumulated in the patients' bodies leading to further health complications [5]. A person with CKD may begin feeling ill, with symptoms like high blood pressure, anemia, poor nutritional health, and weak bones [1]. These symptoms are an indication of kidney diseases. There are five different stages of CKD varying from kidneys being impaired but still functioning at stage one to severely damaged and bordering to failure at stage five [5]. Less severe symptoms such as trouble concentrating, swollen feet, poor appetite, and the need to urinate more often may be early indications of kidney diseases [1]. Early treatment may keep the disease from getting worse [1]. Dialysis or a kidney transplantation will be necessary [5].

2.2 Hemodialysis

The hemodialysis process is a treatment to remove excess water and waste products from a patient's blood when their kidneys are not working correctly. Before they can get hemodialysis, they must undergo surgery for a vascular access that allows a more direct connection to their bloodstream [2]. According to the same source, two needles are connected to the access, one is for outflow and one for inflow of blood. One of the preparation steps before treatment is to get the weight of the patient. Since people with kidney diseases can accumulate extra fluids, getting rid of that is of major importance. Every patient has a dry weight which is the weight they should achieve after treatment [2].

When the patient is connected to the dialysis machine (monitor) and the treatment is initiated, blood will start circulating [6]. According to the same source, the blood then passes through a dialyzer that is made up of many thin, hollow fibers. The thin membrane in the filter separates blood from the dialysis fluid [6]. This will lead to waste products and excess water are filtered from the blood [6]. The cleaned blood can then be returned to the body [6]. Every treatment takes about 4 hours [6].

The treatment can cause several symptoms of varying degrees [7]. Some of these include nausea, restless legs, depression, and lack of energy [7]. By following up these symptoms in the form of a questionnaire the HCPs can take action to improve the patient's quality of life [7].

2.3 Home hemodialysis

Hemodialysis can take place at a dialysis unit where the treatment is pre-scheduled and usually done three times a week [8]. However, it can also be done from home where the patient performs their own treatment [8]. HHD is not a good fit for everyone as it requires a lot of responsibility for their own treatment [8]. It also requires several weeks of training to learn how to use the dialysis machine and to self-cannulate [8]. HHD is favorable from a health perspective since it enables patients to perform the treatment more often [8]. HHD also allows more freedom as it gives them the flexibility of choosing their own treatment schedule [8]. This treatment method has proved to give a better quality of life than for patients undergoing treatment at the hospital [9]. According to Jimmy Friebe, to support the patients in HHD, they have access to several checklists to fill in before, during, and after the treatment to make sure they do not miss any important steps in the process. The patients also fill in a log that includes, among other things, a lot-number on filters, blood pressure, heart rate and weight before and after treatment.

2.4 Stakeholders

This thesis was founded by the Rosenlund's dialysis unit as they saw the need for a mobile application to make it easier for HHD patients to receive information and communicate with the unit. Rosenlund's dialysis unit is a facility where patients can practice doing their own treatment and undergo training to be able to do HHD. The dialysis treatment is done with machines manufactured by Baxter Medical AB. After hearing about the potential project, Baxter also showed interest in the mobile application and communication with Dr. Rita Nohra was established. Baxter Medical AB is an American company that specializes in medical technology.

2.5 Mobile health and remote patient monitoring

The mobile health (M-health) field has emerged as a subsection of electronic health. It is a term that refers to the use of mobile devices to practice medical and public health [10]. M-health applications may include symptom tracking, diary function or appointment reminders [10]. It allows patients to use mobile communication for health information and services to improve their health [11].

RPM includes real-time monitoring of patients' health data [12]. This allows HCPs to monitor vital signs to detect deviations, even though the patient does not reside at the hospital [12]. RPM has proven to reduce emergency visits and hospitalizations for Peritoneal dialysis patients, for visits related to nephrological problems [13]. RPM has improved particularly for patients with comorbidity [13].

3 Method

This chapter describes the method used. To carry out the project it was divided into two different phases: theoretical and development phase.

3.1 Theoretical phase

The theoretical phase began with a literature study with the purpose to familiarize with how HHD works and find research on the use of mobile applications in healthcare and the benefits of following up symptoms remotely. The difficulties and challenges that patients experienced during HHD were also examined.

According to the requirements regarding functionalities for the mobile application set by the dialysis unit at Rosenlund's hospital and Baxter, material was collected to be included in the mobile application. The dialysis unit provided checklists and logs that the patients fill in and Baxter provided manuals and failure identification codes for the Baxter AK98 dialysis machine.

To collect information from patients on HHD, home visits were conducted and semi-structured interviews with three different patients were performed. They were selected by the HHD group at Rosenlund's hospital, which consists of three nurses. The patients were selected primarily due to their experience of HHD, an inexperienced patient might provide more information. Secondly, patients of different ages and gender were picked. The patients had been on HHD for about a year. Prior to the interview, a questionnaire was compiled. After the first interview, a decision was made that the application was going to be implemented for mobile phones. Because of that and an idea from the patient regarding a possible functionality, the questionnaire was updated to keep all questions relevant for the project. To view the two questionnaires, see appendix 1 & 2. During the two remaining interviews questionnaire number two was used.

The purpose of the interviews was to get an overview of their perception of HHD. The focus was on challenges and uncertainty regarding the equipment, as well as the patients' wishes for a digital solution to collect information and features in one place. All interviews were conducted in Swedish, and notes were taken during the meetings. The functionalities that the patients requested did not make it to implementation, but they are brought up in the discussion chapter.

A plan for the user interface was drawn up from the requested functionalities from the dialysis unit at Rosenlund and Baxter, see appendix 3. This plan included how the functionalities were connected between different views in the graphical user interface.

3.2 Development phase

In the development phase the app was developed. It was produced in Xcode (Version 13.2.1 (13C100), Apple Inc) in the language SwiftUI which resulted in an iOS app for iPhone.

The implementation was divided into different parts. First, the fundamental structures were implemented for the app with the different views and navigation between these. Then, one view at a time was done with focus on the functionality it would contain. When all

functionalities were implemented, the design of a user-friendly interface was done. To clarify the functions of different buttons, icons from SF-symbols (version 3.2 (67), Apple Inc) were used.

Firebase was used to implement the function to create and save users, to chat and to save information coupled to users. Firebase is a development platform provided by google [14]. Firebase includes several different functions. To create, save and log in to the app, Firebase Authentication was used [14]. To manage user specific information such as chat messages, logs and symptom tracking, Cloud Firestore was implemented, which is an external database that updates in real-time [14]. The chat was built upon help from the tutorial “SwiftUI Firebase Real Time Chat” [15].

The app was evaluated through test running on an iPhone during the project. This to make sure all the functionalities worked correctly and to see how the graphical interface appeared for a user.

4 Results

The following chapter contains the result from the patient interviews, the information chosen to be included in the app, the development plan, and the developed mobile application. This includes how the app functionalities have been developed, and how these are related to each other in the graphical interface.

4.1 Interviews with patients

The interviews with patients confirmed the need for a digital solution to easily access information and for contacting the nurses. All three patients were sympathetic towards using an app instead of paper logs and manuals. The interviews reveal that none of the patients used the manuals nor the book with failure identification codes. All were positive to try and use them more often if the manuals would be provided in the app. The patients had several requests for further development of the app, including an order form to assist them in ordering supplies for their treatment. Another request was regarding what minerals food contains, this way they could easily decide what to eat and when to compensate for their diet. One patient wanted a video function that would facilitate communication with nurses. For complete questionnaires from conducted patient interviews, see appendix 4, 5 and 6.

4.2 Compile the information that will be included in MVP

The information that was included in the app were parts of manuals and failure identification codes provided by Baxter, and checklists provided by Rosenlund's hospital. Furthermore, symptom follow-up included information from an article from 2019 regarding what symptoms patients may experience due to their treatment [7].

4.3 Establish a development plan for a technical solution

The functionalities that ultimately made it into the app were a chat function, easily accessed information through search bars, digital dialysis protocol and symptom follow-up and the access for RPM.

A development plan resulted in a graphic illustration of the different views in the app, their functionalities and how these are connected. For a complete development plan, see appendix 3.

4.4 Finish implementation of MVP

Sign in page and create new user

The first thing that the users see when using the application is the login page. If they already have a user they can immediately log in or if they do not, they can press the button for creating a new user. When a new user is created it will automatically be a patient user, that is without the extra admin functionalities. To create an admin user, it must be done through Firebase. To view the login and signup page view figure 1.

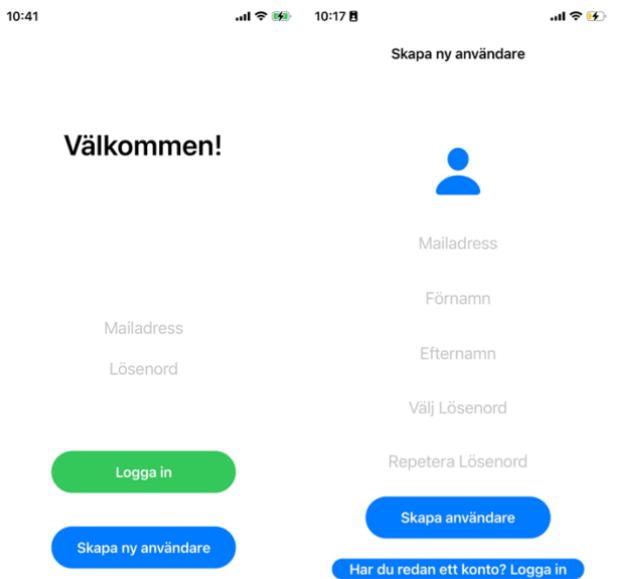


Figure 1: Login and signup page

Main menu and failure identification codes

Once the user is logged in the main menu will appear. The main menu contains five buttons. The first button is for failure identification codes, and it contains a list of codes. They can be accessed by scrolling through the page or by using the search bar. The search bar can be used either by searching for the number of the failure identification codes or by searching for a word. Once the specific code is found, by pressing the button all the information regarding the code will appear. See figure 2.

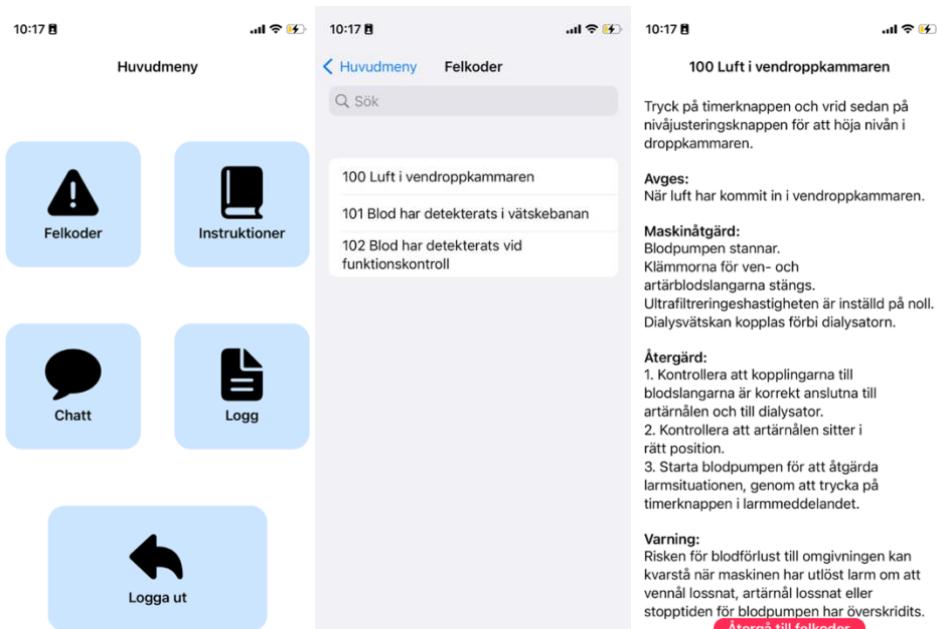


Figure 2: The picture shows the main menu page, the failure identification code page with a search bar and one example of what information the failure identification code provides.

Instructions, checklists, and manual

Another button at the main menu is for instructions. If it is pressed checklists and the manual can be accessed. The manual works equivalent to the failure identification codes with a search bar that can be used to search for the instructions needed. The checklist button contains three checklists that can be viewed, and the steps can be checked off as the patient performs all the steps. See figure 3.

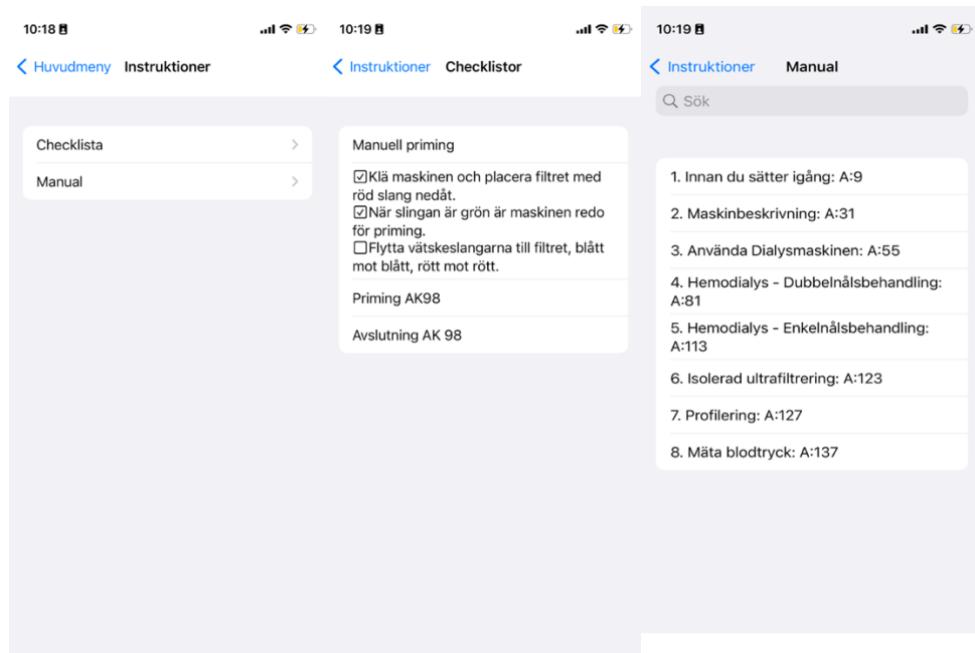


Figure 3: The picture shows the instruction page, checklists and the manual with subchapters that can be accessed through the search bar.

Chat

In the main menu, a chat button will access a chat function for patients to contact the nurses. As a regular patient user only nurses will appear to establish contact with. As for the nurses account, all patients and nurses can be accessed to chat with. The chat uses a function that continuously listens for new messages and updates the chat in real time. See figure 4.

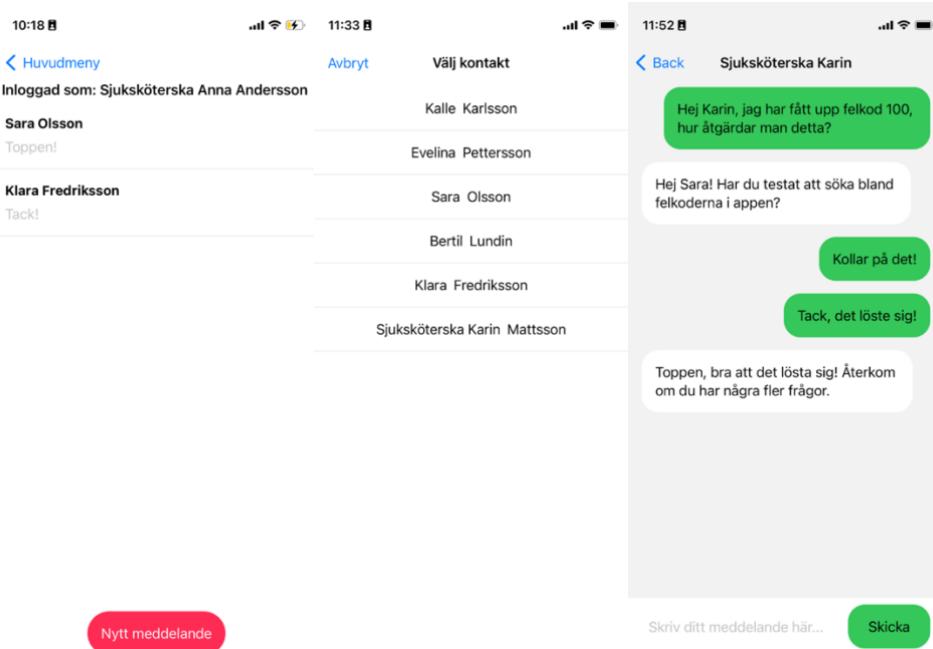


Figure 4: The picture shows the different views for the chat, the contact selection for admin and an example of a conversation.

Patient logs

The fourth button is for patient logs. During treatment they can fill in the dialysis protocol and once done, it will be saved to Firebase. Patients can then access all their old logs. During evaluation of the app, it was shown that the dialysis protocol does not work properly, as not all fields will show. They can also fill in their symptom follow-up where they grade their symptoms from a scale from one to five. The symptoms can be saved and accessed the same way as the dialysis protocol. See figure 5.

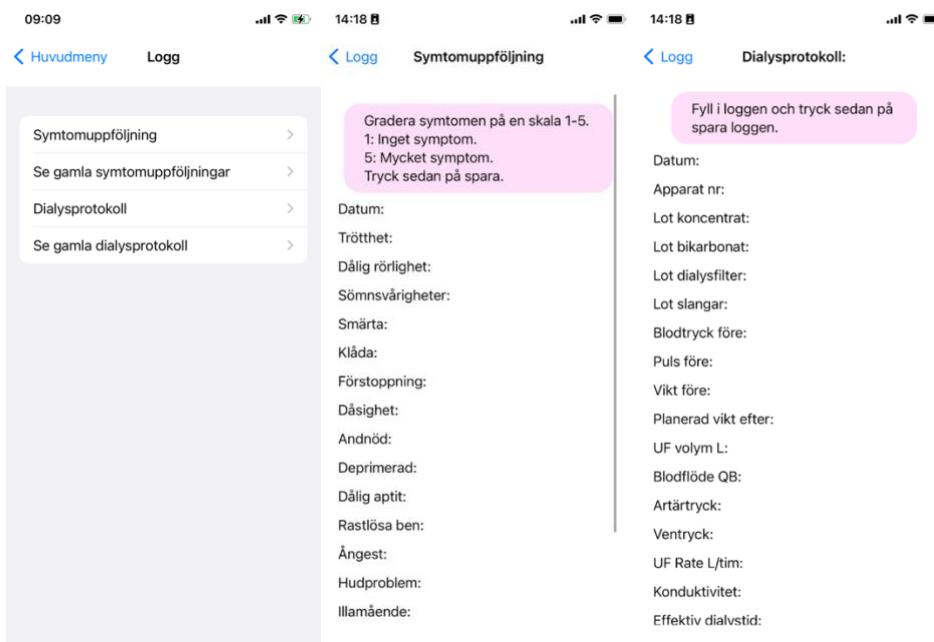


Figure 5: The picture shows a navigation list with different buttons, the symptom follow-up, and the dialysis protocol.

Patient follow-up

When admin is logged in to the app, they can access the old symptom logs and dialysis protocols. The evaluation of the app also showed the same problem as for the patients occurs here, all fields in the protocol will not show. To follow up on the complete dialysis protocol they can access these through firebase instead. See figure 6.

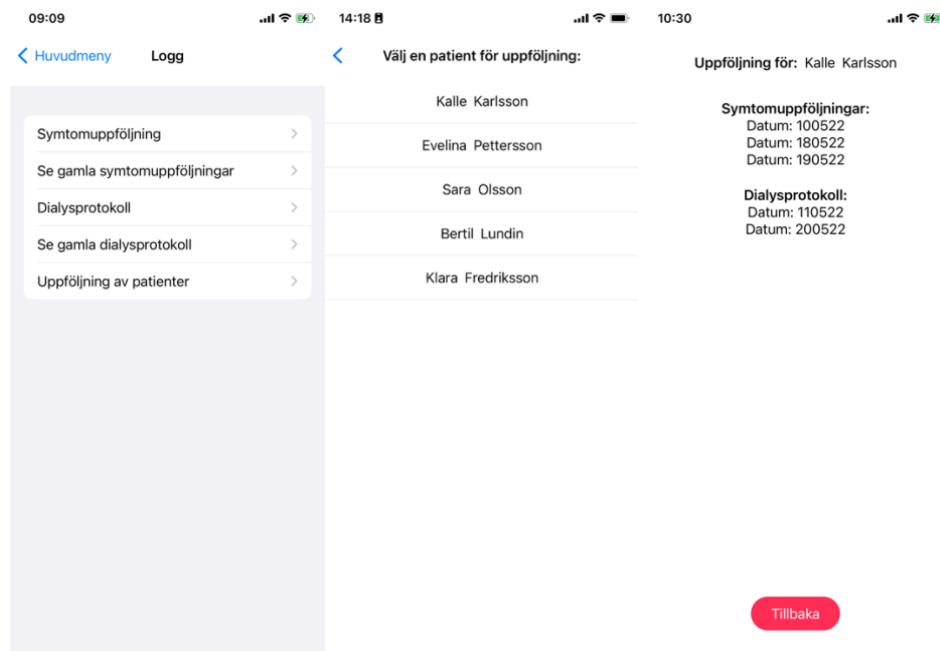


Figure 6: The picture shows what buttons an admin can access, a list with all patients to enable viewing their old logs and one of the patients' logs.

The last button on the main menu screen enables the user to log out of the app. See figure 2.

5 Discussion

As previously mentioned in the method, three of the 23 patients on HHD in Rosenlund's hospital was chosen for interviews. The reason for the limited number of interviewed patients was lack of time. Since the interviews was conducted in their homes it took a lot of time to travel and therefore a decision was made to only visit three.

The interviews with patients discovered the need for a few functionalities previously mentioned. The order form was brought up with Baxter but because it was still early in the project, they thought that it was not possible at this point. The current way to order supplies from Baxter includes calling, filling in an excel file or sending a letter. All three patients found ordering supplies sometimes troubling, with supplies missing or getting too much of some things. Creating an order form where they decide what amount of each supply they need, could be beneficial for both patients and Baxter. It could possibly lead to less misunderstandings and a faster way to collect all orders from patients because they are all in one place. It would be less time consuming for patients and employees at Baxter and more economically efficient as there are fewer shortcomings with ordering. It could also be possible to have a function that calculates how much material that needs to be ordered, based on previous orders and this way avoid ordering too much.

Collecting information to be included in the app was easy and an important part that led to the development plan. This was because the information has a strong connection to the functionalities that the app contains.

The development plan was very helpful during the project. By following the plan, one knew where to start programming and it also quickened the startup process. In addition, it became clear which functionalities that would be included and where they were going to be implemented.

As shown in the result chapter the MVP is a mobile application with several different functionalities. This enables improvements but also the potential for future development in several areas within HHD and is shown below.

The function to log in and create a new user is shown in figure 1. By creating unique users with personal data, it makes it possible to identify the person that contacted HCPs, as well as who filled in the logs. However, since the app handles personal data, a form must be produced where users approve the handling of these.

Figure 2 shows the main menu, the list with failure identification codes and how a chosen failure identification code is displayed. In the main menu, by having large buttons with icons the idea is that the user interface should be easy to use regardless of age and previous technical experience. When the user then has entered the list for failure identification codes it is easy to search and find what they are looking for. The idea with this is to solve the problem that arise during treatment in a quick and easy way without having to contact HCPs to the same extent as today. This could save both time and money for HCPs in a longer perspective.

As earlier mentioned, research has found that some patients found it troubling to search through manuals while connected to the dialysis machine. By using the search function in the app this problem now has a solution. The app allows an easy way to navigate with only one hand. This also applies to the machine manual which is shown in figure 3.

In figure 4 an image from the chat is displayed. By having a chat function, it allows patients to get in touch with HCPs regarding less urgent matters, without having to call and disturb the daily operations. It is also an opportunity for HCPs to become more efficient in their work, as it makes it possible to respond quickly without having to interrupt what they are doing.

Firebase was used to build the chat. This was a good option due to their no cost plan. Up to 5000 users can be created without additional costs [16]. Then there are costs when storage exceeds certain amounts of data [16]. Through this, the app development becomes economical since it can be implemented completely free of charge during the development stage. Before it is possible to use the app with patient data, HCPs need to look through legal framework regarding this and give their approval. However, Firebase is a good choice even going forward since it enables one to build the database in a way that suits the purpose. It is very flexible, and one could for an example use de-identification of the patients to get around handling of sensitive personal data.

In figure 5 and 6 one can see the symptom follow-up and the dialysis protocol. As is shown in the images, it is easy to navigate between old logs and symptom follow-ups as well as to fill in new ones. By having these done in the app it is a way to avoid patient misplacing papers, since everything is saved digitally. In figure 6 it is shown what the view for dialysis protocol and symptom follow-up is shown when an admin is logged in. They can also access all patients' dialysis protocols and symptom follow-ups. A disadvantage of these saved logs is that they can only show one date at a time. This makes it hard to get a good look at change over time. To improve this, one solution can be to use graphs to illustrate how certain parts of the log, such as weight and blood pressure, have changed over time. This could also be applied on the symptom follow-up function. Another problem is how often the HCPs are supposed to follow up on the dialysis protocol and the symptom follow-up. A solution for this could be to enter reference values on the various parameters and a message could be sent automatically to HCPs when the values fall outside these values. This would make it easier and less time consuming to follow up and HCPs can act quickly when encountering deviations. As previously shown some patients choose different treatment options instead of HHD because they are not monitored by HCPs. This symptom follow-up could be the solution to this making more patients feel safe with HHD.

There are still some problems linked to the dialysis protocol and the symptom follow-up. When saving the dialysis protocol to Firebase, everything is correctly saved and can be displayed through Firebase. Though, when trying to save the information from Firebase to the app an error arises. All information can be read but not saved which results in the fact that all fields in the dialysis protocol do not show up when looking through the old ones. It seems to be completely random which fields are saved or not. This has not been resolved and if one wants to follow up the dialysis protocol correctly, this must be done via Firebase until the bug is resolved. The symptom follow-up works correctly in this matter. The only limitation there is that it is possible for the users to enter all numbers 0-9 as well as several numbers when they are only supposed to choose a number between 1-5. This is something that should be resolved before the app is ready to be used.

In figure 3 the different checklists which can be ticked off are shown. The advantage of doing this in the app is that you do not have to bother printing paper that is only used once. It is also easy to correct if one has checked the wrong box, unlike on paper where you must use an eraser.

In addition to making treatment smoother for patients, the app can also benefit the environment. If all the manuals and failure identification codes, logs and checklists are available in the app, there would not be a need to print them out on paper and would therefore be environment friendly.

One problem that was encountered during the project was to read information from a file. This resulted in all information being manually written. When trying to read from the PDF that contained the manual the program could not distinguish images and therefore it was decided not to include the images in the instructions and failure identification codes. It is very time consuming to write everything manually and for future development the file handling needs to be solved to include all information in the app.

One idea to make information easily accessible was to include short videos of steps regarding the treatment. This way patients preferring auditory learning styles would find the material more appealing.

Another possible development of the app could be to implement a calendar function so the patients can set times for their treatments and thus get a reminder to fill in the checklists, log and symptom follow-up.

Before the app can reach a final stage, it should be evaluated. Through this, one can find out if the app satisfies the needs from both patients and HCPs. This would be a way to get feedback regarding both functionality and design. Furthermore, as previously mentioned in limitations the app excludes the group of people who are not iOS-users and to further develop the project one suggestion is to make a corresponding app for android devices.

6 Conclusion

All sub goals including conducting patient interviews, compiling information, establishing a development plan, and implementing the MVP were satisfied. The patient interviews confirmed the need for a mobile application during treatment. The compiling of information necessary for several functionalities contributed to the final application. By establishing a development plan, it was easy to program since it was clear what functionalities it would contain and where. Lastly, the mobile application fulfills the requested functionalities, which includes instructions of use, failure identification codes, chat function between patients and HCPs and a symptom follow-up function.

With some additional work on the mobile application, it could be submitted to App Store and fully operational. If submitted, the app could then provide HHD patients with digital information and guidance in their treatment.

7 References

- [1]“Facts About Chronic Kidney Disease,” *National Kidney Foundation*, May 15, 2020. <https://www.kidney.org/atoz/content/about-chronic-kidney-disease> (accessed Mar. 30, 2022).
- [2]“Hemodialysis, a type of dialysis,” *American Kidney Fund*, Nov. 11, 2021. <https://www.kidneyfund.org/treatments/dialysis/hemodialysis-type-dialysis> (accessed Mar. 30, 2022).
- [3]Rajkomar, Farrington, Mayer, Walker, and Blandford, “Patients’ and carers’ experiences of interacting with home haemodialysis technology: implications for quality and safety,” *BMC Nephrology*, vol. 15, no. 1, pp. 1–12, Dec. 2014, doi: 10.1186/1471-2369-15-195.
- [4]M. Pipkin *et al.*, “Recruitment and Training for Home Hemodialysis: Experience and Lessons from the Nocturnal Dialysis Trial,” *Clinical Journal of the American Society of Nephrology*, vol. 5, no. 9, pp. 1614–1620, Jun. 2010, doi: 10.2215/cjn.02440310.
- [5]“Chronic kidney disease (CKD),” *American Kidney Fund*, Nov. 06, 2021. <https://www.kidneyfund.org/all-about-kidneys/chronic-kidney-disease-ckd#what-causes-ckd> (accessed Mar. 30, 2022).
- [6]“Hemodialysis,” *NIDDK / National Institute of Diabetes and Digestive and Kidney Diseases*, Dec. 09, 2021. Accessed: Mar. 30, 2022. [Online]. Available: <https://www.niddk.nih.gov/health-information/kidney-disease/kidney-failure/hemodialysis>
- [7]J. T. Moskovitch, P. F. Mount, and M. R. P. Davies, “Changes in Symptom Burden in Dialysis Patients Assessed Using a Symptom-Reporting Questionnaire in Clinic,” *Journal of Palliative Care*, vol. 35, no. 1, pp. 59–65, Feb. 2019, doi: 10.1177/0825859719827315.
- [8]“Home Hemodialysis,” *National Kidney Foundation*, Dec. 24, 2015. <https://www.kidney.org/atoz/content/homehemo> (accessed Mar. 30, 2022).
- [9]M. Ageborg, B.-L. Allenius, and C. Cederfjall, “Quality of life, self-care ability, and sense of coherence in hemodialysis patients: A comparative study,” *Hemodialysis International*, vol. 9, no. s4, pp. S8–S14, Oct. 2005, doi: 10.1111/j.1542-4758.2005.01164.x.
- [10]C. Hollis *et al.*, “Technological innovations in mental healthcare: harnessing the digital revolution,” *British Journal of Psychiatry*, vol. 206, no. 4, pp. 263–265, Apr. 2015, doi: 10.1192/bjp.bp.113.142612.
- [11]M. Nacinovich, “Defining mHealth,” *Journal of Communication in Healthcare*, vol. 4, no. 1, pp. 1–3, Apr. 2011, doi: 10.1179/175380611x12950033990296.
- [12]C. Innovations, “What Is Telehealth? What Is Remote Patient Monitoring? How Are They Different?” <https://news.careinnovations.com/blog/what-is-telehealth-what-is-remote-patient-monitoring-how-are-they-different> (accessed May 19, 2022).
- [13]S. Milan Manani *et al.*, “Remote monitoring in peritoneal dialysis: benefits on clinical outcomes and on quality of life,” *Journal of Nephrology*, vol. 33, no. 6, pp. 1301–1308, Aug. 2020, doi: 10.1007/s40620-020-00812-2.

[14]“Firebase Documentation,” *Firebase*. <https://firebase.google.com/docs/auth> (accessed May 20, 2022).

[15]“Lets Build That App.”
<https://www.letsbuildthatapp.com/course/SwiftUI%20Firebase%20Real%20Time%20Chat> (accessed May 20, 2022).

[16]“Firebase Pricing,” *Firebase*. <https://firebase.google.com/pricing> (accessed May 20, 2022).

Appendix 1: Frågor vid hembesök, version 1

Vilken typ av utbildning fick du när du skulle börja med hemhemodialys?
Kände du att du hade fullständiga kunskaper när du kom hem?

Vi ska ta fram en app för att hjälpa till vid användning av utrustningen hemifrån. Skulle du ha några särskilda önskemål kring vad denna ska innehålla?

Tanken är att den bland annat ska innehålla instruktioner kring användningen för att fungera som stöd. Skulle du föredra att ha detta i text, inspelad video eller bilder med text? Skulle du föredra att ha appen på mobilen eller datorn?

Hur ofta har du varit tvungen att kontakta mottagningen för hjälp eller frågor kring utrustning eller genomförande av behandling? Är detta något som skulle kunnat undvikas om du haft mer kunskap eller information tillgänglig?

Har detta främst varit akuta frågor eller mindre akut/generella funderingar?

Tanken är att appen ska innehålla en kommunikationsfunktion mellan patient och vårdgivare i form av till exempel en chatt. Är detta något du skulle kunna dra nytta av som alternativ till att ringa?

Hur ser informationen ut kring användandet, rengöring, utrustning, felsökning?
Är det någon del där informationen inte är tillräcklig?

Har du några personliga tips på hur man kan göra användandet lite lättare

Appendix 2: Frågor vid hembesök, version 2

Vilken typ av utbildning fick du när du skulle börja med hemhemodialys?
Kände du att du hade fullständiga kunskaper när du kom hem?

Vi ska ta fram en app för att hjälpa till vid användning av utrustningen hemifrån. Skulle du ha några särskilda önskemål kring vad denna ska innehålla?

Tanken är att den bland annat ska innehålla instruktioner kring användningen för att fungera som stöd. Skulle du föredra att ha detta i text, inspelad video eller bilder med text?

Hur ofta har du varit tvungen att kontakta mottagningen för hjälp eller frågor kring utrustning eller genomförande av behandling? Är detta något som skulle kunnat undvikas om du haft mer kunskap eller information tillgänglig?

Har detta främst varit akuta frågor eller mindre akut/generella funderingar?

Tanken är att appen ska innehålla en kommunikationsfunktion mellan patient och vårdgivare i form av till exempel en chatt. Är detta något du skulle kunna dra nytta av som alternativ till att ringa?

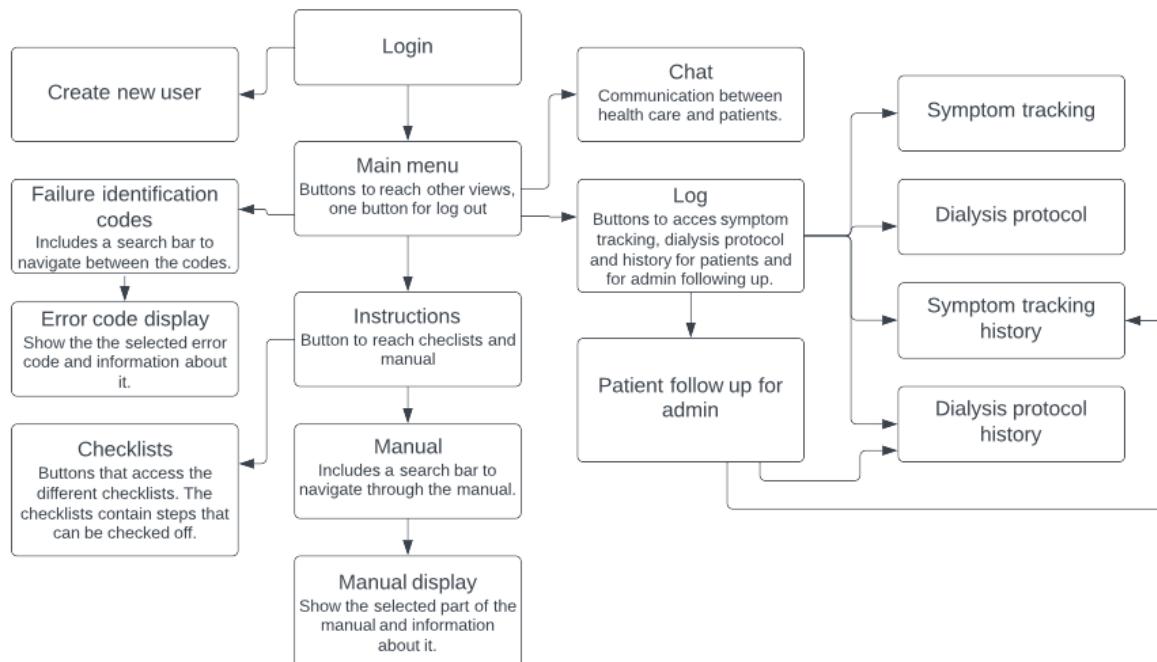
Hur ser informationen ut kring användandet, rengöring, utrustning, felsökning?
Är det någon del där informationen inte är tillräcklig?

Har du några personliga tips på hur man kan göra användandet lite lättare?

Hur tycker du att beställning av material fungerar idag?
Skulle ett webb-formulär kunna vara ett alternativ?

Om sjukvården kan följa upp symptom/ tillstånd före och efter behandling för att kunna fånga upp varningssignaler, skulle det känna tryggare för dig då?

Appendix 3: Development plan for technical solution



Appendix 4: Patient interview, patient 1, man 67 years

Datum: 18 mars

**Vilken typ av utbildning fick du när du skulle börja med hemhemodialys?
Kände du att du hade fullständiga kunskaper när du kom hem?**

Han fick lära sig om underhåll som t.ex. filterbyte på avdelningen, utbildningen fungerade smidigt. Han hade en positiv upplevelse av utbildningen från sjukhuset och upplevde att sjuksköterskorna var väldigt hjälpsamma. Avdelningen är tillgänglig på telefon från 7-21 varje dag.

**Vi ska ta fram en app för att hjälpa till vid användning av utrustningen hemifrån.
Skulle du ha några särskilda önskemål kring vad denna ska innehålla?**

Han upplever att det i nuläget är svårt att beställa rätt mängd av allt förbrukningsmaterial till maskinen, då det ligger olika många förbrukningsmaterial i varje låda för olika produkter. I dagsläget beställer han en gång i månaden från Baxter. Han anser att det skulle underlättat om man hade en återkommande beställning var 14:e dag för att slippa ha så mycket material hemma. Samt att beställningen skulle innehålla ett färdigt paket med allt i. Förutom vilken nål man vill ha, den skulle isäfall beställas separat.

Han tog upp sin vikt som ett exempel. Efter jul hade han gått upp väldigt mycket, men trodde det berodde på ett större matintag. Han tror att om detta sparats online och personalen fått tillgång till det, hade varningssignaler kunnat uppmärksammats tidigare. Han tycker att digitala checklistor är en bra idé.

Tanken är att den bland annat ska innehålla instruktioner kring användningen för att fungera som stöd. Skulle du föredra att ha detta i text, inspelad video eller bilder med text? Skulle du föredra att ha appen på mobilen eller datorn?

Han skulle föredra korta videos och ibland illustrationer tillsammans med text. Han tycker att det är bra att ha flera alternativ, eftersom det skulle finnas något som passar alla. Enheten skulle inte spela någon roll, dock tycker han att man ser bättre på en dator. Men i slutänden är det viktigaste att den är användarvänlig.

**Hur ofta har du varit tvungen att kontakta mottagningen för hjälp eller frågor kring utrustning eller genomförande av behandling? Är detta något som skulle kunnat undvikas om du haft mer kunskap eller information tillgänglig?
Har detta främst varit akuta frågor eller mindre akut/generella funderingar?**

De flesta felkoder brukar han oftast lösa på egen hand. Han ringer om det uppstår akuta situationer.

Tanken är att appen ska innehålla en kommunikationsfunktion mellan patient och vårdgivare i form av till exempel en chatt. Är detta något du skulle kunna dra nytta av som alternativ till att ringa?

Han gillar att ringa och de gånger han kontaktat avdelningen för hjälp, har det varit akuta grejer. Avdelningen kan då guida en till vad man ska göra.

Hur ser informationen ut kring användandet, rengöring, utrustning, felsökning?
Är det någon del där informationen inte är tillräcklig?

Han upplever informationen som tillräcklig. Han tycker att checklistorna är användningsbara när man är ny med hemhemodialys.

Har du några personliga tips på hur man kan göra användandet lite lättare?

Hans tips är att vara aktiv under dialySEN och att hemhemodialys inte passar alla, då det är mycket att ha koll på och viktigt att allt utförs i rätt ordning

Appendix 5: Patient interview, patient 2, man 50 years

Datum: 1 april

**Vilken typ av utbildning fick du när du skulle börja med hemhemodialys?
Kände du att du hade fullständiga kunskaper när du kom hem?**

Han fick 3,5 månaders utbildning på Rosenlund, när han genomförde detta upplevde han det som för lång tid. Han tyckte att man lärde sig allt på en vecka. Fördelen med längre utbildning var att man kunde upptäcka fel. Med facit i hand tycker han utbildningstiden var rimlig då det gav möjlighet att göra misstag på plats och testa sig fram där stöd fanns. Han jobbar 100% och det fungerar bra med hemhemodialys. Han har vart hemma i ca 1 år.
Han tycker att informationen som finns är bra för otekniska.

**Vi ska ta fram en app för att hjälpa till vid användning av utrustningen hemifrån.
Skulle du ha några särskilda önskemål kring vad denna ska innehålla?**

Han gillade idén med att ha checklista och logg digitalt, han förstod inte hur detta inte kunnat digitaliseras ännu. Han skulle välja app framför papper.

Tanken är att den bland annat ska innehålla instruktioner kring användningen för att fungera som stöd. Skulle du föredra att ha detta i text, inspelad video eller bilder med text?

Han tycker att bilder är bra för att ge en bättre förståelse. Han tycker att video kan vara jobbigt ibland och likaså för långa texter. Han tror att det är bra med en lagom blandning, samt att användningen av appar är en generationsfråga.

**Hur ofta har du varit tvungen att kontakta mottagningen för hjälp eller frågor kring utrustning eller genomförande av behandling? Är detta något som skulle kunnat undvikas om du haft mer kunskap eller information tillgänglig?
Har detta främst varit akuta frågor eller mindre akut/generella funderingar?**

Vid ett tillfälle har en akut situation uppstått. Annars har han haft mer generella frågor som inte behövts besvarats direkt. Han föredrar att kommunicera via sms.

Tanken är att appen ska innehålla en kommunikationsfunktion mellan patient och vårdgivare i form av till exempel en chatt. Är detta något du skulle kunna dra nytta av som alternativ till att ringa?

Han tycker chatten låter superbra. Han frågade hur tanken var att sjukvården skulle vara tillgängliga via chatten.

**Hur ser informationen ut kring användandet, rengöring, utrustning, felsökning?
Är det någon del där informationen inte är tillräcklig?**

Han använder sig ej av manualer men tycker det är bra att ta med i appen.
Han får inte fel koder så ofta och då de uppstår förstår han oftast vad problemet är.

Har du några personliga tips på hur man kan göra användandet lite lättare?

Han tycker att det var bra saker vi tagit upp och att det skulle vara smidigt att ha allt samlat på ett ställe. Han ger tipset att skapa en användarvänlig app och tänka på att appen inte ska bli rörig.

Hur tycker du att beställning av material fungerar idag?

Skulle ett webb-formulär kunna vara ett alternativ?

Idag har han har en excelfil som han fyller i och skickar iväg tillsammans med det önskade datumet för leverans. Han får ofta bekräftelse från Baxter att beställningen har gått igenom. Han beställer en gång i månaden men han vet att man även får beställa var 14:e dag. Han tycker en gång i månaden är skönt, då en hel dag går åt att vänta på leverans och packa upp den.

Han tycker i dagsläget att beställningen fungerar helt okej. Ibland saknas dock grejer till följd av att Baxter t.ex. endast läst sida 1-2 men missat sida 3.

Om sjukvården kan följa upp symptom/ tillstånd före och efter behandling för att kunna fånga upp varningssignaler, skulle det känna tryggare för dig då?

Han känner inte att detta skulle gynna honom då han inte känner sig påverkad av dialysbehandlingen. Han mår likadant innan som efter behandling. Det största problemet med sjukdomen är att kroppen samlar på sig vatten. Han tror att symptomuppföljning kan vara ett bra sätt för sjukvården att få feedback.

Appendix 6: Patient interview, patient 3, woman 73 years

Datum: 5 april

Vilken typ av utbildning fick du när du skulle börja med hemhemodialys?

Kände du att du hade fullständiga kunskaper när du kom hem?

Hon var inte förberedd på de fel som skulle uppstå, hon kände sig osäker i början. Det uppstod flera nya fel men det löste sig genom att hon kunde ringa avdelningen. Hon tycker att det är svårt och opraktiskt att bläddra i boken. Hon saknar dock att jourtelefon inte har en facetime-funktion, eftersom det ibland är svårt att förstå vad man ska göra. Hon tycker att det skulle underlättat om hon kunde filma varje steg. I dagsläget kan hon fota och skicka bild, hon tycker att det känns tryggare om personalen på avdelningen kan se vad hon gör. Hon har varit hemma med dialys i ett års tid.

Avdelningens telefon är ej tillgänglig på natten. Då problem uppstår under natten, avbryter hon behandlingen istället och ringer dagen efter.

Hon genomför dialys 5 gånger i veckan. Hon upplever stor frihet trots att man samtidigt är bunden till hemmet. Innan hon fick börja med hemhemodialys höll hon på att gå under av depression. Hon upplever att hon fick bra information från sjukhuset under utbildningen.

**Vi ska ta fram en app för att hjälpa till vid användning av utrustningen hemifrån.
Skulle du ha några särskilda önskemål kring vad denna ska innehålla?**

Hon skulle absolut kunna tänka sig att använda en app. Hon tror att det skulle kunna vara viktigt att ha i början, när man är osäker. Hon tycker att det är bra med checklistor eftersom hon glömde heparin en gång. Hon tycker att en bra funktion skulle vara om appen kan beräkna hur mycket hon behöver kompensera för den mat som intas. I dagsläget måste hon googla för att ta reda på hur mycket kalium t.ex. choklad och mango innehåller.

Tanken är att appen, bland annat ska innehålla instruktioner kring användningen för att fungera som stöd. Skulle du föredra att ha detta i text, inspelad video eller bilder med text?

Hon tror att video skulle vara bra eftersom hennes ålder skulle göra så att hon skulle uppleva att det går för fort. Hon föredrar bild med text eftersom det är bättre för äldre mäniskor.

Hur ofta har du varit tvungen att kontakta mottagningen för hjälp eller frågor kring utrustning eller genomförande av behandling? Är detta något som skulle kunnat undvikas om du haft mer kunskap eller information tillgänglig?

Har detta främst varit akuta frågor eller mindre akut/generella funderingar?

I början uppstod akuta situationer 2 gånger i veckan. Senaste gången hon behövde hjälp var för 6 veckor sen.

Tanken är att appen ska innehålla en kommunikationsfunktion mellan patient och vårdgivare i form av till exempel en chatt. Är detta något du skulle kunna dra nytta av som alternativ till att ringa?

Hon skulle absolut använda sig av det.

**Hur ser informationen ut kring användandet, rengöring, utrustning, felsökning?
Är det någon del där informationen inte är tillräcklig?**

Om det fanns information om filterbyten i appen, skulle hon kunna tänka sig att använda sig av den. Hon använder aldrig manual för vare sig felkoder eller stegen i dialysbehandlingen.

Har du några personliga tips på hur man kan göra användandet lite lättare?

Hon tror att med en app skulle fler våga ha hemhemodialys, samt att det skulle bli billigare för vården. Hon tycker fler borde ta chansen att genomföra behandlingen hemma.
Hon vill ha nattdialys, men på grund av en olycka för 2 år sen så får patienter inte längre ha det.

Ibland vaknar hon mitt i natten för att avsluta behandlingen och det tycker hon skulle vara skönt att slippa.

**Hur tycker du att beställning av material fungerar idag?
Skulle ett webb-formulär kunna vara ett alternativ?**

I dagsläget fyller hon i papper från Baxter och mailar in detta. Hon skulle vilja ha en funktion där det går att trycka in vad man vill ha. När man ringer till Baxter för att beställa hamnar man alltid i telefonkö, vilket hon upplever som något negativt. Förbrukningsartiklarna är lätt att få hem men det uppstår ibland fel vid leveransen.

Om sjukvården kan följa upp symptom/ tillstånd före och efter behandling för att kunna fånga upp varningssignaler, skulle det känna tryggare för dig då?

Hon skulle inte ha något emot att sjukvården följer upp hennes symptom. Hon tycker att det vore smidigare att lägga in i en app. Idag ser sjukvården hennes logg var 3:e månad då hon åker in för besök hos sjukvården.

Appendix 7: Programming code

```
//  
// MainMenu.swift  
// Examensarbete  
//  
// Created by Matilda Byström on 2022-04-04.  
  
import SwiftUI  
  
struct MainMenu: View {  
    @State var userIsLoggedIn = false  
    var body: some View {  
  
        VStack{  
            Spacer()  
            HStack {  
                NavigationLink(destination: FailureIdentificationCodes())  
                {  
                    VStack{  
                        Image(systemName:  
                            "exclamationmark.triangle.fill").font(.system(size:60))  
                            .frame(width: 0, height: 60)  
                        Text("Felkoder")  
                            .font(.headline).foregroundColor(.black).padding().frame(width: 160,  
height: 150).background(Color(red: 204/255, green: 229/255, blue: 255/255))  
                            .cornerRadius(20).padding().buttonStyle(PlainButtonStyle())  
                }  
  
                NavigationLink(destination: Instructions())  
                {  
  
                    VStack{  
                        Image(systemName: "book.closed.fill").font(.system(size:60))  
                            .frame(width: 0, height: 60)  
                        Text("Instruktioner")  
                            .font(.headline).foregroundColor(.black).padding().frame(width: 160, height:  
150).background(Color(red: 204/255, green: 229/255, blue: 255/255))  
                            .cornerRadius(20).padding().buttonStyle(PlainButtonStyle())  
                }  
            }  
        }  
        Spacer()  
        HStack {  
  
            NavigationLink(destination: Chat())  
            {  
  
            }  
        }  
    }  
}
```

```

VStack{
    Image(systemName: "message.fill").font(.system(size:60))
        .frame(width: 0, height: 60)
    Text("Chatt")
}.font(.headline).foregroundColor(.black).padding().frame(width: 160, height: 150).background(Color(red: 204/255, green: 229/255, blue: 255/255))
    .cornerRadius(20).padding().buttonStyle(PlainButtonStyle())

}

NavLink(destination: Log())
{

VStack{
    Image(systemName: "doc.text.fill").font(.system(size:60))
        .frame(width: 0, height: 60)
    Text("Logg")
}.font(.headline).foregroundColor(.black).padding().frame(width: 160, height: 150).background(Color(red: 204/255, green: 229/255, blue: 255/255))
    .cornerRadius(20).padding().buttonStyle(PlainButtonStyle())
}

Spacer()
HStack {

    Button{
        self.userIsLoggedIn.toggle()
    } label: {

        VStack{
            Image(systemName: "arrowshape.turn.up.left.fill").font(.system(size:60))
                .frame(width: 0, height: 60)
            Text("Logga ut")
}.font(.headline).foregroundColor(.black).padding().frame(width: 260, height: 150).background(Color(red: 204/255, green: 229/255, blue: 255/255))
    .cornerRadius(20).padding().buttonStyle(PlainButtonStyle())
}

.fullScreenCover(isPresented: $userIsLoggedIn, content: ContentView.init)

}

```

```

        }.navigationBarBackButtonHidden(true) .navigationBarTitle("Huvudmeny",
displayMode: .inline)
.padding()

}

private func handleAction(){
    self.userIsLoggedIn = true
}

}

struct MainMenu_Previews: PreviewProvider {
    static var previews: some View {
        MainMenu()
    }
}

// KEXnyApp.swift
// KEXny
//
// Created by Matilda Byström on 2022-04-13.
//


import SwiftUI

@main
struct KEXnyApp: App {

    var body: some Scene {
        WindowGroup {
            ContentView()
        }
    }
}

//
// ContentView.swift
// Examensarbete
//
// Created by Matilda Byström on 2022-04-04.
//


import SwiftUI
import Firebase
import FirebaseFirestore

```

```

struct ContentView: View {

    @State var isLoginMode = true
    @State private var username = ""
    @State private var password = ""
    @State private var Repeatepassword = ""
    @State private var Surname = ""
    @State private var Lastname = ""
    @State private var Admin = "false"

    @State var passWordIncorrect = false
    @State var passWordNoLength = false

    @State var returnToMainMenu: Bool = false
    @State var authentificationFailed: Bool = false
    @State var userCreated: Bool = false
    @State var userNotCreated: Bool = false
    @State var showMainMenu: Bool = false
    @State var passwordNotSame: Bool = false

    var body: some View {
        NavigationView{
            ZStack{
                VStack (alignment: .center) {
                    if(isLoginMode == true)
                    {
                        Welcome()
                        VStack(alignment: .center){
                            if authentificationFailed {
                                Text("Fel användarnamn eller lösenord").foregroundColor(.red).offset(y: -100)
                            }
                            if userCreated{
                                Text("Användaren har skapats.")
                                    .foregroundColor(.green).offset(y: -100)
                            }
                        UsernameTextfield(username: $username).multilineTextAlignment(.center)
                            .padding()
                            .font(.system(size:20))
                    }
                    PasswordTextfield(password: $password).multilineTextAlignment(.center)
                }
            }
        }
    }
}

```

```

        .font(.system(size:20))
    }.frame(width: 400, height: 250)

}

NavigationLink(destination: MainMenu(), isActive: $showMainMenu){}
Button{ } label: {
    HStack{
        Spacer()
        Text("Logga in")
            .foregroundColor(.white)
            .padding()
        Spacer()
    }.font(.headline).foregroundColor(.white).padding().frame(width: 220,
height: 50).background(.green).cornerRadius(40).offset(y: -10).padding()
        .onTapGesture{
            handleAction()
    }
}

Text("Skapa ny användare")
    .onTapGesture {
        self.userCreated = false
        isLoginPage = false
    }.font(.headline).foregroundColor(.white).padding().frame(width: 220,
height: 50).background(.blue).cornerRadius(40).offset(y: -10).padding()
}

}

if(isLoginPage == false)
{

Text("")
    .navigationBarTitle("Skapa ny användare", displayMode: .inline)
    .navigationBarBackButtonHidden(false)
VStack {

Image(systemName: "person.fill")
    .font(.system(size: 64))
    .padding()

if userNotCreated == true {

```

```

Text("Användaren kunde ej skapas").foregroundColor(.red)

if passwordNotSame == true
{
    Text("Lösenorden stämmer ej överens").foregroundColor(.red)
}
}

```

```

VStack{

    TextField(text: $username, prompt: Text("Mailadress")){
        }
        .multilineTextAlignment(.center)
        .padding()
        .font(.system(size:20))
    }
}

```

```

    TextField(text: $Surname, prompt: Text("Förnamn")){
        }
        .multilineTextAlignment(.center)
        .padding()
        .font(.system(size:20))
    }
}

```

```

    TextField(text: $Lastname, prompt: Text("Efternamn")){
        }
        .multilineTextAlignment(.center)
        .padding()
        .font(.system(size:20))
    }
}

```

```

    SecureField(text: $password, prompt: Text("Välj Lösenord")){
        }
        .multilineTextAlignment(.center)
        .padding()
        .font(.system(size:20))
    }
}

```

```

    SecureField(text: $Repeatepassword, prompt: Text("Repetera Lösenord")){
        }
        .multilineTextAlignment(.center)
        .padding()
        .font(.system(size:20))
    }
}

```

```

Button{} label: {
    HStack{
        Spacer()
        Text("Skapa användare")
        .foregroundColor(.white)
    }
}

```

```

        .padding()
        Spacer()
    }.font(.headline).foregroundColor(.white).padding().frame(width: 220,
height: 50).background(.blue).cornerRadius(40).offset(y: -10).padding()
        .onTapGesture{

    if password == Repeatepassword {
        self.passwordNotSame = false
        handleAction()
    }

    else
    {
        self.passwordNotSame = true
        userNotCreated = true
    }

}

Text("Har du redan ett konto? Logga in")
    .font(.headline).foregroundColor(.white).padding().frame(width: 300,
height: 25).background(.blue).cornerRadius(40).offset(y: -10)
    .onTapGesture {
        isLoginPage = true
    }

}
}.foregroundColor(.blue)
}

}.navigationViewStyle(StackNavigationViewStyle())
}

}

private func handleAction(){
if isLoginPage{
    logIn()
}
else{
    createAccount()
}
}

```

```

private func logIn(){
    FirebaseDealer.shared.auth.signIn(withEmail: username, password: password){ result,
        err in

        if let err = err {
            authenticationFailed = true
            print(err)
            return
        }

        self.showMainMenu = true

    }
}

private func createAccount(){
    FirebaseDealer.shared.auth.createUser(withEmail: username, password: password){
        result, err in
        if let err = err {

            self.userNotCreated = true
            print(err)
            return
        }

        self.userCreated = true
        self.userNotCreated = false
        self.saveUserInfoToFirebase(surname: Surname, lastname: Lastname, username:
username)
        username = ""
        self.password = ""
        self.Repeatepassword = ""
        self.Surname = ""
        self.Lastname = ""
        self.isLoginMode = true

    }
}

private func saveUserInfoToFirebase(surname: String, lastname: String, username: String)
{
    guard let uid = FirebaseDealer.shared.auth.currentUser?.uid else{ return }

    let userData = ["Admin": "false", "Email": self.username, "Lastname": lastname,
"Surname": surname, "uid": uid] as [String : Any]
    FirebaseDealer.shared.firestore.collection("users")
        .document(uid).setData(userData){ err in
            if let err = err{

```

```

        print(err)
        return
    }

}

}

struct UsernameTextfield: View{
    @Binding var username: String
    var body: some View{
        TextField(text: $username, prompt: Text("Mailadress")){
        }
    }
}

struct PasswordTextfield: View{
    @Binding var password: String
    var body: some View{
        SecureField(text: $password, prompt: Text("Lösenord")){
        }
    }
}

```

```

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}

struct Welcome: View{
    var body: some View{
        return Text("Välkommen!")
            .font(.largeTitle).fontWeight(.semibold).padding(.bottom, 80)
    }
}

// Felkoder.swift
// Examensarbete
//
// Created by Matilda Byström on 2022-04-04.
//
```

```

import SwiftUI

struct FailureIdentificationCodes: View {
    private var a = FailureCodeClass()

```

```

@State var searchText = ""
private var codeName = ""
@State var codeNameShow = ""
@State var showDetailsCode = false
@State var failureCodeIndex = -1

var body: some View {

    if showDetailsCode == false{
        Form{

            ForEach(a.searchResults(searchText: searchText), id: \.self){ codeName in

                Text(codeName).onTapGesture{
                    self.showDetailsCode = true
                    self.codeNameShow = codeName
                }
            }

            }.searchable(text: $searchText, placement: .navigationBarDrawer(displayMode: .always), prompt: "Sök")
            Spacer()

        Text("").navigationBarTitle("Felkoder")

        Spacer()
    }

    else if showDetailsCode == true {

        ForEach(0..<a.failureCodeOnlyNameArray.count, id: \.self){ index in

            if codeNameShow == a.failureCodeOnlyNameArray[index]
            {

                ScrollView{
                    VStack{

                        VStack{
                            Text(a.failureCodeArray[index].machineWarning)
                                .frame(maxWidth: .infinity, alignment: .leading)
                                .fixedSize(horizontal: false, vertical: true)

                            Text("\nAvges:").bold()
                                .frame(maxWidth: .infinity, alignment: .leading)
                        }
                    }
                }
            }
        }
    }
}

```

```

Text(a.failureCodeArray[index].released)
    .frame(maxWidth: .infinity, alignment: .leading)
    .fixedSize(horizontal: false, vertical: true)
}

Text("\nMaskinåtgärd:").bold()
    .frame(maxWidth: .infinity, alignment: .leading)
Text(a.failureCodeArray[index].machineArrangement)
    .frame(maxWidth: .infinity, alignment: .leading)
    .fixedSize(horizontal: false, vertical: true)

Text("\nÅtergård:").bold()
    .frame(maxWidth: .infinity, alignment: .leading)
Text(a.failureCodeArray[index].arrangement)
    .frame(maxWidth: .infinity, alignment: .leading)
    .fixedSize(horizontal: false, vertical: true)

Text("\nVarning:").bold()
    .frame(maxWidth: .infinity, alignment: .leading)
Text(a.failureCodeArray[index].warning)
    .frame(maxWidth: .infinity, alignment: .leading)
    .fixedSize(horizontal: false, vertical: true)

Spacer()

Text("Återgå till felkoder").bold()
    .frame(maxWidth: .infinity, alignment: .leading)
    .font(.headline).foregroundColor(.white).padding().frame(width: 180,
height: 25).background(.pink).cornerRadius(40).offset(y: -10)
    .onTapGesture{
        showDetailsCode = false
    }

}

.navigationBarTitle(codeNameShow)
.navigationBarBackButtonHidden(true)
.navigationBarHidden(false)

Spacer()
}.padding(15)
}

}

```

```

}

struct FailureIdentificationCodes_Previews: PreviewProvider {
    static var previews: some View {
        FailureIdentificationCodes()
    }
}

//
// FailureCodeClass.swift
// Examensarbete
//
// Created by Matilda Byström on 2022-04-07.
//

import Foundation

class FailureCodeClass: ObservableObject{

    public class FailureCode{
        @Published var nameErrorCode: String
        @Published var machineWarning: String
        @Published var released: String
        @Published var machineArrangement: String
        @Published var arrangement: String
        @Published var warning: String

        init(nameErrorCode: String, machineWarning: String, released: String,
        machineArrangement: String, arrangement: String, warning: String) {
            self.nameErrorCode = nameErrorCode
            self.machineWarning = machineWarning
            self.released = released
            self.machineArrangement = machineArrangement
            self.arrangement = arrangement
            self.warning = warning
        }
    }

    var failureCodeArray = [FailureCode(nameErrorCode: "100 Luft i vendroppkammaren.",
    machineWarning: "Tryck på timerknappen och vrid sedan på nivåjusteringssknappen för att
    höja nivån i droppkammaren.", released: "När luft har kommit in i vendroppkammaren.",
    machineArrangement: "Blodpumpen stannar. \nKlämmorna för ven- och artärblodslangarna
    stängs. \nUltrafiltreringeshastigheten är inställd på noll. \nDialysvätskan kopplas förbi
    dialysatorn.", arrangement: "1. Kontrollera att kopplingarna till blodslangarna är korrekt
    anslutna till artärnålen och till dialysator. \n2. Kontrollera att artärnålen sitter i rätt position.
    \n3. Starta blodpumpen för att åtgärda larmsituationen, genom att trycka på timerknappen i
    ")
}

```

larmmeddelandet.", warning: "Risken för blodförlust till omgivningen kan kvarstå när maskinen har utlöst larm om att vennål lossnat, artärnål lossnat eller stopptiden för blodpumpen har överskridits."), FailureCode(nameErrorCode: "101 Blod har detekterats i vätskebanan", machineWarning: "Om du vill...", released: "När blod har kommit in i maskinens vätskebana nedströms från dialysatorn.", machineArrangement: "Blodpumpen stannar.", arrangement: "Kontrollera om det finns blod i...", warning: "Varning"), FailureCode(nameErrorCode: "102 Blod har detekterats vid funktionskontroll", machineWarning: "Blod i primingdetektorn.", released: "När primingdetektor detekterar blod under funktionskontrollen.", machineArrangement: "Blodpumpen stannar", arrangement: "Kontrollera först att patienten inte är kopplad till blodslangarna.", warning: "Varning")]

```
var failureCodeOnlyNameArray = ["100 Luft i vendroppkammaren", "101 Blod har detekterats i vätskebanan", "102 Blod har detekterats vid funktionskontroll"]
```

```
func copyNames()
{
    for index in 0...failureCodeArray.count-1 {
        failureCodeOnlyNameArray.insert(failureCodeArray[index].nameErrorCode, at: failureCodeOnlyNameArray.endIndex)
    }
}
```

```
func searchResults(searchText: String) -> [String]
{
    if searchText.isEmpty{
        return failureCodeOnlyNameArray
    }
    else{
        //Skickar tillbaka en filtrerad lista beroende på vad SearchText innehåller
        return failureCodeOnlyNameArray.filter({
            $0.localizedCaseInsensitiveContains(searchText) })
    }
}
```

```
//  
// Felkoder.swift  
// Examensarbete  
//
```

```

// Created by Matilda Byström on 2022-04-04.
//

import SwiftUI

struct Manual: View {
    @State var ManualArray = ["1. Innan du sätter igång: A:9", "2. Maskinbeskrivning: A:31",
    "3. Använda Dialysmaskinen: A:55", "4. Hemodialys - Dubbelnålsbehandling: A:81", "5.
    Hemodialys - Enkelnålsbehandling: A:113", "6. Isolerad ultrafiltrering: A:123", "7.
    Profilering: A:127", "8. Mäta blodtryck: A:137"]
    @State var searchText = ""
    private var manualName = ""
    @State var manualNameShow = ""
    @State var showDetail = false

    var body: some View {

        if showDetail == false{
            Form{
                ForEach(searchResults(searchText: searchText), id: \.self){ codeName in
                    Text(codeName).onTapGesture{
                        self.showDetail = true
                        self.manualNameShow = codeName
                    }
                }
                .searchable(text: $searchText, placement: .navigationBarDrawer(displayMode:
                .always), prompt: "Sök")
                Spacer()
            }
            Text("").navigationBarTitle("Manual")
        }

        else if showDetail == true {

            ForEach(0..<ManualArray.count, id: \.self){ index in
                if manualNameShow == ManualArray[index]
                {

```

```

VStack{
    Text("Här finns informationen")
        .frame(maxWidth: .infinity, alignment: .leading)
        .padding()
}

Spacer()

Text("Återgå till Manual")
    .font(.headline).foregroundColor(.white).padding().frame(width: 180,
height: 25).background(.blue).cornerRadius(40).offset(y: -10)
    .onTapGesture{
        showDetail = false
    }

}

.navigationBarTitle(manualNameShow)
.navigationBarBackButtonHidden(true)
.navigationBarHidden(false)

Spacer()
}

}

}

func searchResults(searchText: String) -> [String]
{
    if searchText.isEmpty{

        return ManualArray
    }
    else{
        //Skickar tillbaka en filtrerad lista beroende på vad SearchText innehåller
        return ManualArray.filter({ $0.localizedCaseInsensitiveContains(searchText) })
    }
}

}

struct Manual_Previews: PreviewProvider {
    static var previews: some View {
        Manual()
    }
}

```

```
}
```



```
//  
// Logg.swift  
// Examensarbete  
//  
// Created by Matilda Byström on 2022-04-04.  
//
```

```
import SwiftUI

struct Log: View {

    let userID = FirebaseDealer.shared.auth.currentUser?.uid
    @ObservedObject var vm = ChooseContactChatModel()

    var body: some View {
        Text("")
            .navigationBarTitle("Logg", displayMode: .inline)

        Form{
            NavigationLink(destination: SymptomFollowUp())
            {
                Text("Symtomuppföljning")
            }
            .buttonStyle(PlainButtonStyle())

            NavigationLink(destination: SymptomFollowUpHistory())
            {
                Text("Se gamla symtomuppföljningar")
            }
            .buttonStyle(PlainButtonStyle())

            NavigationLink(destination: Protocol())
            {
                Text("Dialysprotokoll")
            }
            .buttonStyle(PlainButtonStyle())

            NavigationLink(destination: ProtocolHistory())
            {
                Text("Se gamla dialysprotokoll")
            }
            .buttonStyle(PlainButtonStyle())
        }
    }
}
```

```

ForEach(vm.users){user in

    if user.uid == userID{
        if user.Admin == "true"
        {

            NavigationLink(destination: PatientFollowUp())
            {
                Text("Uppföljning av patienter")
            }
            .buttonStyle(PlainButtonStyle())
        }

    }
}

struct Log_Previews: PreviewProvider {
    static var previews: some View {
        Log()
    }
}

// 
// PatientFollowUp.swift
// KEXny
//
// Created by Matilda Byström on 2022-05-09.
//

import SwiftUI
import Firebase
import FirebaseFirestore

struct PatientFollowUp: View {

    @ObservedObject var vm = ChooseContactChatModel()
    @ObservedObject var vp = PatientViewModel()
    @State var choosenIndex = 0
    @State var showNewScreenFollowUp = false
    @State var showNewScreenFollowUp2 = false
    @State var showNewScreenFollowUp3 = false
    @State var showNewScreenWithHistory = false

    @State var ob = ProtocolHistory()

}

```

```

@State var dateNotChoosen = false

var body: some View {
    VStack{
        Text(" ")
            .navigationBarTitle("Välj en patient för uppföljning:", displayMode: .inline)
        ForEach(0..<6){index in
            if vm.users[index].Admin == "false"{
                Button{
                    self.chooseIndex = index
                    self.vp.chooseUID = vm.users[index].uid
                    vp.getOldLogs()
                    vp.getOldSymptomLogs()
                    showNewScreenFollowUp.toggle()

                }
                label :{
                    HStack{
                        Text(vm.users[index].Surname)
                        Text(vm.users[index].LastName)

                    }
                }
            }
        }.padding(8)
            .foregroundColor(.black)
            .fullScreenCover(isPresented: $showNewScreenFollowUp){
                VStack{
                    HStack{
                        Text("Uppföljning för:").bold()
                        Text(vm.users[chooseIndex].Surname)
                        Text(vm.users[chooseIndex].LastName)

                    }
                }
            }
        if dateNotChoosen == false {
            ScrollView{
                Text("\nSymtomuppföljningar:").bold()

                ForEach(0..<vp.arrayOldSymptomLogs.count, id: \.self){index in
                    Button{

                        self.vp.chooseIndexLogs = index
                        self.showNewScreenFollowUp2.toggle()
                        self.showNewScreenFollowUp.toggle()

                    }
                    label :{
                        Text("Datum: \(vp.arrayOldSymptomLogs[index].Datum)")

                    }
                }
            }
        }
    }
}

```

```

        }
        .foregroundColor(.black)
    }

Text("\nDialysprotokoll:").bold()

ForEach(0..<vp.arrayOldLogs.count, id: \.self){index1 in
    Button{
        self.vp.chooseIndex1 = index1
        self.showNewScreenFollowUp3.toggle()
        self.showNewScreenFollowUp.toggle()

    }
    label : {
        Text("Datum: \(vp.arrayOldLogs[index1].Datum)")
    }.foregroundColor(.black)

}

Spacer()
Button{
    self.showNewScreenFollowUp.toggle()

} label: {
    Text("Tillbaka")
}.padding(.horizontal,24)
.padding(.vertical, 10)
.background(.pink)
.cornerRadius(28)
.foregroundColor(.white)

}
.padding(20)

}
.fullScreenCover(isPresented: $showNewScreenFollowUp2){

ScrollView{

VStack(){
    Text("\nDatum:
\(vp.arrayOldSymptomLogs[vp.chooseIndexLogs].Datum)")

    Text("Trötthet:
\(vp.arrayOldSymptomLogs[vp.chooseIndexLogs].lackOfEnergy)")

    Text("Dålig rörlighet:
\(vp.arrayOldSymptomLogs[vp.chooseIndexLogs].poorMobility)")

}

```

```

        Text("Sömnsvårigheter:
\(vp.arrayOldSymptomLogs[vp.chooseIndexLogs].TroubleSleeping)")
        Text("Smärta:
\(vp.arrayOldSymptomLogs[vp.chooseIndexLogs].Pain)")
        Text("Klåda:
\(vp.arrayOldSymptomLogs[vp.chooseIndexLogs].Pruritus)")
        Text("Förstopning:
\(vp.arrayOldSymptomLogs[vp.chooseIndexLogs].Constipation)")
        Text("Dåsighet:
\(vp.arrayOldSymptomLogs[vp.chooseIndexLogs].Drowsiness)")
        Text("Andnöd:
\(vp.arrayOldSymptomLogs[vp.chooseIndexLogs].ShortnessOfBreath)")
        Text("Deprimerad:
\(vp.arrayOldSymptomLogs[vp.chooseIndexLogs].Depressed)")
    }
    VStack(){
        Text("Dålig appet:
\(vp.arrayOldSymptomLogs[vp.chooseIndexLogs].PoorApetite")
        Text("Rastlösa ben:
\(vp.arrayOldSymptomLogs[vp.chooseIndexLogs].RestlessLegs")
        Text("Ångest:
\(vp.arrayOldSymptomLogs[vp.chooseIndexLogs].Anxious")
        Text("Hudproblem:
\(vp.arrayOldSymptomLogs[vp.chooseIndexLogs].SkinProblems")
        Text("Illamående:
\(vp.arrayOldSymptomLogs[vp.chooseIndexLogs].Nausea")
        Text("Munproblem:
\(vp.arrayOldSymptomLogs[vp.chooseIndexLogs].MouthProblems")
        Text("Diarre:
\(vp.arrayOldSymptomLogs[vp.chooseIndexLogs].Diarrhoea")
        Text("Kräkningar:
\(vp.arrayOldSymptomLogs[vp.chooseIndexLogs].Krakning)")

    }

    Spacer()

    Button{
        self.showNewScreenFollowUp2.toggle()
        self.showNewScreenFollowUp.toggle()

    } label: {
        Text("Tillbaka")
    }.padding(.horizontal,24)
    .padding(.vertical, 10)
    .background(.pink)
    .cornerRadius(28)
    .foregroundColor(.white)

```

```

}.padding()

}

.fullScreenCover(isPresented: $showNewScreenFollowUp3){
    VStack{
        VStack(){
            Text("Datum: \$(vp.arrayOldLogs[vp.chooseIndex1].Datum)")
            Text("Apparat nr:
\$(vp.arrayOldLogs[vp.chooseIndex1].ApparatNr)")
            Text("Lot koncentrat:
\$(vp.arrayOldLogs[vp.chooseIndex1].LotKoncentrat)")
            Text("Lot bikarbonat:
\$(vp.arrayOldLogs[vp.chooseIndex1].LotBikarbonat)")
            Text("Lot dialysfilter:
\$(vp.arrayOldLogs[vp.chooseIndex1].LotDialysfilter)")
            Text("Lot slangar:
\$(vp.arrayOldLogs[vp.chooseIndex1].LotSlangar)")
            Text("Blodtryck före:
\$(vp.arrayOldLogs[vp.chooseIndex1].BlodTryckFore)")
            Text("Puls före: \$(vp.arrayOldLogs[vp.chooseIndex1].PulsFore)")
            Text("Vikt före: \$(vp.arrayOldLogs[vp.chooseIndex1].ViktFore)")
            Text("Planerad vikt efter:
\$(vp.arrayOldLogs[vp.chooseIndex1].PlaneradViktEfter) ")
        }
        VStack(){
            Text("UF volym L:
\$(vp.arrayOldLogs[vp.chooseIndex1].UFRateLtim)")
            Text("Blodflöde QB:
\$(vp.arrayOldLogs[vp.chooseIndex1].BlodflodeQB)")
            Text("Artärtryck:
\$(vp.arrayOldLogs[vp.chooseIndex1].Artartryck)")
            Text("Ventryck: \$(vp.arrayOldLogs[vp.chooseIndex1].Ventryck)")
            Text("UF Rate L/tim:
\$(vp.arrayOldLogs[vp.chooseIndex1].UFRateLtim)")
            Text("Konduktivitet:
\$(vp.arrayOldLogs[vp.chooseIndex1].Konduktivitet)")
            Text("Effektiv dialystid:
\$(vp.arrayOldLogs[vp.chooseIndex1].EffektivDialystid)")
            Text("Ack QB L: \$(vp.arrayOldLogs[vp.chooseIndex1].AckQBL)")
            Text("Ack UFV L:
\$(vp.arrayOldLogs[vp.chooseIndex1].AckUFVL)")
            Text("Vikt efter:
\$(vp.arrayOldLogs[vp.chooseIndex1].ViktAfter) ")
        }
        VStack(){
            Text("Blodtryck efter:
\$(vp.arrayOldLogs[vp.chooseIndex1].BlodtryckAfter)")
            Text("Puls efter: \$(vp.arrayOldLogs[vp.chooseIndex1].PulsAfter)")
            Text("Kommentarer:
\$(vp.arrayOldLogs[vp.chooseIndex1].Kommentarer) ")
        }
    }
}

```

```

        Button{
            self.showNewScreenFollowUp.toggle()
            self.showNewScreenFollowUp3.toggle()

            } label: {
                Text("Tillbaka")
            }.padding(.horizontal,24)
            .padding(.vertical, 10)
            .background(.pink)
            .cornerRadius(28)
            .foregroundColor(.white)

        }

    }
}

}

```

```

class PatientViewModel: ObservableObject{
    @Published var arrayOldLogs = [oldLog]()
    @Published var chosenUID = ""
    @Published var arrayOldSymptomLogs = [oldSymptomLog]()
    @Published var chosenIndex1 = 0
    @Published var chosenIndexLogs = 0

    func getOldLogs(){
        arrayOldLogs.removeAll()

        let db = Firestore.firestore()
        db.collection("log")
            .document(chosenUID)
            .collection("samling")
            .order(by: "timestamp")
            .getDocuments{ snapshot, error in

            if error == nil {
                snapshot?.documents.forEach({ snap in

```

```

        let data = snap.data()
        self.arrayOldLogs.append(.init(data: data))
    })
}

}

func getOldSymptomLogs(){
    arrayOldSymptomLogs.removeAll()
    FirebaseDealer.shared.firestore
        .collection("symptom")
        .document(chooseenUID)
        .collection("samling")
        .order(by: "timestamp")
        .addSnapshotListener {
            querySnapshot, error in
            if let error = error {
                print("Error: \(error)")
                return
            }
            querySnapshot?.documentChanges.forEach({ change in
                if change.type == .added {
                    let data = change.document.data()
                    self.arrayOldSymptomLogs.append(.init(data: data)) }
            })
        }
}

struct PatientFollowUp_Previews: PreviewProvider {
    static var previews: some View {
        PatientFollowUp()
    }
}

// 
// SymptomFollowUp.swift
// KEXny
//
// Created by Matilda Byström on 2022-05-06.
//

import SwiftUI

```

```

import Firebase

struct SymptomFollowUp: View {

    @State var Datum = ""
    @State var lackOfEnergy = ""
    @State var poorMobility = ""
    @State var TroubleSleeping = ""
    @State var Pain = ""
    @State var Pruritus = ""
    @State var Constipation = ""
    @State var Drowsiness = ""
    @State var ShortnessOfBreath = ""
    @State var Depressed = ""
    @State var PoorApetite = ""
    @State var RestlessLegs = ""
    @State var Anxious = ""
    @State var SkinProblems = ""
    @State var Nausea = ""
    @State var MouthProblems = ""
    @State var Diarrhoea = ""
    @State var Krakning = ""
    @State var symptomsIsSaved = false
    @State var allFieldsNotFilled = false

    var body: some View {

        ScrollView{

            Text("") .navigationBarTitle("Symtomuppföljning", displayMode: .inline)

            VStack{
                if symptomsIsSaved == true{
                    Text("Symtomen har sparats")
                        .foregroundColor(.green)
                }
                else if allFieldsNotFilled{
                    Text("Alla fält är ej ifyllda. Inget har sparats.")
                        .foregroundColor(.red)
                }
            }

            Text("Gradera symtomen på en skala 1-5. \n1: Inget symptom. \n5: Mycket symptom. \nTryck sedan på spara.")
                .padding(.horizontal,24)
                .padding(.vertical, 10)
                .background(Color("Pink"))
                .cornerRadius(20)
                .foregroundColor(.black)
        }
    }
}

```

```

HStack{
    Text("Datum:")
    TextField("", text: $Datum)
        .keyboardType(.decimalPad)
}

HStack{
    Text("Trötthet:")
    TextField("", text: $lackOfEnergy)
        .keyboardType(.decimalPad)
}
HStack{
    Text("Dålig rörlighet:")
    TextField("", text: $poorMobility)
        .keyboardType(.decimalPad)
}
HStack{
    Text("Sömnsvårigheter:")
    TextField("", text: $TroubleSleeping)
        .keyboardType(.decimalPad)
}
HStack{
    Text("Smärta:")
    TextField("", text: $Pain)
        .keyboardType(.decimalPad)
}
HStack{
    Text("Klåda:")
    TextField("", text: $Pruritus)
        .keyboardType(.decimalPad)
}
HStack{
    Text("Förstopnning:")
    TextField("", text: $Constipation)
        .keyboardType(.decimalPad)
}
HStack{
    Text("Dåsighet:")
    TextField("", text: $Drowsiness)
        .keyboardType(.decimalPad)
}

VStack{

HStack{
    Text("Andnöd:")
    TextField("", text: $ShortnessOfBreath)
}

```

```

        .keyboardType(.decimalPad)
    }

HStack{
    Text("Deprimerad:")
    TextField("", text: $Depressed)
        .keyboardType(.decimalPad)
}
HStack{
    Text("Dålig aptit:")
    TextField("", text: $PoorApetite)
        .keyboardType(.decimalPad)
}
HStack{
    Text("Rastlösa ben:")
    TextField("", text: $RestlessLegs)
        .keyboardType(.decimalPad)
}
HStack{
    Text("Ångest:")
    TextField("", text: $Anxious)
        .keyboardType(.decimalPad)
}
HStack{
    Text("Hudproblem:")
    TextField("", text: $SkinProblems)
        .keyboardType(.decimalPad)
}
HStack{
    Text("Illamående:")
    TextField("", text: $Nausea)
        .keyboardType(.decimalPad)
}
HStack{
    Text("Munproblem:")
    TextField("", text: $MouthProblems)
        .keyboardType(.decimalPad)
}
HStack{
    Text("Diarre:")
    TextField("", text: $Diarrhoea)
        .keyboardType(.decimalPad)
}
HStack{
    Text("Kräkningar:")
    TextField("", text: $Krakning)
        .keyboardType(.decimalPad)
}

```

```

    }

    VStack{
        Button{

            guard let userID = FirebaseDealer.shared.auth.currentUser?.uid else{return}

            let symptomData = ["Datum": self.Datum, "lackOfEnergy": self.lackOfEnergy,
"poorMobility": self.poorMobility, "TroubleSleeping": self.TroubleSleeping, "Pain":
self.Pain, "Pruritus": self.Pruritus, "Constipation": self.Constipation, "Drowsiness":
self.Drowsiness, "ShortnessOfBreath": self.ShortnessOfBreath, "Depressed": self.Depressed,
"PoorApetite": self.PoorApetite, "RestlessLegs": self.RestlessLegs, "Anxious": self.Anxious,
"SkinProblems": self.SkinProblems, "Nausea": self.Nausea, "MouthProblems":
self.MouthProblems, "Diarrhoea": self.Diarrhoea, "Krakning": self.Krakning,"timestamp":
Timestamp() as [String : Any]

            let logDataDoc = FirebaseDealer.shared.firestore.collection("symptom")
                .document(userID)
                .collection("samling")
                .document()

            logDataDoc.setData(symptomData)

            self.symptomsIsSaved = true

        }label: {
            Text("Spara loggen")
        }.padding(.horizontal,24)
        .padding(.vertical, 10)
        .background(.pink)
        .cornerRadius(28)
        .foregroundColor(.white)

    }
}.padding(20)
}
}

struct SymptomFollowUp_Previews: PreviewProvider {
    static var previews: some View {
        SymptomFollowUp()
    }
}

```

```

//  

// SymptomFollowUpHistory.swift  

// KEXny  

//  

// Created by Matilda Byström on 2022-05-06.  

//  

import SwiftUI  

import Firebase  

class oldSymptomViewModel: ObservableObject{  

    @Published var arrayOldSymptomLogs = [oldSymptomLog]()  

    init(){  

        getOldSymptomLogs()  

    }  

    func getOldSymptomLogs(){  

        guard let userID = FirebaseDealer.shared.auth.currentUser?.uid else{ return }  

        FirebaseDealer.shared.firestore  

            .collection("symptom")  

            .document(userID)  

            .collection("samling")  

            .order(by: "timestamp")  

            .addSnapshotListener {  

                querySnapshot, error in  

                if let error = error {  

                    print("Error: \(error)")  

                    return  

                }  

                querySnapshot?.documentChanges.forEach({ change in  

                    if change.type == .added {  

                        let data = change.document.data()  

                        self.arrayOldSymptomLogs.append(.init(data: data)) } })  

            }  

        }  

    }  

//Ger en logg för ett datum  

struct oldSymptomLog: Identifiable{  

    var id: String {uid}  

    let Datum, lackOfEnergy, poorMobility, TroubleSleeping, Pain, Pruritus, Constipation,  

    Drowsiness, ShortnessOfBreath, Depressed, PoorApetite, RestlessLegs, Anxious,  

    SkinProblems, Nausea, MouthProblems, Diarrhoea, Krakning, uid: String
}

```

```

init(data: [String: Any]){
    self.Datum = data["Datum"] as? String ?? ""
    self.lackOfEnergy = data["lackOfEnergy"] as? String ?? ""
    self.poorMobility = data["poorMobility"] as? String ?? ""
    self.TroubleSleeping = data["TroubleSleeping"] as? String ?? ""
    self.Pain = data["Pain"] as? String ?? ""
    self.Pruritus = data["Pruritus"] as? String ?? ""
    self.Constipation = data["Constipation"] as? String ?? ""
    self.Drowsiness = data["Drowsiness"] as? String ?? ""
    self.ShortnessOfBreath = data["ShortnessOfBreath"] as? String ?? ""
    self.Depressed = data["Depressed"] as? String ?? ""
    self.PoorApetite = data["PoorApetite"] as? String ?? ""
    self.RestlessLegs = data["RestlessLegs"] as? String ?? ""
    self.Anxious = data["Anxious"] as? String ?? ""
    self.SkinProblems = data["SkinProblems"] as? String ?? ""
    self.Nausea = data["Nausea"] as? String ?? ""
    self.MouthProblems = data["MouthProblems"] as? String ?? ""
    self.Diarrhoea = data["Diarrhoea"] as? String ?? ""
    self.Krakning = data["Krakning"] as? String ?? ""
    self.uid = data["uid"] as? String ?? ""
}
}

```

```

struct SymptomFollowUpHistory: View {

    @ObservedObject var object = oldSymptomViewModel()
    @State var showNewScreenWithLog = false
    @State var choosenIndex = 0

    var body: some View {
        VStack(alignment: .leading){
            ForEach(0..\.self) { index in
                Button{
                    showNewScreenWithLog.toggle()
                    self.choosenIndex = index
                }
                label :{
                    Text("Datum: \((object.arrayOldSymptomLogs[index].Datum)"))
                }
                .foregroundColor(.black)
            }
        }
    }
}

```

```

.fullScreenCover(isPresented: $showNewScreenWithLog)
{
    Spacer()

    VStack(){
        Text("Datum: \$(object.arrayOldSymptomLogs[chooseenIndex].Datum)")
        Text("Trötthet:
\$(object.arrayOldSymptomLogs[chooseenIndex].lackOfEnergy)")
        Text("Dålig rörlighet:
\$(object.arrayOldSymptomLogs[chooseenIndex].poorMobility)")
        Text("Sömnsvårigheter:
\$(object.arrayOldSymptomLogs[chooseenIndex].TroubleSleeping)")
        Text("Smärta: \$(object.arrayOldSymptomLogs[chooseenIndex].Pain)")
        Text("Klåda: \$(object.arrayOldSymptomLogs[chooseenIndex].Pruritus)")
        Text("Förstopning:
\$(object.arrayOldSymptomLogs[chooseenIndex].Constipation)")
        Text("Dåsigheit:
\$(object.arrayOldSymptomLogs[chooseenIndex].Drowsiness)")
        Text("Andnöd:
\$(object.arrayOldSymptomLogs[chooseenIndex].ShortnessOfBreath)")
        Text("Deprimerad:
\$(object.arrayOldSymptomLogs[chooseenIndex].Depressed)")

    }
    VStack(){
        Text("Dålig aptit:
\$(object.arrayOldSymptomLogs[chooseenIndex].PoorApetite)")
        Text("Rastlösa ben:
\$(object.arrayOldSymptomLogs[chooseenIndex].RestlessLegs)")
        Text("Ångest: \$(object.arrayOldSymptomLogs[chooseenIndex].Anxious)")
        Text("Hudproblem:
\$(object.arrayOldSymptomLogs[chooseenIndex].SkinProblems)")
        Text("Illamående: \$(object.arrayOldSymptomLogs[chooseenIndex].Nausea)")
        Text("Munproblem:
\$(object.arrayOldSymptomLogs[chooseenIndex].MouthProblems)")
        Text("Diarre: \$(object.arrayOldSymptomLogs[chooseenIndex].Diarrhoea)")
        Text("Kräkningar:
\$(object.arrayOldSymptomLogs[chooseenIndex].Krakning)")
    }
}

```

```

Spacer()
Spacer()
Button{

    self.showNewScreenWithLog = false

} label: {

```

```

        Text("Tillbaka")
    }.padding(.horizontal,24)
    .padding(.vertical, 10)
    .background(.pink)
    .cornerRadius(28)
    .foregroundColor(.white)

    Spacer()
}

}
.navigationBarTitle("Symtomhistorik", displayMode: .inline)
Spacer()
}

```

```

struct SymptomFollowUpHistory_Previews: PreviewProvider {
    static var previews: some View {
        SymptomFollowUpHistory()
    }
}

```

```

//  

// Chat.swift  

// Examensarbete  

//  

// Created by Matilda Byström on 2022-04-04.  

//Chatten har byggt upp av kod från "lets build that app", viss modifiering har gjorts för att  

anpassa till vårt syfte.  

//“Lets Build That App.”  

https://www.letsbuildthatapp.com/course/SwiftUI%20Firebase%20Real%20Time%20Chat  

(accessed May 20, 2022).

```

```

import SwiftUI
import Firebase

struct RecentMessage: Identifiable{
    var id: String {documentID}
    let documentID: String
    let text, fromUserID, toUserID, Admin, Email, LastName, Surname, uid: String
    let timestamp: Timestamp

```

```

init(documentID: String, data: [String: Any]){
    self.documentID = documentID
    self.text = data["text"] as? String ?? ""
    self.fromUserID = data["fromUserID"] as? String ?? ""
    self.toUserID = data["toUserID"] as? String ?? ""
    self.timestamp = data["timestamp"] as? Timestamp ?? Timestamp(date: Date())
    self.Admin = data["Admin"] as? String ?? ""
    self.Email = data["Email"] as? String ?? ""
    self.LastName = data["LastName"] as? String ?? ""
    self.Surname = data["Surname"] as? String ?? ""
    self.uid = data["uid"] as? String ?? ""
}
}

```

```

class ChatModel: ObservableObject{

    @Published var errorMessage = ""
    @Published var chatUser: ChatUser?
    @Published var recentMessages = [RecentMessage]()

    init(){
        getLoggedInUser()
        fetchRecentMessages()
    }
}

```

```

private func fetchRecentMessages(){

    guard let uid = FirebaseDealer.shared.auth.currentUser?.uid
    else{return}
}

```

```

FirebaseDealer.shared.firestore.collection("recentMessages").document(uid).collection("chat
Messages").order(by: "timestamp")

```

```

.addSnapshotListener{ querySnapshot, error in
    if (error != nil){
        print("Failed to listen to recent messages")
        return
    }
    querySnapshot?.documentChanges.forEach({ change in
        let docId = change.document.documentID
        if let index = self.recentMessages.firstIndex(where: { rm in
            return rm.documentID == docId
        }) {
            self.recentMessages.remove(at: index)
        }
    })
}

```

```

self.recentMessages.insert(.init(documentID: docId, data:
change.document.data()), at: 0)

```

```

        })
    }
}

private func getLoggedInUser(){

    guard let uid = FirebaseDealer.shared.auth.currentUser?.uid
    else
    {
        self.errorMessage = "Hittade inte firebase uid"
        return
    }
}

```

FirebaseDealer.shared.firestore.collection("users").document(uid).getDocument{snapshot,
error in

```

if let error = error{
    self.errorMessage = "Misslyckade att hämta nuvarande användare: \(error)"
    return
}

guard let data = snapshot?.data() else{
    self.errorMessage = "No data found"
    return
}

self.chatUser = ChatUser(data: data)

}
}

}

```

```

struct Chat: View {

    @ObservedObject private var vm = ChatModel()
    @State var goToChat = false
    @State var showContactsForNewMessage = false
    @State var chatUser: ChatUser?
    @ObservedObject private var vt = ChooseContactChatModel ()
    private var chatLogViewModel = ChatLogViewModel(chatUser: nil)

    var body: some View {

```

```

VStack{
    HStack{
        Text("Inloggad som: \(vm.chatUser?.Surname ?? "") \(vm.chatUser?.LastName ?? "")")
            .font(.headline)
    }
    messageView
}

NavLink("", isActive: $goToChat) {
    ChatLogView(vm: chatLogViewModel)
        .foregroundColor(.black)
}

Button{
    showContactsForNewMessage.toggle()

    }label: {
        Text("Nytt meddelande")
    }.padding(.horizontal)
        .padding(.vertical)
        .background(.pink)
        .cornerRadius(28)
        .foregroundColor(.white)

}
.fullScreenCover(isPresented: $showContactsForNewMessage)
{
    ChooseContactChat(userSelected: { user in
        self.goToChat.toggle()
        self.chatLogViewModel.chatUser = user
        self.chatLogViewModel.getAllChatMessages()
        self.chatUser = user
    })
}

private var messageView: some View{
    ScrollView{
        ForEach(vm.recentMessages) { num in
            VStack(alignment: .leading){
                Button{

```

```

let uid = FirebaseDealer.shared.auth.currentUser?.uid == num.fromUserID ?
num.toUserID : num.fromUserID

self.chatUser = .init(data: [FirebaseConstants.Email: num.Email,
FirebaseConstants.Surname: num.Surname, FirebaseConstants.uid: uid,
FirebaseConstants.LastName: num.LastName, FirebaseConstants.Admin: num.Admin])

self.chatLogViewModel.chatUser = self.chatUser
self.chatLogViewModel.getAllChatMessages()
self.goToChat.toggle()

}

label :{
    VStack(alignment: .leading, spacing: 8){
        HStack{
            Text((num.Surname) + " " + (num.LastName)).font(.system(size: 16,
weight: .bold)).foregroundColor(Color(.label))
            Spacer()
        }
    }
    Text(num.text)
        .foregroundColor(Color("Gray")).multilineTextAlignment(.leading)
    }
}.padding(8)

Divider()
}

.navigationBarHidden(false)
}

struct ChatUser: Identifiable{

var id: String{ uid }
let Admin, Email, LastName, Surname, uid: String

init(data: [String: Any]){

```

```

self.Admin = data["Admin"] as? String ?? ""
self.Email = data["Email"] as? String ?? ""
self.LastName = data["LastName"] as? String ?? ""
self.Surname = data["Surname"] as? String ?? ""
self.uid = data["uid"] as? String ?? ""

}

}

struct Chat_Previews: PreviewProvider {
    static var previews: some View {
        Chat()
    }
}

//  

// ChatLogView.swift  

// KEXny  

//  

// Created by Sandra Rodlund on 2022-05-03.  

//Chatten har byggt upp av kod från "lets build that app", viss modifiering har gjorts för att  

anpassa till vårt syfte.  

//“Lets Build That App.”  

https://www.letsbuildthatapp.com/course/SwiftUI%20Firebase%20Real%20Time%20Chat  

(accessed May 20, 2022).

import SwiftUI
import Firebase

class ChatLogViewModel: ObservableObject{
    @Published var count = 0
    @Published var chatText = ""
    var chatUser: ChatUser?
    @Published var arrayChatMessages = [message]()

    init(chatUser: ChatUser?){
        self.chatUser = chatUser
        getAllChatMessages()
    }

    public func getAllChatMessages(){
        arrayChatMessages.removeAll()

        guard let fromUserID = FirebaseDealer.shared.auth.currentUser?.uid else{ return }
    }
}

```

```

guard let toUserID = chatUser?.uid else { return }

FirebaseDealer.shared.firestore
    .collection("chatMessages")
    .document(fromUserID)
    .collection(toUserID)
    .order(by: "timestamp")
    .addSnapshotListener { [self]
        querySnapshot, error in
        if let error = error {
            print("Error: \(error)")
            return
        }

        querySnapshot?.documentChanges.forEach({ change in
            if change.type == .added {
                let data = change.document.data()

                if self.arrayChatMessages.count == 0{
                    self.arrayChatMessages.append(.init(documentID:
change.document.documentID, data: data))

                }

                if self.arrayChatMessages.count != 0 && change.document.documentID != self.arrayChatMessages[self.arrayChatMessages.count-1].documentID{
                    self.arrayChatMessages.append(.init(documentID:
change.document.documentID, data: data))
                }
            }
        })
    }
}

func handleSendMessage(){

guard let fromUserID = FirebaseDealer.shared.auth.currentUser?.uid
else{ return }

```

```

guard let toUserID = chatUser?.uid else { return }

let document = FirebaseDealer.shared.firestore.collection("chatMessages")
    .document(fromUserID)
    .collection(toUserID)
    .document()

let chatMessageData = [FirebaseConstants.fromUserID: fromUserID,
    FirebaseConstants.toUserID: toUserID, FirebaseConstants.text: self.chatText, "timestamp": Timestamp() as [String : Any]

document.setData(chatMessageData) { error in
    if let error = error {
        print("Gick ej att spara meddelande: \(error)")
        return
    }

    print("Successfully saved current user sending message")
    self.persistRecentMessage()
    self.chatText = ""
    self.count += 1
}

let recipientChatDoc = FirebaseDealer.shared.firestore.collection("chatMessages")
    .document(toUserID)
    .collection(fromUserID)
    .document()

recipientChatDoc.setData(chatMessageData) { error in
    if let error = error {
        print("Gick ej att spara meddelande: \(error)")
        return
    }

    print("Recipient saved message as well")
}

private func persistRecentMessage(){
    guard let chatUser = chatUser else{ return }
    guard let fromUserID = FirebaseDealer.shared.auth.currentUser?.uid else{
        return }
    guard let toUserID = self.chatUser?.uid else{ return }

    let document = FirebaseDealer.shared.firestore
        .collection("recentMessages")

```

```

.document(fromUserID)
.collection("chatMessages")
.document(toUserID)

let data = ["timestamp": Timestamp(), FirebaseConstants.text: self.chatText,
FirebaseConstants.fromUserID: fromUserID, FirebaseConstants.toUserID: toUserID,
FirebaseConstants.Email: chatUser.Email, FirebaseConstants.Surname: chatUser.Surname,
FirebaseConstants.LastName: chatUser.LastName, FirebaseConstants.uid: chatUser.uid] as
[String : Any]

```

```

document.setData(data){error in
if let error = error{
    print(error)
    return
}
}
}

```

```

struct ChatLogView: View {

@ObservedObject var vm: ChatLogViewModel

static let emptyScollToString = "Empty"

var body: some View {
ZStack{
VStack{
ScrollView{
ScrollViewReader{ ScrollViewProxy in
VStack{

ForEach(vm.arrayChatMessages){ messages in
VStack{
if messages.fromUserID ==
FirebaseDealer.shared.auth.currentUser?.uid{
HStack{
Spacer()
HStack{
Text(messages.text)
.foregroundColor(.black)

}
.padding()

```

```

        .background(Color.green)
        .cornerRadius(20)
    }

} else{

HStack{
    HStack{

        Text(messages.text)
        .foregroundColor(.black)

    }
    .padding()
    .background(Color.white)
    .cornerRadius(20)
    Spacer()
}
}

}.padding(.horizontal)
.padding(.top, 7)

}

HStack{
    Spacer()
}.id(Self.emptyScollToString)

}.onReceive(vm.$count){ _ in
    withAnimation(.easeOut(duration: 0.5)){
        ScrollViewProxy.scrollTo(Self.emptyScollToString, anchor: .bottom)
    }
}

}

.navigationTitle(vm.chatUser?.Surname ?? "").background(Color(.init(white: 0.95,
alpha: 1))).navigationBarTitleDisplayMode(.inline)
.safeAreaInset(edge: .bottom){
    chatBottomBar.background(Color(.systemBackground)).ignoresSafeArea()
}

}

}

```

```

private var chatBottomBar: some View{
    HStack {
        Spacer()
        TextField("Skriv ditt meddelande här...", text: $vm.chatText)

        Button{
            vm.handleSentMessage()

            } label: {
                Text("Skicka")
                .foregroundColor(.black)
                .padding(8)
            } .padding(.horizontal)
            .padding(.vertical, 8)
            .background(Color.green)
            .cornerRadius(20)
        }
        .padding(.horizontal)
        .padding(.vertical, 8)
        .background(.white)
    }
}

```

```

struct FirebaseConstants{
    static let fromUserID = "fromUserID"
    static let toUserID = "toUserID"
    static let text = "text"
    static let Admin = "Admin"
    static let Surname = "Surname"
    static let LastName = "LastName"
    static let Email = "Email"
    static let uid = "uid"
}

```

```

struct message: Identifiable{

    var id: String { documentID }

    let documentID: String
    let fromUserID, toUserID, text, Email: String
    let Surname, LastName, Admin, uid: String

    init(documentID: String, data: [String: Any]){
        self.documentID = documentID
        self.fromUserID = data[FirebaseConstants.fromUserID] as? String ?? ""
        self.toUserID = data[FirebaseConstants.toUserID] as? String ?? ""
        self.text = data[FirebaseConstants.text] as? String ?? ""
        self.LastName = data[FirebaseConstants.LastName] as? String ?? ""
    }
}

```

```

self.Surname = data[FirebaseConstants.Surname] as? String ?? ""
self.Email = data[FirebaseConstants.Email] as? String ?? ""
self.Admin = data[FirebaseConstants.Admin] as? String ?? ""
self.uid = data[FirebaseConstants.uid] as? String ?? ""
}

}

struct ChatLogView_Previews: PreviewProvider {
    static var previews: some View {
        NavigationView{
            ChatLogView(vm: .init(chatUser: nil))
        }
    }
}

// FirebaseDealer.swift
// KEXny
//
// Created by Sandra Rodlund on 2022-05-03.
//Chatten har byggt upp av kod från "lets build that app", viss modifiering har gjorts för att
//anpassa till vårt syfte.
//“Lets Build That App.”
https://www.letsbuildthatapp.com/course/SwiftUI%20Firebase%20Real%20Time%20Chat
(accessed May 20, 2022).
import Foundation
import Firebase
class FirebaseDealer: NSObject{

    let auth: Auth
    let firestore: Firestore

    static let shared = FirebaseDealer()

    override init() {
        FirebaseApp.configure()
        self.auth = Auth.auth()
        self.firestore = Firestore.firestore()

        super.init()
    }

}

//
// Instruktioner.swift
// Examensarbete
//
// Created by Matilda Byström on 2022-04-04.
//

```

```

import SwiftUI

struct Instructions: View {

    var body: some View {
        Text("")
            .navigationBarTitle("Instruktioner", displayMode: .inline)

        Form{
            NavigationLink(destination: Checklist())
            {
                Text("Checklista")
            }
            .buttonStyle(PlainButtonStyle())
        }

        NavigationLink(destination:Manual())
        {
            Text("Manual")
        }
        .buttonStyle(PlainButtonStyle())
    }
}

struct Instructions_Previews: PreviewProvider {
    static var previews: some View {
        Instructions()
    }
}

// Checklista.swift
// Examensarbete
//
// Created by Matilda Byström on 2022-04-04.
//

```

```

import SwiftUI

struct Checklist: View {

    @State private var showDetails1 = false
    @State private var showDetails2 = false
    @State private var showDetails3 = false

```

```

@State var checkState: Bool = false

struct checklistItem {
    var item : String
    var isChecked: Bool

    var image: Image
    {
        if isChecked
        {
            return Image(systemName: "checkmark.square")
        }
        else
        {
            return Image(systemName: "square")
        }
    }

    init(toDo: String, isChecked: Bool)
    {
        self.item = ToDo
        self.isChecked = isChecked
    }
}

@State var manuellPriming = [checklistItem(toDo: "Klä maskinen och placera filtret med röd slang nedåt.", isChecked: false), checklistItem(toDo: "När slingan är grön är maskinen redo för priming.", isChecked: false), checklistItem(toDo: "Flytta vätskeslangarna till filtret, blått mot blått, rött mot rött.", isChecked: false)]]

@State var PrimingAK98 = [checklistItem(toDo: "Sätt på maskinen och låt maskinen starta och ”banka” klart", isChecked: false), checklistItem(toDo: "Sätt i BiCart", isChecked: false), checklistItem(toDo: "Koppla till koncentratet", isChecked: false)]]

@State var AvslutingAK98 = [checklistItem(toDo: "Behandlingstiden är slut-tryck bekräfta (Uppmärksamhetslarm - blå lampa blinkar).", isChecked: false), checklistItem(toDo: "Blodretur - tryck och sedan bekräfta.", isChecked: false), checklistItem(toDo: "Bekräfta - pumpen stannar.", isChecked: false)]]

@State var tapCount = false

var body: some View {

    Text("")
        .navigationBarTitle("Checklistor", displayMode: .inline)

    Form{

```

```

Button("Manuell priming") {
    showDetails1.toggle()

}.buttonStyle(PlainButtonStyle())
if showDetails1{
    VStack(alignment: .leading){
        ForEach(0..<3){index in
            HStack(alignment: .top, spacing: 10){
                (Text(manuellPriming[index].image) + Text(manuellPriming[index].item))
                    .fixedSize(horizontal: false, vertical: true).onTapGesture {
                        self.manuellPriming[index].isChecked =
                            !self.manuellPriming[index].isChecked
                }
            }
        }
    }
}

Button("Priming AK98") {
    showDetails2.toggle()

}.buttonStyle(PlainButtonStyle())
if showDetails2{
    VStack(alignment: .leading){
        ForEach(0..<3){index in
            HStack(alignment: .top, spacing: 10){
                (Text(PrimingAK98[index].image) + Text(PrimingAK98[index].item))
                    .fixedSize(horizontal: false, vertical: true).onTapGesture {
                        self.PrimingAK98[index].isChecked =
                            !self.PrimingAK98[index].isChecked
                }
            }
        }
    }
}

Button("Avslutning AK 98") {
    showDetails3.toggle()
}

```

```

    }.buttonStyle(PlainButtonStyle())
    if showDetails3{
        VStack(alignment: .leading){
            ForEach(0..<3){index in
                HStack(alignment: .top, spacing: 10){
                    (Text(AvslutingAK98[index].image) + Text(AvslutingAK98[index].item))
                        .fixedSize(horizontal: false, vertical: true).onTapGesture {
                            self.AvslutingAK98[index].isChecked =
                            !self.AvslutingAK98[index].isChecked
                }
            }
        }
    }
}

```

```

struct Checklist_Previews: PreviewProvider {
    static var previews: some View {
        Checklist()
    }
}

// ChooseContactChat.swift
// KEXny
//
// Created by Matilda Byström on 2022-04-27.
//Chatten har byggt upp av kod från "lets build that app", viss modifiering har gjorts för att
//anpassa till vårt syfte.
//“Lets Build That App.”
https://www.letsbuildthatapp.com/course/SwiftUI%20Firebase%20Real%20Time%20Chat
(accessed May 20, 2022).

```

```

import SwiftUI
import FirebaseFirestore
import Firebase

```

```

class ChooseContactChatModel: ObservableObject{
    @Published var users = [ChatUser]()
    @Published var usersCollectionRef: CollectionReference!
    @Published var errorMessage = ""

    init(){
        getAllUsers()
    }

    private func getAllUsers() {

        let db = Firestore.firestore()
        db.collection("users").getDocuments{snapshot, error in

            if error == nil {
                snapshot?.documents.forEach({ snap in
                    let data = snap.data()
                    self.users.append(.init(data: data))

                })
            }
        }

    }
}

struct ChooseContactChat: View {

    @Environment(\.presentationMode) var presentationMode

    @ObservedObject var vm = ChooseContactChatModel()
    let userID = FirebaseDealer.shared.auth.currentUser?.uid

    let userSelected: (ChatUser) -> ()

    var body: some View {

        NavigationView{
            ScrollView{
                Text(vm.errorMessage)

                ForEach(vm.users){user1 in
                    if user1.uid == userID{

```

```

if user1.Admin == "true"
{
    ForEach(vm.users){user in
        if user.uid != userID {
            Button{
                presentationMode.wrappedValue.dismiss()
                userSelected(user)
            }
            label :
            {
                HStack{
                    Text(user.Surname)
                    Text(user.LastName)
                }
            }.padding(8)
            .foregroundColor(.black)
            Divider()
        }
    }
}
else{
    ForEach(vm.users){user in
        if user.uid != userID && user.Admin == "true" {
            Button{
                presentationMode.wrappedValue.dismiss()
                userSelected(user)
            }
            label :
            {
                HStack{
                    Text(user.Surname + " " + user.LastName)
                }
            }.padding(8)
            .foregroundColor(.black)
            Divider()
        }
    }
}
}

}.navigationBarTitle("Välj kontakt", displayMode: .inline)

```

```

.toolbar {
    ToolbarItemGroup(placement: .navigationBarLeading)
    {
        Button{
            presentationMode.wrappedValue.dismiss()

        } label: {
            Text("Avbryt")
        }
    }
}

struct ChooseContactChat_Previews: PreviewProvider {
    static var previews: some View {
        Chat()
    }
}

// Protocol.swift
// KEXny
//
// Created by Sandra Rodlund on 2022-05-03.
//

import SwiftUI
import Firebase

struct Protocol: View {

    @State var Datum = ""
    @State var ApparatNr = ""
    @State var LotKoncentrat = ""
    @State var LotBikarbonat = ""
    @State var LotDialysfilter = ""
    @State var LotSlangar = ""
    @State var BlodTryckFore = ""
    @State var PulsFore = ""
    @State var ViktFore = ""
    @State var PlaneradViktEfter = ""
    @State var UFVolymL = ""
    @State var BlodflodeQB = ""
}

```

```

@State var Artartryck = ""
@State var Ventryck = ""
@State var UFRateLtim = ""
@State var Konduktivitet = ""
@State var EffektivDialystid = ""
@State var AckQBL = ""
@State var AckUFVL = ""
@State var ViktAfter = ""
@State var BlodtryckAfter = ""
@State var PulsAfter = ""
@State var Kommentarer = ""
@State var logIsSaved = false
@State var userID = ""

var body: some View {
    ScrollView{
        VStack(alignment: .leading){
            VStack{
                if logIsSaved == true{
                    Text("Loggen har sparats")
                        .foregroundColor(.green)
                }
            }

            Text("Fyll i loggen och tryck sedan på spara loggen.")
                .padding(.horizontal,24)
                .padding(.vertical, 10)
                .background(Color("Pink"))
                .cornerRadius(20)
                .foregroundColor(.black)

            HStack{
                Text("Datum:")
                TextField("", text: $Datum)
            }

            HStack{
                Text("Apparat nr:")
                TextField("", text: $ApparatNr)
            }

            HStack{
                Text("Lot koncentrat:")
                TextField("", text: $LotKoncentrat)
            }

            HStack{

```

```

        Text("Lot bikarbonat:")
        TextField("", text: $LotBikarbonat)
    }

HStack{
    Text("Lot dialysfilter:")
    TextField("", text: $LotDialysfilter)
}
HStack{
    Text("Lot slangar:")
    TextField("", text: $LotSlangar)
}

HStack{
    Text("Blodtryck före:")
    TextField("", text: $BlodTryckFore)
}

HStack{
    Text("Puls före:")
    TextField("", text: $PulsFore)
}
}

VStack{
    HStack{
        Text("Vikt före:")
        TextField("", text: $ViktFore)
    }

    HStack{
        Text("Planerad vikt efter:")
        TextField("", text: $PlaneradViktEfter)
    }
    HStack{
        Text("UF volym L:")
        TextField("", text: $UfVolymL)
    }

    HStack{
        Text("Blodflöde QB:")
        TextField("", text: $BlodflodeQB)
    }

    HStack{
        Text("Artärtryck:")
        TextField("", text: $Artartryck)
    }
}

```

```

HStack{
    Text("Ventryck:")
    TextField("", text: $Ventryck)
}

HStack{
    Text("UF Rate L/tim:")
    TextField("", text: $UFRateLtim)
}

HStack{
    Text("Konduktivitet:")
    TextField("", text: $Konduktivitet)
}

HStack{
    Text("Effektiv dialystid:")
    TextField("", text: $EffektivDialystid)
}

HStack{
    Text("Ack QB L:")
    TextField("", text: $AckQBL)
}
}

VStack{
    HStack{
        Text("Ack UFV L:")
        TextField("", text: $AckUFVL)
    }

    HStack{
        Text("Vikt efter:")
        TextField("", text: $ViktAfter)
    }

    HStack{
        Text("Blodtryck efter:")
        TextField("", text: $BlodtryckAfter)
    }

    HStack{
        Text("Puls efter:")
        TextField("", text: $PulsAfter)
    }
}

```

```

HStack{
    Text("Kommentarer:")
    TextField("", text: $Kommentarer)
}

Button{
    guard let userID1 = FirebaseDealer.shared.auth.currentUser?.uid else{return}

    self.userID = userID1

    let logData = ["Ack QBL": self.AckQBL,"Ack UFVL": self.AckUFVL,
    "ApparatNr": self.ApparatNr,"Artärtryck": self.Artartryck, "BlodTryckAfter":
    self.BlodtryckAfter, "BlodTryckFore": self.BlodTryckFore,"Blodflöde QB":
    self.BlodflodeQB, "Datum": self.Datum, "Effektiv dialystid": self.EffektivDialystid,
    "Kommentarer": self.Kommentarer, "Konduktivitet": self.Konduktivitet, "LotBikarbonat":
    self.LotBikarbonat, "LotDialysfilter": self.LotDialysfilter, "LotKoncentrat":
    self.LotKoncentrat, "LotSlangar": self.LotSlangar, "Planerad vikt efter":
    self.PlaneradViktEfter, "Puls efter": self.PulsAfter, "PulsFore": self.PulsFore, "UF Rate
L/tim": self.UFRateLtim, "UF Volym L": self.UFVolymL, "Ventryck": self.Ventryck,
"Vikt efter": self.ViktAfter, "ViktFore": self.ViktFore, "uid": self.userID, "timestamp":
Timestamp(), ] as [String : Any]

    let logDataDoc = FirebaseDealer.shared.firestore.collection("log")
        .document(userID)
        .collection("samling")
        .document()

    logDataDoc.setData(logData)

    self.logIsSaved = true

}label: {
    Text("Spara loggen")
}.padding(.horizontal,24)
.padding(.vertical, 10)
.background(.pink)
.cornerRadius(28)
.foregroundColor(.white)

.navigationBarTitle("Dialysprotokoll:", displayMode: .inline)

}
.padding(20)
Spacer()
}

```

```
}
```

```
struct Protocol_Previews: PreviewProvider {
    static var previews: some View {
        Protocol()
    }
}
```

```
//  
// Protocol.swift  
// KEXny  
//  
// Created by Sandra Rodlund on 2022-05-03.  
//
```

```
import SwiftUI  
import Firebase
```

```
struct Protocol: View {
```

```
    @State var Datum = ""  
    @State var ApparatNr = ""  
    @State var LotKoncentrat = ""  
    @State var LotBikarbonat = ""  
    @State var LotDialysfilter = ""  
    @State var LotSlangar = ""  
    @State var BlodTryckFore = ""  
    @State var PulsFore = ""  
    @State var ViktFore = ""  
    @State var PlaneradViktEfter = ""  
    @State var UFVolymL = ""  
    @State var BlodflodeQB = ""  
    @State var Artartryck = ""  
    @State var Ventryck = ""  
    @State var UFRateLtim = ""  
    @State var Konduktivitet = ""  
    @State var EffektivDialystid = ""  
    @State var AckQBL = ""  
    @State var AckUFVL = ""  
    @State var ViktAfter = ""  
    @State var BlodtryckAfter = ""  
    @State var PulsAfter = ""  
    @State var Kommentarer = ""  
    @State var logIsSaved = false  
    @State var userID = ""
```

```

var body: some View {
    ScrollView{
        VStack(alignment: .leading){
            VStack{
                if logIsSaved == true{
                    Text("Loggen har sparats")
                        .foregroundColor(.green)
                }
            }

            Text("Fyll i loggen och tryck sedan på spara loggen.")
                .padding(.horizontal,24)
                .padding(.vertical, 10)
                .background(Color("Pink"))
                .cornerRadius(20)
                .foregroundColor(.black)
        }

        HStack{
            Text("Datum:")
            TextField("", text: $Datum)
        }

        HStack{
            Text("Apparat nr:")
            TextField("", text: $ApparatNr)
        }

        HStack{
            Text("Lot koncentrat:")
            TextField("", text: $LotKoncentrat)
        }

        HStack{
            Text("Lot bikarbonat:")
            TextField("", text: $LotBikarbonat)
        }

        HStack{
            Text("Lot dialysfilter:")
            TextField("", text: $LotDialysfilter)
        }

        HStack{
            Text("Lot slangar:")
            TextField("", text: $LotSlangar)
        }

        HStack{
            Text("Blodtryck före:")
        }
    }
}

```

```

        TextField("", text: $BlodTryckFore)
    }

    HStack{
        Text("Puls före:")
        TextField("", text: $PulsFore)
    }

}

VStack{
    HStack{
        Text("Vikt före:")
        TextField("", text: $ViktFore)
    }

    HStack{
        Text("Planerad vikt efter:")
        TextField("", text: $PlaneradViktEfter)
    }
    HStack{
        Text("UF volym L:")
        TextField("", text: $UfVolymL)
    }

    HStack{
        Text("Blodflöde QB:")
        TextField("", text: $BlodflodeQB)
    }

    HStack{
        Text("Artärtryck:")
        TextField("", text: $Artartryck)
    }

    HStack{
        Text("Ventryck:")
        TextField("", text: $Ventryck)
    }

    HStack{
        Text("UF Rate L/tim:")
        TextField("", text: $UfRateLtim)
    }

    HStack{
        Text("Konduktivitet:")
        TextField("", text: $Konduktivitet)
    }
}

```

```

HStack{
    Text("Effektiv dialystid:")
    TextField("", text: $EffektivDialystid)
}

HStack{
    Text("Ack QB L:")
    TextField("", text: $AckQBL)
}

}

VStack{

    HStack{
        Text("Ack UFV L:")
        TextField("", text: $AckUFVL)
    }

    HStack{
        Text("Vikt efter:")
        TextField("", text: $ViktAfter)
    }

    HStack{
        Text("Blodtryck efter:")
        TextField("", text: $BlodtryckAfter)
    }

    HStack{
        Text("Puls efter:")
        TextField("", text: $PulsAfter)
    }

    HStack{
        Text("Kommentarer:")
        TextField("", text: $Kommentarer)
    }

    Button{

        guard let userID1 = FirebaseDealer.shared.auth.currentUser?.uid else{return}

        self.userID = userID1

        let logData = ["Ack QBL": self.AckQBL, "Ack UFVL": self.AckUFVL,
        "ApparatNr": self.ApparatNr, "Artärtryck": self.Artartryck, "BlodTryckAfter": self.BloodtryckAfter, "BlodTryckFore": self.BloodtryckFore, "Blodflöde QB": self.BloodfloodeQB, "Datum": self.Datum, "Effektiv dialystid": self.EffektivDialystid,
    
```

```

"Kommentarer": self.Kommentarer, "Konduktivitet": self.Konduktivitet, "LotBikarbonat": self.LotBikarbonat, "LotDialysfilter": self.LotDialysfilter, "LotKoncentrat": self.LotKoncentrat, "LotSlangar": self.LotSlangar, "Planerad vikt efter": self.PlaneradViktEfter, "Puls efter": self.PulsAfter, "PulsFore": self.PulsFore, "UF Rate L/tim": self.UFRateLtim, "UF Volym L": self.UFVolymL, "Ventryck": self.Ventryck, "Vikt efter": self.ViktAfter, "ViktFore": self.ViktFore, "uid": self.userID, "timestamp": Timestamp(), ] as [String : Any]

```

```

let logDataDoc = FirebaseDealer.shared.firestore.collection("log")
    .document(userID)
    .collection("samling")
    .document()

logDataDoc.setData(logData)

self.logIsSaved = true

}label: {
    Text("Spara loggen")
}.padding(.horizontal,24)
    .padding(.vertical, 10)
    .background(.pink)
    .cornerRadius(28)
    .foregroundColor(.white)

}.navigationBarTitle("Dialysprotokoll:", displayMode: .inline)

}

}.padding(20)
Spacer()
}

}

```

```

struct Protocol_Previews: PreviewProvider {
    static var previews: some View {
        Protocol()
    }
}

```

```

// ProtocolHistory.swift
// KEXny
//
// Created by Matilda Byström on 2022-05-04.

```

```

//



import SwiftUI
import Firebase

struct oldLog: Identifiable{

    var id: String {uid}

    let Datum, ApparatNr, LotKoncentrat, LotBikarbonat, LotDialysfilter, LotSlangar,
    BlodTryckFore, PulsFore, ViktFore: String
    let PlaneradViktEfter, UFVolymL, BlodflodeQB, Artartryck, Ventryck, UFRateLtim,
    Konduktivitet, EffektivDialystid, AckQBL, AckUFVL, ViktAfter, BlodtryckAfter, PulsAfter,
    Kommentarer, uid: String

    init(data: [String: Any]){
        self.AckQBL = data["AckQBL"] as? String ?? ""
        self.AckUFVL = data["AckUFVL"] as? String ?? ""
        self.ApparatNr = data["ApparatNr"] as? String ?? ""
        self.Artartryck = data["Artartryck"] as? String ?? ""
        self.BlodtryckAfter = data["BlodtryckAfter"] as? String ?? ""
        self.BlodTryckFore = data["BlodTryckFore"] as? String ?? ""
        self.BlodflodeQB = data["BlodflodeQB"] as? String ?? ""
        self.Datum = data["Datum"] as? String ?? ""
        self.EffektivDialystid = data["EffektivDialystid"] as? String ?? ""
        self.Kommentarer = data["Kommentarer"] as? String ?? ""
        self.Konduktivitet = data["Konduktivitet"] as? String ?? ""
        self.LotBikarbonat = data["LotBikarbonat"] as? String ?? ""
        self.LotDialysfilter = data["LotDialysfilter"] as? String ?? ""
        self.LotKoncentrat = data["LotKoncentrat"] as? String ?? ""
        self.LotSlangar = data["LotSlangar"] as? String ?? ""
        self.PlaneradViktEfter = data["PlaneradVikt"] as? String ?? ""
        self.PulsAfter = data["PulsAfter"] as? String ?? ""
        self.PulsFore = data["PulsFore"] as? String ?? ""
        self.UFRateLtim = data["UFRateLtim"] as? String ?? ""
        self.UFVolymL = data["UFVolymL"] as? String ?? ""
        self.Ventryck = data["Ventryck"] as? String ?? ""
        self.ViktAfter = data["ViktAfter"] as? String ?? ""
        self.ViktFore = data["ViktFore"] as? String ?? ""
        self.uid = data["uid"] as? String ?? ""
    }
}

struct ProtocolHistory: View {

    @ObservedObject var object = oldLogViewModel()
    @State var showNewScreenWithLog = false
    @State var choosenIndex = 0
}

```

```

var body: some View {
    VStack(alignment: .leading){
        ForEach(0..<object.arrayOldLogs.count, id: \.self){index in
            Button{
                showNewScreenWithLog.toggle()
                self.chooseIndex = index
            }
            label :
            {
                Text("Datum: \((object.arrayOldLogs[index].Datum)"))
            }
            .foregroundColor(.black)
            .fullScreenCover(isPresented: $showNewScreenWithLog)
            {
                VStack(){
                    Text("Datum: \((object.arrayOldLogs[chooseIndex].Datum)"))
                    Text("Apparat nr: \((object.arrayOldLogs[chooseIndex].ApparatNr)"))
                    Text("Lot koncentrat: \((object.arrayOldLogs[chooseIndex].LotKoncentrat)"))
                    Text("Lot bikarbonat: \((object.arrayOldLogs[chooseIndex].LotBikarbonat)"))
                    Text("Lot dialysfilter:
                    \((object.arrayOldLogs[chooseIndex].LotDialysfilter)"))
                    Text("Lot slangar: \((object.arrayOldLogs[chooseIndex].LotSlangar)"))
                    Text("Blodtryck före:
                    \((object.arrayOldLogs[chooseIndex].BlodTryckFore)"))
                    Text("Puls före: \((object.arrayOldLogs[chooseIndex].PulsFore)"))
                    Text("Vikt före: \((object.arrayOldLogs[chooseIndex].ViktFore)"))
                    Text("Planerad vikt efter:
                    \((object.arrayOldLogs[chooseIndex].PlaneradViktEfter)) } )
                    VStack(){
                        Text("UF volym L: \((object.arrayOldLogs[chooseIndex].UfVolymL)"))
                        Text("Blodflöde QB: \((object.arrayOldLogs[chooseIndex].BlodflodeQB)"))
                        Text("Artärtryck: \((object.arrayOldLogs[chooseIndex].Artartertryck)"))
                        Text("Ventryck: \((object.arrayOldLogs[chooseIndex].Ventryck)"))
                        Text("UF Rate L/tim: \((object.arrayOldLogs[chooseIndex].UfRateLtim)"))
                        Text("Konduktivitet: \((object.arrayOldLogs[chooseIndex].Konduktivitet)"))
                        Text("Effektiv dialystid:
                        \((object.arrayOldLogs[chooseIndex].EffektivDialystid)"))
                        Text("Ack QB L: \((object.arrayOldLogs[chooseIndex].AckQBL)"))
                        Text("Ack UFV L: \((object.arrayOldLogs[chooseIndex].AckUFVL)"))
                        Text("Vikt efter: \((object.arrayOldLogs[chooseIndex].ViktAfter)")) }
                    VStack(){
                        Text("Blodtryck efter:
                        \((object.arrayOldLogs[chooseIndex].BlodtryckAfter)"))
                
```

```

        Text("Puls efter: \\" + (object.arrayOldLogs[chooseenIndex].PulsAfter))
        Text("Kommentarer: \\" + (object.arrayOldLogs[chooseenIndex].Kommentarer))
    }

    Spacer()

    Button{
        self.showNewScreenWithLog = false

    } label: {
        Text("Tillbaka")
    }.padding(.horizontal,24)
    .padding(.vertical, 10)
    .background(.pink)
    .cornerRadius(28)
    .foregroundColor(.white)

    Spacer()
}

}

.navigationBarTitle("Protokollhistorik", displayMode: .inline)
Spacer()
}

```

```

class oldLogViewModel: ObservableObject{

    @Published var arrayOldLogs = [oldLog]()

    init()
    {
        getOldLogs()
    }

    func getOldLogs(){
        guard let userID = FirebaseDealer.shared.auth.currentUser?.uid else{ return }
        FirebaseDealer.shared.firestore
            .collection("log")
            .document(userID)
            .collection("samling")
            .order(by: "timestamp")
            .addSnapshotListener {

```

```

querySnapshot, error in
if let error = error {
    print("Error: \(error)")
    return
}

querySnapshot?.documentChanges.forEach({ change in
    if change.type == .added {
        let data = change.document.data()
        self.arrayOldLogs.append(.init(data: data))
    }
}
}

struct ProtocolHistory_Previews: PreviewProvider {
    static var previews: some View {
        ProtocolHistory()
    }
}

```