Bachelor's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Architecture and Design
Department of Design

Simen Tokerød Bergo
Sandra Kristiansen Smaaberg
Kristian Teppan

Development of «PreViS»

Video-assisted determination support in the emergency chain

Bachelor's thesis in Web development
Supervisor: Lefteris Papachristos
May 2022

NTNU
Norwegian University of
Science and Technology

Simen Tokerød Bergo
Sandra Kristiansen Smaaberg
Kristian Teppan

Development of «PreViS»

Video-assisted determination support in the emergency chain

Bachelor's thesis in Web development
Supervisor: Lefteris Papachristos
May 2022

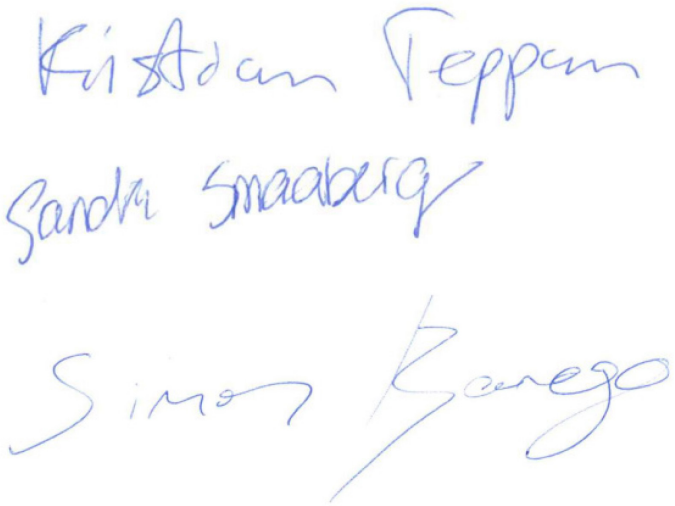
Norwegian University of Science and Technology
Faculty of Architecture and Design
Department of Design

NTNU
Norwegian University of
Science and Technology

Preface

Firstly, we would like to thank our employer, Peder Andreas Stokke. He has given us much valuable guidance and insight on the project, that we could not be without. We also wish to thank his colleagues Tom Bakken and Nils Halvor Gryting, which have been great resources for us, and their knowledge about the PreViS project has helped us understand its importance. We want to especially thank them for letting us be a part of such an innovative and exciting project, and hope that their project is realized, as it can help improve prehospital healthcare for the better.

We would also like to thank our tutor, Lefteris Papachristos, for valuable guidance and encouragement throughout this project. Lastly, we would like to thank family and friends, for giving motivation and support whenever needed. This project has been a big challenge for us, but at the same time extremely rewarding.



Abstract

Title: Development of «PreViS».

Date: 13.05.2022

Participants: Simen Tokerød Bergo, Sandra Kristiansen Smaaberg, Kristian Teppan.

Supervisor: Lefteris Papachristos.

Employer: Peder Andreas Stokke, Innovation advisor from the Innovation department at Sykehuset Innlandet HF.

Keywords: Web development, prehospital support, paramedics, user interface, video streaming, design.

Number of pages: 75

Number of attachments: 11

The PreViS project is an innovative attempt to use video technology as decision support to improve prehospital services, by enabling doctors to remotely assess patients before they reach the hospital. With insight from test users, we formulated the following thesis problem statement: *How can we make the interaction between paramedics and specialists more intuitive and efficient through a web application?* To answer this statement, we conducted semi-structured interviews with test users of the project, used relevant design theories, and compared streaming technologies to improve quality of communication. Our main goal was to develop an effective and interactive application that would enable doctors or specialists to evaluate patient needs more quickly, optimize resource utilization, and improve patient quality of life. Our final application is a remote, decision support tool that transmits video and audio between two users.

Sammendrag

Tittel: Utvikling av «PreViS».

Dato: 13.05.2022

Deltakere: Simen Tokerød Bergo, Sandra Kristiansen Smaaberg, Kristian Teppan.

Veileder: Lefteris Papachristos.

Oppdragsgiver: Peder Andreas Stokke, Innovasjonsrådgiver fra Forsknings- og Innovasjonsavdelingen ved Sykehuset Innlandet HF.

Stikkord: Webutvikling, prehospitalt, paramedics, brukergrensesnitt, videostrømming, design.

Antall sider: 75

Antall vedlegg: 11

PreViS er et innovativt forsøk på å ta i bruk videoteknologi som beslutningsstøtte for å forbedre den prehospitale tjenesten, ved å tilrettelegge for muligheten til å vurdere pasienter digitalt før de blir sendt til sykehus. Med innsikt fra testbrukere, formulerte vi følgende problemstilling: *Hvordan kan vi gjøre interaksjonen mellom paramedics og spesialister mer intuitiv og effektiv gjennom en webapplikasjon?* For å besvare problemstillingen utførte vi semi-strukturerte intervjuer med testbrukere, tok i bruk relevante design teorier og sammenlignet strømmeteknologier for å forbedre kvaliteten til kommunikasjonen. Vårt prosjektmål var å utvikle en effektiv og interagerende applikasjon som ville muliggjøre leger og spesialister til å evaluere pasientbehov raskere, optimalisere bruk av ressurser, og forbedre pasientenes livskvalitet. Vår utviklede applikasjon er et fjernstyrt beslutningsstøtte verktøy som sender video og lyd mellom to brukere.

Table of contents

1 Introduction

The healthcare system is one of the most important backbones of our society. It gives people comfort knowing they will be taken good care of if they were to be ill or injured. However, this comfort may not be as powerful for people living in rural areas or in distant locations. People do not have the same comfort in the healthcare system's offerings because of longer routes to emergency rooms and hospitals. To provide equal support to everyone in the population, health care should be available regardless of where a person lives (NOU 2015:17). PreViS, the project we are part of in this bachelor thesis, tries to help solve this problem by bringing the doctor to you, virtually. The Norwegian government wishes to improve prehospital[1] care for the public by providing better assistance where the patient lives rather than transporting them to the doctors first.

Our society's technological improvements have a significant impact on the current healthcare system, by continually presenting new opportunities to improve critical social processes. PreViS, the interface application we were tasked with developing, is also an illustration of how digitalization can help save lives. The prehospital service has transformed from a transport service with simple first aid possibilities into a vital component of medical emergency services. The project will help to advance research, technical development, and innovation in the use of video technology as assessment support in prehospital services, ensuring that as many patients as possible receive better and more effective care where they are. The emphasis is on improving video support in general, but particularly for patients in trauma and mental health care (Appendix 1). Sykehuset Innlandet's involvement comes from a shared problem statement with PICTA, that regions with large geographical distances, that make the distance between patients and health services big. This makes it hard to offer equal healthcare to everyone in the population.

With this as a background, the regional development plan proposed to start several innovation projects to improve this, one of them being PreViS. This project aims to help realize this goal, and give prehospital support a video communication tool, that allows paramedics in ambulances to work more efficiently and in synchronization with doctors remotely. It is an

[1] Prehospital is the part of the specialist health care service that handles patients outside of hospitals.

innovative project which is relatively new and there are currently no similar applications that solve this problem. Through interviews with the test users of the PreViS team's own prototype, and applying design principles, our goal is to design and develop an application that makes the interaction between the users easier.

1.1 Project description

PreViS[2] is a collaborative project between PICTA and Sykehuset Innlandet. PreViS is a Scandinavian cross-border investment in innovative environments. The project will contribute to research, technical development, and innovation in the use of video technology as prehospital assessment assistance. They are now working on a system that employs video communication as decision support for health personnel in the emergency chain (Appendix 1). The project is owned by PICTA[3], a Swedish prehospital arena in which multiple organizations collaborate to contribute to further advancements in information technology in the healthcare system (PICTA, n.d.).

Sykehuset Innlandet is therefore researching and developing the use of video communication as a decision support tool for emergency medical workers. The project's employer is Peder Andreas Stokke, innovation advisor at Sykehuset Innlandet. Together with a team of experienced paramedics, they are currently testing and further developing this technology. Paramedics will have a greater possibility of diagnosis and triage[4] by connecting specialists or doctors through video, ensuring that the patient receives better treatment, faster. The PreViS project will be finished in September 2022.

1.2 Problem description

The PreViS team are well on their way in developing and testing the existing prototype. They developed their own prototype to test the system. Usability and other design principles were not necessarily prioritized when they developed the mentioned prototype. As a result, the

[2] Prehospital Video i Samverkan

[3] Prehospital ICT Arena

[4] The process of quickly examining patients who are taken to a hospital in order to decide which ones are the most seriously ill and must be treated first

project owner contacted the Department of Design at NTNU[5] in Gjøvik, to get a different input on the project. We were given the opportunity to participate in the project by developing and designing our own application with an efficient user interface in order to improve communication between paramedics and specialists.

Designing an effective and intuitive graphical interface is vital in order to provide doctors viewing the patients with simpler access to relevant camera angles and information, allowing them to make better decisions (Bergo, Smaaberg, Teppan, 2021b, s. 1).

1.3 Thesis problem statement

Based on our employer's requests for the application, we formed a thesis statement:

How can we make the interaction between paramedics and specialists more intuitive and efficient through a web application?

By intuitive, we mean developing an interface that makes navigation effortless, as well as including features that were important to the user groups. By efficient, we mean how fast an experienced user can accomplish tasks (Usability.gov, 2019).

1.3.1 Research question

Considering this project focuses greatly on topics within healthcare, we chose to add a research question based on the UN's third sustainability goal, *Good health and well-being* (FN, n.d.). On this basis, we formed the following research question:

How can the developed application contribute to improve good health and life quality?

1.4 Existing prototype

Sykehuset Innlandet's innovation department has been working on this project together with PICTA for almost two years and has a functional prototype used for testing. Their prototype

[5] Norwegian University of Science and Technology

has a touch-based interface in the ambulance, with the main information being video feeds relayed to the doctor or specialist through the use of the existing system. As of now, there is a desire to achieve higher quality video streams that are transmitted through the system, while keeping latency as low as possible. This is something we hope to improve with our application. On the current prototype, as shown in figure 1, you must first choose your option, for example, the multimonitor. Then one of the patient streams is substituted with the selected option, as shown in figure 2.

Figure 1: Current PreViS system.

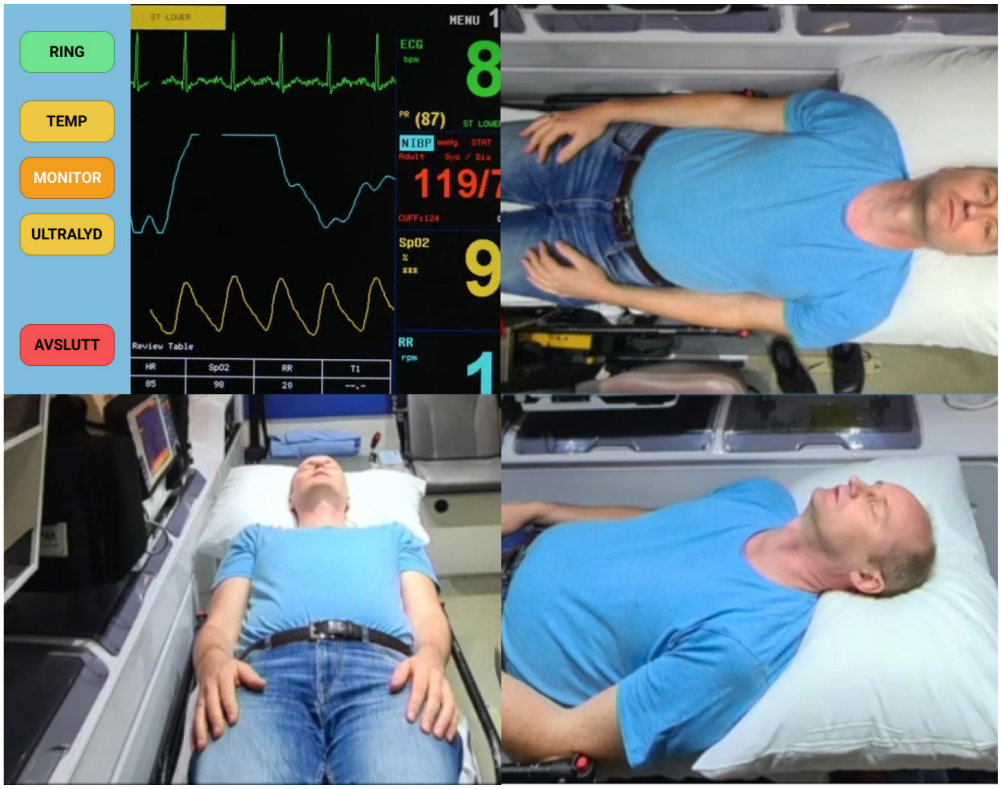


Figure 2: Multimonitor option replacing the patient stream.

1.5 Project Goal

Our goal is to develop an effective interactive application that will enable doctors or specialists to evaluate patient needs more quickly, optimize resource utilization, and improve patient quality of life. The application can help Sykehuset Innlandet by providing valuable solutions to the design of the interface and the technologies that they can apply to the PreViS project. To achieve this goal, we will attempt to create a functional application that improves design and usability, at the employer's request. Hopefully, our project will make the completed application easier to use, simplifying paramedic assignments and freeing up additional resources (Bergo, Smaaberg, Teppan, 2021a, s. 3-4).

1.6 Structure of the report

This report is divided into eight chapters. Chapter one discusses the project's context, the problem it addresses, and the objective it seeks to accomplish. Chapter two contains collected data from a previous project, user groups, semi-structured interviews, and the requirements specification for the application to be developed. Chapter three discusses the organization of the project. Chapters four and five discuss the design and development of our application, as well as the various methods and theories employed. Chapter six contains the final application

together with a test guide. Chapter seven summarizes our results. Lastly, chapter eight contains discussion, limitations, reflection, conclusion and areas for future improvements.

2 Background

2.1 Innovation in the health sector

In St.meld. No. 7 (2008-2009), it conveys that one of the techniques that the health and care industry should utilize is research and innovation in conjunction with the business community. This can produce patient services that give higher quality of treatment, increased efficiency, and better welfare. The collaborative Scandinavian cross-border project, PreViS, is a good example of this. Working toward seamless prehospital services across the border is a requirement for providing an equal health service to the local community, and it is a service development that benefits the entire population of Inner Scandinavia. The project aims to connect relevant actors from both the public and private sectors, in order to not only accelerate the development of prehospital video support, but also to build a solid, resourceful, and cross-border network that will continue to contribute to the development of prehospital services long after the project is completed (Appendix 1).

The issues must be solved through electronic collaboration, with greater emphasis on appropriate organization and the implementation of medical-technical solutions that enhance treatment chains. In order to give patients a better and more complete health service, the government has developed a collaborative reform in the health sector (St. meld. no. 7 (2008-2009)).

To summarize the collaboration reform, the government has chosen to focus on five major measures. The first primary measure emphasizes a clearer patient role, the second on a new future municipal function, and the third on the implementation of financial incentives. Main measures four and five are about the specialist health service, which is to be developed so that it can use its specialized knowledge more effectively and facilitate clearer priorities. The fourth major measure is to expand the specialist health service so it can make better use of its specialized skill. It is a goal to strengthen the specialist health service in order to provide specialized high-quality health care to the population of Norway. The provision of high-

quality services that address both national and worldwide technology and method development within medicine and health sciences, as the PreViS project does, results in an important foundation for the population's assurance (St. Meld. No. 47 (2008-2009)).

2.2 Coordination in the health sector

The ability of health and care services to divide work among themselves in order to reach a shared, agreed-upon goal, as well as the ability to carry out duties in a coordinated and rational manner, is referred to as interaction as described in St.meld. No. 47 (2008-2009). It also states that the government will aim to establish a future health and care service that both meets the patient's need for integrated services and responds to important socio-economic concerns through the collaboration reform. Equal access to good and equal health and care services, regardless of personal finances and place of residence, will continue to be the most important pillar in the Norwegian welfare model. Despite the implementation of initiatives relating to innovation, financial incentives, organization, and digitization, there is still a lot of potential for improvements in terms of collaboration. Because there may be a lack of awareness about each other's professional practices, it is critical that the many actors behave in the best interests of the whole rather than on their own. The engagement of health professionals must be inspired by a shared professional and social understanding according to St. Meld. nr. 47 (2008-2009).

2.2.1 How does PreViS contribute to coordination?

Langhelle *et al.* (2009) claim that outside of hospitals, the ambulance service, along with emergency medical service and municipal emergency services, form the backbone of emergency medical care. As a result, good inter-actor communication is critical for providing high-quality emergency medical services to the public. This is precisely what the PreViS project can offer. It permits communication between the paramedics and the doctor or specialist by utilizing the application. They can collaborate through the application to reach a common purpose, which is to deliver the best possible care to the patient. Paramedics are extremely knowledgeable in most fields, but they are not specialists in various disciplines. But with a specialist contributing through the PreViS system, they can accomplish a lot (Stokke, 2021).

2.3 Costs in the specialist health service

The purpose of SAMDATA is to generate statistics and analysis on trends and variations in the specialized health service. Over the previous three years, the cost of specialized health care has steadily increased. We have taken numbers and data from SAMDATA's reports and looked at how the PreViS project can help reduce costs.

As Innlandet is a large county, resource management is important, as ambulances need to cover great areas on their own, to provide sufficient support. Ambulances can remain out on assignment for up to six hours, which implies that ambulances must travel across greater distances to maintain their emergency preparedness (Stokke, 2021). Ambulance and patient transportation expenditures accounted for 6.7 percent of overall costs in the special health service in 2018 (Helsedirektoratet, 2019). The costs of ambulance and patient transport in 2019 accounted for 6.8 percent of the total costs (Dalheim og Vågseter, 2020). Ambulance and patient transportation costs climbed by 45 percent between 2008 and 2020, accounting for 6.7 percent of total costs in 2020 (Dalheim, Grøtheim, Vågseter, 2021).

One of the project's expected results is to reduce the number of patients admitted to hospital from prehospital services (Appendix 1). SAMDATA experts assessed that one day's admission to the hospital in general costs 17 000 NOK[6], per patient (Legemiddelverket, 2020).

If the project is successful, ambulances can treat patients where they are and sooner be ready for the next assignment. This can make the general preparedness in the area better, as ambulances might not need to be relocated. If these described costs can be reduced, unnecessary expenses will decrease considerably.

2.4 Previous findings

We were introduced to the PreViS project in the previous course IDG3101 In-depth project. During the in-depth project, we gained relevant knowledge, and the results obtained were valuable guidance that we included in this project, allowing us to begin work at a higher technical level.

[6] Norwegian kroner

We built a foundation for developing a video transmission application through in-depth interviews, prototyping, and insight work. We performed semi-structured interviews to learn more about the project and what might be useful to incorporate in the future. Comparison of the codecs, research on coverage and speed, resolution, frame rate, and bitrate were some of the studies we did to obtain information. The work that was done in the in-depth project provided a theoretical understanding of which technologies would best help this project. The subject also gave us a small insight into working with video streaming technologies. This influenced the decision on what technologies to use.

During this previous project we established a solid theoretical foundation for understanding the technologies involved in online video transmission. We also got a good insight into the PreViS project and all the involved parties. It is important to separate what we did in the previous project, and in this project. Regarding the choice of technology, the conclusion of the in-depth project gave us indications that WebRTC[7] and Socket.IO would suit the needs of the project (Bergo, Smaaberg, Teppan, 2021b, s. 20). But this knowledge was heavily based on insight from the employer, and we later decided to compare different streaming protocols to get a better foundation for our own application. We also found out that it is very hard to control deeper settings in WebRTC, as this is implemented into the browsers and not easily accessible without coding a self-made solution from the ground up. This means we could not change codecs that managed the compression of the video, and efficiently set bitrates in the video transmission application. Therefore, the project gave us good insight into some things that would *not* work and allowed us to look at other factors, instead of wasting time on this when starting the bachelor project.

However, there were several shortcomings in the prototype from the in-depth project. Each user was given a unique ID to connect with, but no automated sending of this ID was implemented. This means each user had to copy this ID, and manually send it through another media channel, to the user they wanted to connect with. This proved very inefficient in use and gave us other ideas about how to solve this issue in this project.

[7] Web Real-Time Communication

2.5 User groups

We have done extensive insight work on the PreViS project since working on it in the preceding in-depth project. During the research process, the first user group was identified as doctors and specialists, and the second user group was identified as paramedics. The application will be used on a regular basis by the user groups. Semi-structured interviews and user testing were conducted based on the user groups. This allowed us to gain a lot of information on how the user groups imagined the application, and what could be improved from their own prototype.

2.6 Organizing information

Since this project was well underway when we took part in it, the PreViS team already had an existing prototype. As Jamie Steane (2018) states in his book about principles and processes of interaction design, it is essential to gain insights into the current products' limitations and capabilities. To better understand a user's expectation of the product's functionality and content, it can be a good idea to retrieve information through research methods. We decided to achieve and gain insight by conducting semi-structured interviews.

2.7 Semi-structured interviews

Based on a book written about design methods by Tomitsch *et al.* (2018), interviews are one of the most adaptable research tools that designers have at their disposal. They can be used to elicit information from experts, users, and other stakeholders at numerous stages of the design process. An interview may be conducted to ascertain information about a problem area, to ascertain users' perceptions of concepts, or to elicit detailed feedback on a prototype. In addition, it is necessary to carefully choose participants to ensure that they appropriately represent the target audience; participants who are representative of the user groups will provide valuable insights.

To get a good idea of how we should design and develop a good solution, it became clear that semi-structured interviews with relevant participants were necessary, to get quality input from the ones with the best knowledge of the system.

The interview process began before any design and prototyping were started. The interviewees were two paramedics who have participated in demos and tested the system regularly, and an emergency physician who also has been involved with testing the current system.

Before conducting the interviews, we prepared an interview guide. The interviews were divided into three parts which were participant, PreViS prototype, and closing questions. Prior to the interviews, we asked each participant if it was possible to record the interviews. This way it was easier to transcribe the interviews later and ensure that we received the correct information. The interviews took place over Microsoft Teams and were used as a tool for audio recording as the program has a built-in function that records meetings (Microsoft, n.d.). During the interviews, we also had the opportunity to ask follow-up questions that were not written in the interview guide.

We began transcribing after the interviews were completed. It was easier for us to transcribe and include all the material that was spoken during the interviews when we used an audio recording as a tool. The transcript was distributed to each group member so that everyone contributed. The total number of minutes was distributed to each member so that everyone had the same amount. Transcripts of each interview are attached to the appendix folder together with the interview guide.

To get an overview of the information from the interviews, we systematized a structure. Our starting point was the main questions about the prototype. Conducting these interviews gave us a greater understanding of what is desired and what can be improved.

	Question 5 (prototype)	Question 6 (prototype)
Informant 1	Yes, because there is one important thing in this. First, it must be seamless and easy to use.	What we want for the system is to be controlled from the other end. That the person who is decision support can control.
Informant 2	It is very simple and it is absolutely essential that it is easy to use. It must work.	Good sound, stable image, and good coverage. Can work without using the system him or herself.
Informant 3	The most important thing is if you want it to get used a lot, it must be easy.	To function as optimally as possible all the way, so that the sound and image can be trusted.

Table 1: An overview of the informants' answers to key questions.

2.7.1 Results from the interviews

To summarize the interviews we held with the three informants, it was emphasized that the primary need for the application was to be as simple as possible to use, which reduces the chances of making mistakes. High-quality video and sound, and low latency were also key features the informants wanted for the application to work as intended. In regard to design and layout, the informants specified that few button presses is crucial, and no unnecessary components or functionality needs to be present. This means that the actual interface of the application will only consist of control buttons, and video feeds from the ambulance.

2.8 Requirement specification

The application's objective is to enable doctors to view and evaluate patients digitally. This enables them to give assistance to paramedics in particularly difficult situations. This expedites treatment while avoiding unnecessary journeys that can be stressful for both the patient and the emergency response team. To accomplish this sole objective, we chose to

create an application requirement specification that encompassed both functional and non-functional requirements, based on the input from the employer and informants.

Functional requirements	Non-functional requirements
▪ Low latency ▪ High-quality feed ▪ Remote control from the hospital	▪ Simple information architecture ▪ Error prevention and usability ▪ Accessible design

Table 2: Specification for both functional and non-functional requirements.

2.8.1 Functional requirements

2.8.1.1 Low latency

Low latency is critical to ensure the best possible communication between paramedics and doctors or specialists using the application. Low latency is also critical to make sure that everything occurs in real-time without delay. This can help prevent information loss between the two parties, as even small amounts of latency can have a significant impact on the quality and efficiency of the communication.

2.8.1.2 High-quality feed

To ensure that doctors or specialists can see their patients clearly and provide adequate support, the application must deliver video with a high resolution and a reasonable refresh rate. This requirement is contingent upon a number of factors, several of which are beyond our control, such as signal interference and areas with inadequate coverage.

2.8.1.3 Remote control from the hospital

The application must be monitored and controlled from the hospital, not the ambulance. They have little or no time to use the application when they are working with a critical patient. As a result, it is critical that the person providing decision support has control over which images they view. The semi-structured interviews revealed that the paramedics in the ambulance

desired to do as little as possible after starting the application, allowing the doctors or specialists to have control.

2.8.2 Nonfunctional requirements

2.8.2.1 Simple information architecture

The information gathered during the semi-structured interviews revealed that one of the application's highest priorities is that it should be simple to use and understand, particularly for the paramedics in the ambulance, who are busy treating patients while also starting the application; thus, this process must be as simple as possible to avoid diverting attention away from patients.

2.8.2.2 Error prevention and usability

Another point that came through in the interviews was the application's low tolerance for errors. The design and usability of the application should prioritize the critical buttons, making it difficult for users to become confused or press the incorrect buttons. Making it possible for paramedics to focus exclusively on patients is critical; it must be an aid, not a liability.

2.8.2.3 Accessible design

The application's ability to be used by everyone is a primary priority. In order to achieve this, we must focus on user-friendly design. All of the application's elements must be straightforward to comprehend and should be indicated with colors. This means following WCAG guidelines on color, contrast, and other relevant principles.

3 Project organizing

In the startup phase of this project, we made a project plan, where we planned the different stages we should do. It was important to do this in the early stages, so we had a plan to follow if something was unclear. This plan also contained an estimated workload, how frequent we should meet, and role distribution. By defining this before the project started, we could save time by avoiding misunderstandings and unnecessary discussions. We also defined some group rules regarding communication, work hours, prevention of data loss, and conflict

resolution. It also included some decision points, to ensure a good workflow by dividing the project into different phases.

3.1 Group organizing

During this project we wanted each of us to have a leadership role, therefore the roles got changed throughout the project phases. This was done to play to each of our strengths, as we have a lot of experience working together and know the workflow and dynamic of each other very well.

Members	**Responsibility**
Simen Tokerød Bergo	Project manager
Kristian Teppan	Development manager
Sandra Kristiansen Smaaberg	Report manager

Table 3: Role distribution.

Organization of this project was set up and edited in Monday, with a GANNT chart, kanban boards, and tables, to keep track of tasks. New weeks were started with recap meetings of last week's progress, and a more specific plan of the coming week and tasks that needed to be completed was formed, usually agreed upon by everyone and divided by the project manager. The comprehensive GANNT chart can be seen in figure 3.

Figure 3: Overview of our GANNT chart with main deadlines.

3.2 Interaction tools

We chose to use many interaction tools we were already familiar with, to improve workflow and avoid using too much time getting used to new ones. All these tools also allowed for real-time editing, which has been appreciated in a time where physical meetings are not a guarantee.

Discord

Tens of millions of people use Discord, a free voice, video, and text chat application. The vast majority of servers are privately hosted and invite-only. Several larger servers are open to the public, and any user may create a new server and invite their friends for free (Discord, n.d.). We chose Discord as our meeting tool because it is well-known and frequently used by all of us.

Monday

"Monday.com is a work operating system that powers teams to run projects and workflows" (Monday.com, n.d.). Monday was designated for planning and structuring. None of us had previously used the tool and it made it simple to keep track of what should be accomplished, when, and by whom.

Google Docs

Google Docs is a web-based word processor that enables you to create and format documents as well as collaborate with others (Google, n.d.). Google Docs enables you to create and edit text documents directly in your web browser, without the need for additional software. Multiple people can work concurrently, and you can also see people's changes as they occur (Google, n.d.). We have been using Google Docs for several years and are well-versed in the application. This was the tool that we preferred the most.

Microsoft Teams

Microsoft Teams is a digital conference tool, used for video meetings, messaging and collaborations (Microsoft Teams, n.d.). The employer and his innovation team mostly reside in Lillehammer, meaning most of the meetings have been digital, through the use of Microsoft Teams.

3.3 Meetings

As a result of the PreViS team being located in Lillehammer and their busy schedule, we had to mostly rely on digital meetings with them. However, we are used to online work and meetings due to the global pandemic over the last two years. The distance has limited the opportunities of physical meetings with the employer, participants in testing, and interviewing. Good interaction technologies as described in chapter 3.2, allowed us to have informative meetings, some testing, and interviews with the PreViS team.

4 Design

After gathering information both during the in-depth project, and interviews held during this project, we could start designing a visual prototype.

During the interview process, the design and layout of the application were important for the PreViS prototype testers, as they talked about it a lot. This means that a lot of the design choices made in this project was a result of the opinions and experience of the main users of the application. They pointed out clearly that specific functionality and fewer button presses are what matter to them when using the application in stressful environments. We still tried to follow accepted design philosophies; however, this has sometimes gone against the needs of the users. The application will be used in a closed environment, meaning it will only be used by healthcare professionals. These factors have made it hard to incorporate accessibility principles, heuristics, and interaction design theorems.

4.1 Theory

In order to meet the employers' and informants' wishes for simplicity and efficient design, we used a lot of information obtained from the semi-structured interviews. Some theory concepts were also included, that didn't interfere with the very specific requirements of the application.

4.1.1 Mobile first

To ensure a seamless design process, we chose the mobile first approach. This means that the smallest screen size should be designed first, followed by desktop and larger screens. Additionally, this process simplifies determining which components are necessary to retain,

as the limited space available reduces the amount of content (AdobeXD, 2021). Our application will be used on different size screens, with tablets in the ambulance and desktop or television screens in the hospital.

4.1.2 Colors and contrasts

One design area where we had some room to follow WCAG guidelines within the frames of the requirements, was contrast and color. Optimizing this can help improve the readability and make important elements stand out.

Web Content Accessibility Guidelines (WCAG) 2.1 covers a wide range of recommendations for making Web content more accessible. Following these guidelines will make content more accessible to a wider range of people with disabilities, including accommodations for blindness and low vision (W3C, n.d.).

This direct citation explains some of the importance of following WCAG guidelines, and since doctors also can have color deficiencies, designing with this in mind was important.

4.1.2.1 Color selection

Color is one of the most important and influential tools a designer has. In designs, it can set the brand tone and influence its image, draw user's attention, affect their emotions, and increase usability. However, finding the right combination of colors can be tricky and requires some basic knowledge and practice (Gordon, 2021, paragraph 1).

When creating our prototype, color played a big role in making the interface usable and accessible. Having strong colors with good contrast between backgrounds and buttons is crucial as the system will be used in stressful and time-critical environments. This means there is no room for mistakes. Strong colors as seen in figure 4 were therefore applied to CTA[8] buttons in the application, making them stand out. The dark gray color on the far left is used as a background color.

[8] Call to action

Figure 4: Color palette for the prototype.

Based on the semi-structured interviews, the informants were asked about how they associate green and red in regard to buttons, to which they answered that green represents start or initiation of something, and red with stop or close (Gryting, 2022). This means there is a much larger focus on usability rather than aesthetics, and this has influenced the finished application greatly. According to Gordon (2021), certain colors have achieved standard meanings such as red for stop, and green for go. It is also important to keep in mind that color associations can vary depending on context, culture and time (Snl, 2021). We aimed to make all the components in the application as usable as possible, by using color associations, and as accessible as possible by using strong colors with good contrast. The ambition was to test usability through SUS[9] user testing, and accessibility through relevant tools such as Wave[10] and WebAIM[11] contrast checker.

However, according to the Office for National Statistics (n.d.), color alone might not be enough for people with color blindness. According to an article by Hem (2004), around 900 male and 200 female Norwegian doctors have a weakened color perception. This represents 6.5 percent of physicians. Red-green colorblindness is the most common kind of colorblindness (Snl, 2020). It was crucial that we were aware of this, given the colors we had chosen. The Office for National Statistics (n.d.) recommended adding related symbols on the buttons to further emphasize their meaning. We chose to rather add text such as *ring* on the call button, and *avslutt* on the end button. This explains exactly what the buttons does and what will occur when they are pressed. This will be further reviewed in the discussion chapter of the report.

4.1.2.2 Contrast

In design, contrasts are one of the most important factors to catch the user's attention and draw them to specific elements. According to Gordon (2020), contrast creates a visible

[9] System Usablity Scale
[10] WAVE Web Accessibility Evaluation Tool
[11] WebAIM contrast checker

distinction between two objects in order to emphasize their separation. Contrasts are frequently expressed through differences in color, which we incorporated in our application.

4.1.3 Fitt's Law

Fitt's law is a model that represents the time it takes to move to and press a target. This is mainly based on the distance to the target, and the size of the target. A longer distance and smaller target mean a longer time used to precisely hit it (Interaction Design.org, n.d.). Fitt's law was applied to the project in an attempt to lower the time used to press buttons in the application. This could be used as a tool to directly make the application more efficient in use. After design and development, the PreViS prototype and our application were tested against each other, using a Fitts law calculator to find their respective scores (Codepen, n.d.). The outcome of these tests will be reviewed in the results.

4.2 Design tool

We used Figma as the design tool for the design process and production of both low-fidelity and high-fidelity prototypes. We were very familiar with the tool, having used it on a consistent basis throughout our bachelor's degree. "Figma is a free, online user interaction tool to create, collaborate, prototype, and handoff" (Figma, n.d.). Figma enables real-time collaboration between groups and made it easier for us to see what each of us were currently working on. It also made it simple to keep track of which tasks needed to be completed, and which had already been completed.

4.3 Methods

This chapter discusses the methods used to gather information and insight into how the final solution should be designed. Following that, we present the results of the methods used.

4.3.1 Sketches

"Rather than trying to verbalize what ideas are going to look like, they can be quickly represented through a sketch" (Tomitsch *et al,* 2018, s. 116). We took inspiration from the already existing prototype and formed our own ideas as to how the prototype could be as effective as possible. To express our various ideas and designs, we created sketches. The sketches were done on paper before we started working on a low-fidelity prototype.

Each of us created a sheet, outlining potential design solutions, which we then combined into a single sheet containing all our contributions. We divided the sketching into two halves for the ambulance and the hospital. This needed to be considered because multiple screens would be utilized.

4.3.2 Wireframes

Based on Tomitsch *et al.* (2018), line drawings that depict a product's or system's basic structure of functionality are called wireframes. Designers can use wireframes to think about and express what a design can achieve in a more formal approach. There are no high-fidelity elements, such as photos, fonts, colors, or typography, that could detract from the main goal. We utilized Figma to create wireframes, which made it easier to evaluate the system's functionality.

4.3.3 Low-fidelity prototype

According to Tomitsch *et al.* (2018), a prototype is a representation of a planned product. Low-fidelity prototyping permits the rapid exploration of ideas early in the design process. They can be used to reflect on a design, discuss design solutions within collaboration, and obtain input from future users through the use of usability testing. After the interview process and sketching, we could start designing a low-fidelity prototype. Our sketching concepts were turned into a low-fidelity prototype in Figma. The hospital's existing prototype was used as a reference, and we tried to simplify its UI[12] for greater efficiency. The low-fidelity prototype was created to test the application's content, functionality, and layout.

4.3.4 High-fidelity prototype

Real content and polished graphics may be included in the high-fidelity prototype. It is possible to use specialized wireframing, prototyping, and authoring technologies; however, complicated functionality may not be showcased (Steane, 2018). In our example, we used Figma to construct a high-fidelity prototype that included a clickable version with all of the desired features. This was done to provide more detailed input on the content, functionalities,

[12] User Interface

and appearance. It also allowed us to show clickable demonstrations for the employer and his test team, giving them further insight into our proposed application.

4.3.5 User testing

4.3.5.1 The System Usability Scale

The System Usability Scale (SUS) is a low-cost usability scale that can be used to assess the usability of systems on a worldwide scale. SUS is a ten-item scale that provides a comprehensive understanding of subjective usability evaluation. Participants must respond to questions that require them to choose between strongly agreeing or strongly disagreeing. SUS provides a simple number that indicates a complicated measure of the overall usability of the system under consideration. The scores on each question are not meaningful in themselves, but the total score for all the questions is, and it provides a score on the system usability scale. The questions will be graded on a scale of 1 to 5, with 1 indicating significant disagreement and 5 indicating strong agreement (Brooke, 1995).

4.3.5.2 Calculating SUS Score

To calculate the SUS score, we needed to sum the score contributions from each item. Each item's score contribution will range from 0 to 4. For items 1,3,5,7, and 9 the score contribution is the scale position minus 1. For items 2,4,6,8 and 10, the contribution is 5 minus the scale position. Multiply the sum of the scores by 2.5 to obtain the overall value of the system’s usability. SUS scores have a range of 0 to 100 (Brooke, 1995, p. 5).

According to Sauro the average score on the SUS is calculated to 68 (2011, as cited in John Brooke, 1995; 2013, p. 36).

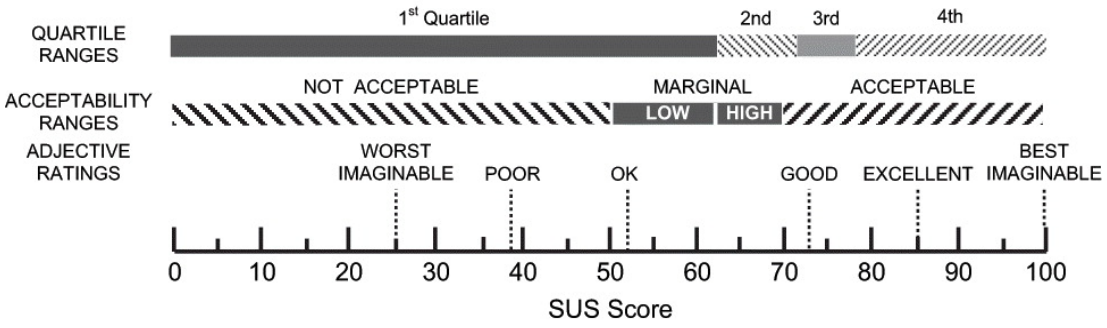


Figure 5: A comparison of mean System Usability Scale (SUS) scores by quartile, adjective rating, and the acceptability of the overall SUS score.[13]

4.4 Design results

This chapter shows how the discussed methods are applied to the design, from sketching to prototyping.

4.4.1 Sketches

We needed to create sketches for both the ambulance and the hospital because the application will have two sides. Due to the fact that the ambulance and hospital have screens of varying sizes, this was also something we needed to consider. The ambulance's design included a variety of options for the navigation bar and the patient stream. This was done to determine how our various ideas would appear in the application that we were going to create.

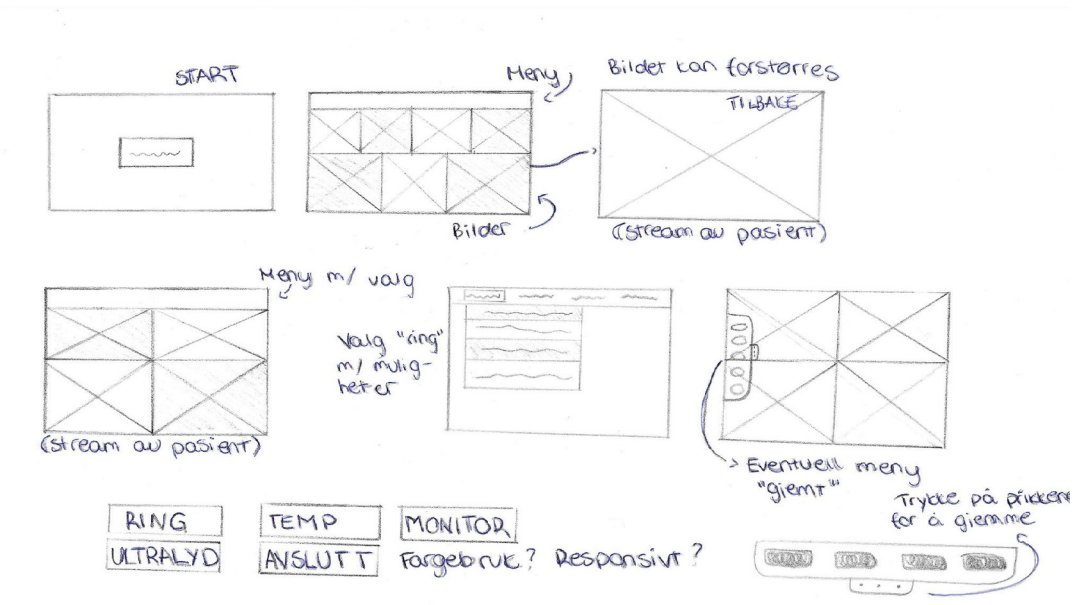


Figure 6: Sketches of the ambulance side.

[13] An empirical evaluation of the system usability scale.

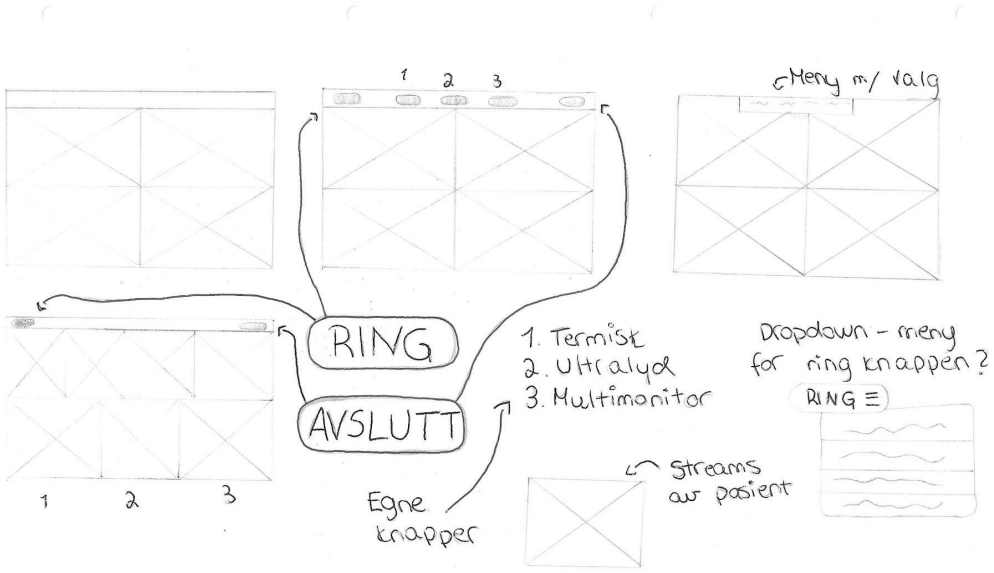


Figure 7: Sketches of the hospital side.

4.4.2 Wireframes

4.4.2.1 First draft

We selected to add four patient streams in the first draft of wireframes as well as options like multimonitor, ultrasound, and thermal being displayed at all times. We agreed that all images should be displayed on both the screens, for the ambulance and the hospital. This was done so that if an angle was set incorrectly, the ambulance crew could adjust it.

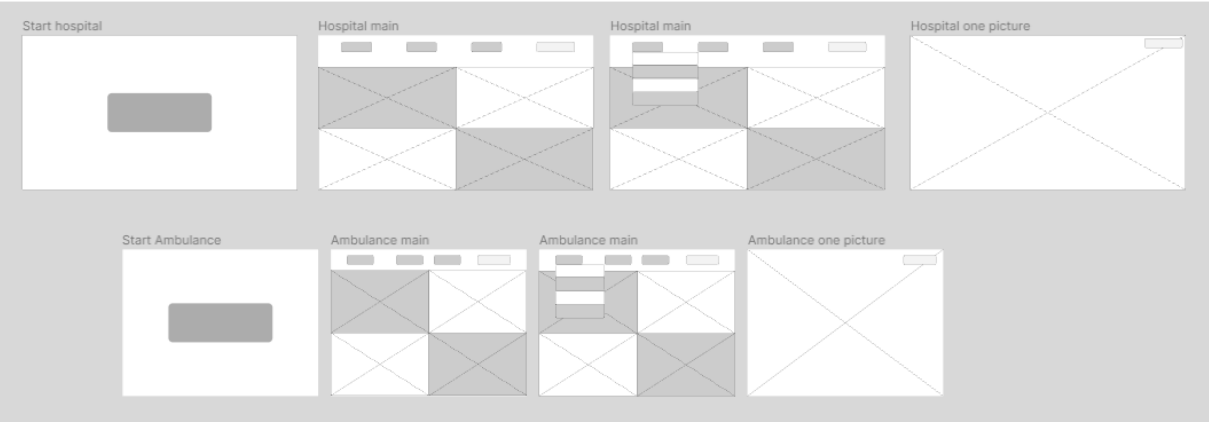


Figure 8: Wireframes for both hospital and ambulance prototype.

4.4.3 Low-fidelity prototype

4.4.3.1 First draft

We chose to include all the buttons in the first iteration of the low-fidelity prototype, just as they are in the existing PreViS prototype. Additionally, we decided to keep all of the patient's streams active at all times, except when they pressed one of the buttons, at which point an image was replaced with the selected feature. The first iteration of the low-fidelity prototype is depicted in figure 9.

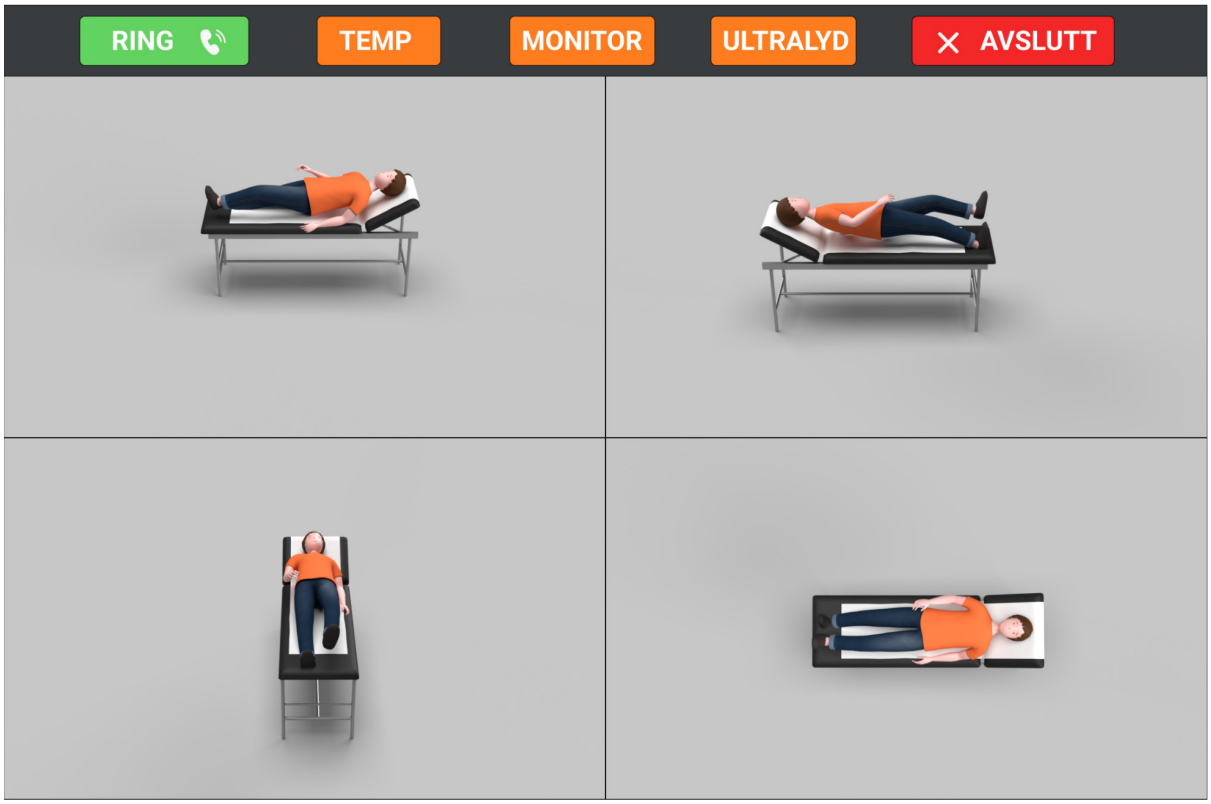


Figure 9: First iteration of the low-fidelity prototype.

4.4.3.2 Second draft

After completing the first draft, we quickly realized that some changes were necessary. Because we wanted the application to be as simple as possible, the buttons for multimonitor, ultrasound, and temperature were completely removed. We decided to bring up all the cameras and streams simultaneously to allow users to view everything without pressing a button. This was also done to simplify the ambulance's interface and overall appearance.

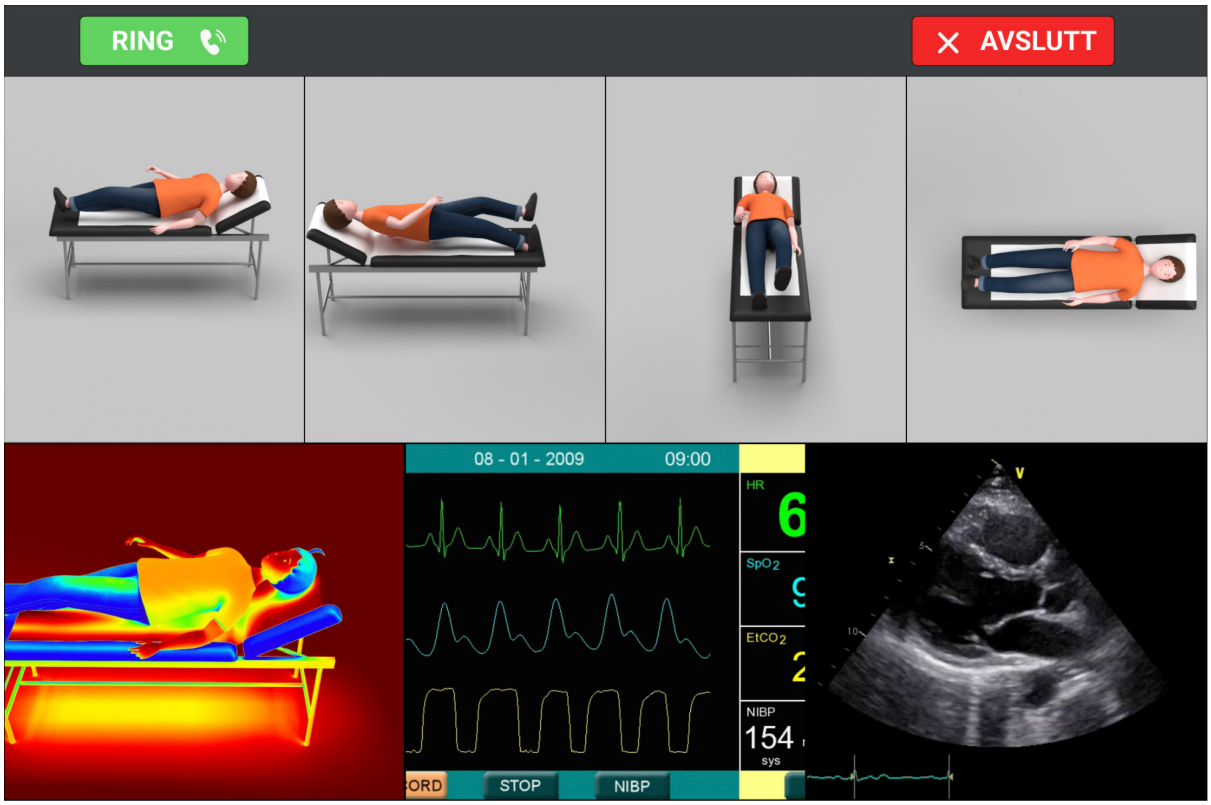


Figure 10: Second iteration of the low-fidelity prototype.

The buttons are positioned at the top of the user interface and made large and clear to be faster and easier to interact with. They are both placed on each side of the interface to minimize the possibility of pressing the wrong button.

4.4.4 High-fidelity prototype

4.4.4.1 First draft

The initial draft of the high-fidelity prototype was designed following the second iteration of the low-fidelity prototype. The prototype underwent a few minor adjustments. We altered the colors of the buttons and the navigation bar, as well as the button icons.

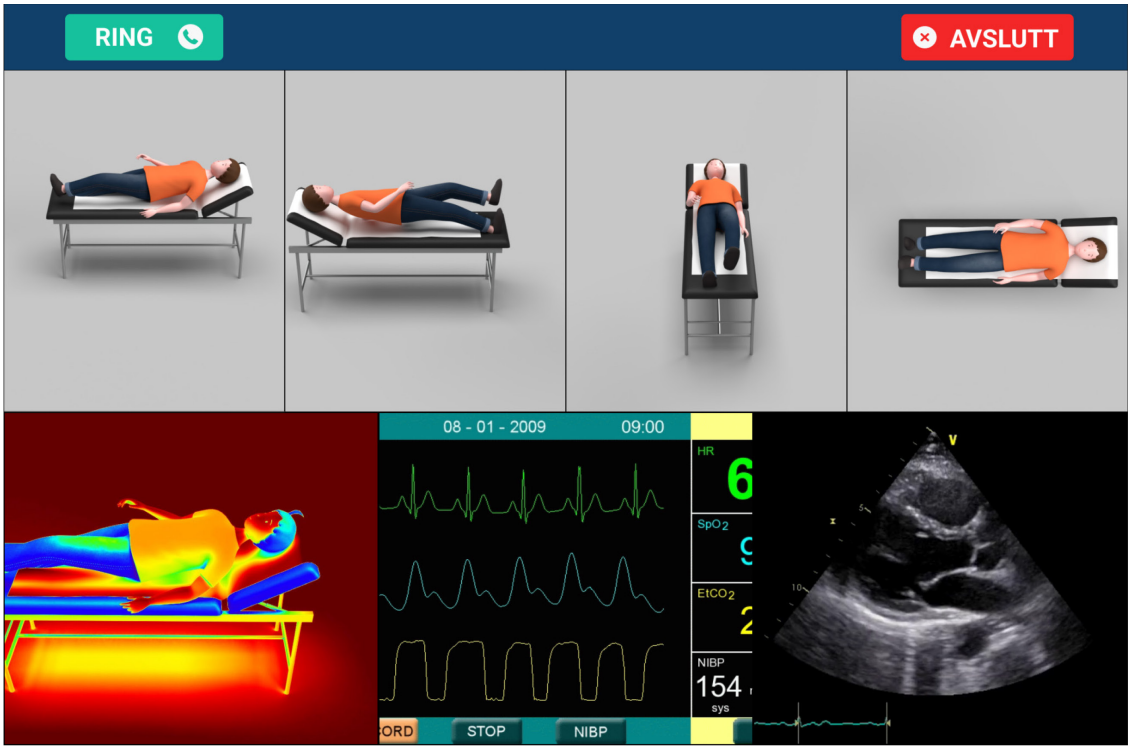


Figure 11: First iteration of the high-fidelity prototype.

4.4.4.2 Second draft

We were notified by the employer that the initial draft of the high-fidelity prototype needed to be altered after we completed it. As with ultrasound, thermal, and multimonitor, we meant that all the images of the patient should be displayed at the same time. According to the employer, there were only two streams of the patient, as well as the multimonitor that should be exhibited at all times. As a result, there was a decision to re-add the buttons. We chose to remove the icons from the call- and end buttons from the low-fidelity prototype in order to keep the application as basic as possible.

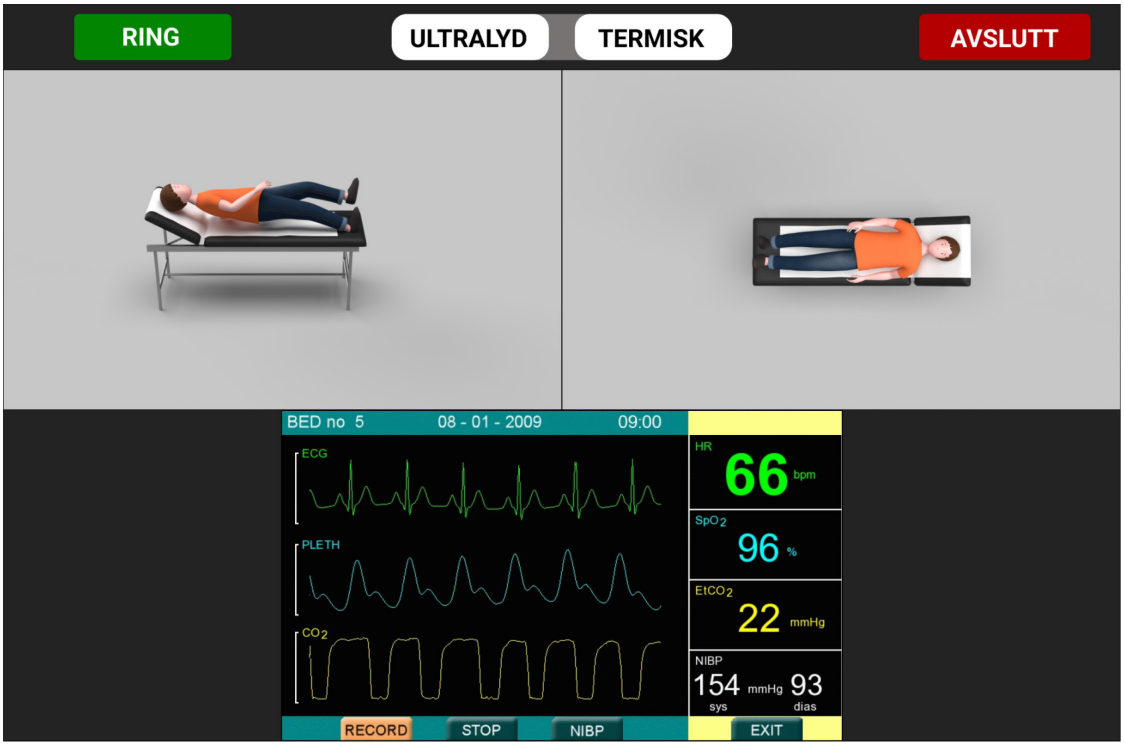


Figure 12: Second iteration of the high-fidelity prototype on the ambulance.

Another implementation requested by the informants was a back button when in fullscreen mode on one of the video feeds. Regular browser behavior allows for pressing anywhere on a full screen element to go out of it, but the informants believed this could be counter-productive in an ambulance driving fast or on unstable surfaces, as this could lead to accidental screen touches (Gryting, 2022). Therefore, we implemented such a button in the prototype, as seen in figure 13.

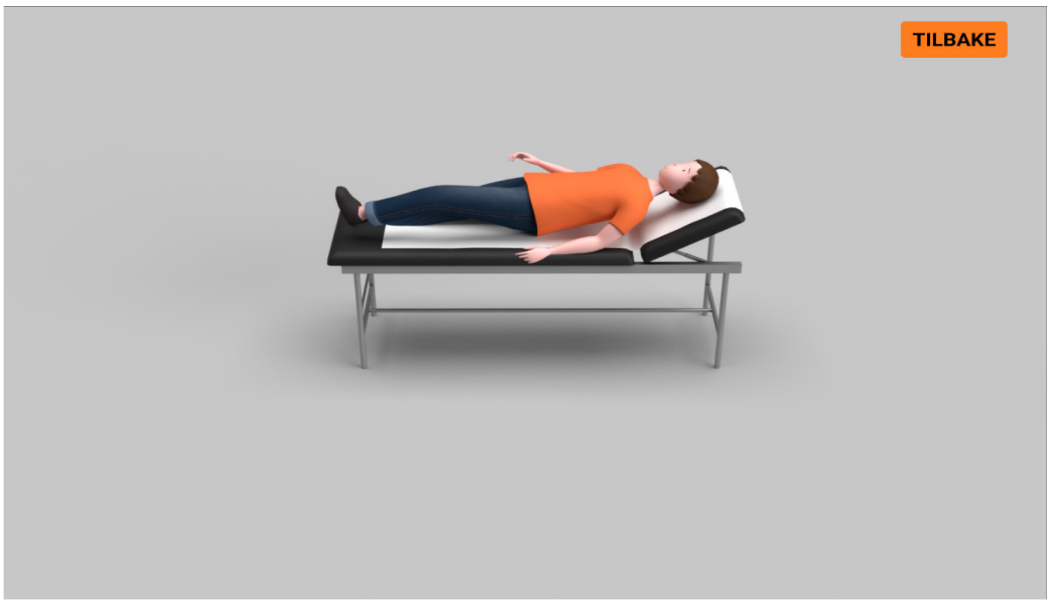


Figure 13: Patient view in fullscreen mode, with back button visible.

4.5 Summary of the design phase

During the design process, we utilized relevant design concepts to create a prototype. We employed techniques such as sketching and wireframes to establish the prototype's starting point. We then moved on to low-fidelity prototyping, followed by high-fidelity prototyping once the concept was complete. Several iterations were designed to establish the final prototype. User testing was also conducted based on our prototype. This resulted in a solid foundation for bringing the mentioned prototype into the phase of development.

5 Development

The high-fidelity Figma prototype served as the foundation for the application's design and layout. Developing the solution entails building a functional application from the ground up, utilizing the most appropriate technology and programming languages for the project's requirements. Throughout the previous in-depth project, we researched and developed a small and simple working prototype. The findings from the in-depth project were used to refine the prototype's design and functionality.

5.1 Development tools

GitHub

"Millions of developers and companies build, ship, and maintain their software on GitHub - the largest and most advanced development platform in the world" (GitHub, no date). GitHub is a platform for collaborative development and version control. It enables collaboration between you and others on the same project (W3, no date). We were well acquainted with GitHub from previous projects, and this made it easy for us to collaborate when it came to development.

Visual Studio Code

"Visual Studio Code is a lightweight but powerful source code editor which runs on your desktop and is available for Windows, macOS and Linux" (Visual Studio Code, no date). We chose Visual Studio Code as our code editor because we were already familiar with it, having used it for all of our school assignments throughout our bachelor degree.

5.2 Technologies

Numerous technologies and modules are required for the streaming application to function properly. Choosing how and which technologies to implement has a significant impact on the project's outcome and the working application's success. According to the employer's insights, developing a web-based application would be the most practical option for all users in terms of time spent connecting to the application (Stokke, 2021). In order for our application to work, we had to look at different streaming technologies that were available. It's a changing field, like the technology industry in general, where new technologies arise frequently.

5.2.1 Streaming technologies

According to a book written by Mohsen Amini Salehi and Xiangbo Li (2021) on low-latency streaming, the streaming protocols have gotten better regarding latency during the last years. Beginning with progressive downloading, which is a streaming method where viewers had to wait for the whole content to download before watching. The latency becomes very high when the user have to wait for all of the content to download previous to watching it. This led to the evolvement of newer and faster streaming protocols. Adobe solved this problem by introducing RTMP[14] streaming server, which splits the content into smaller chunks, allowing the viewing to begin as soon as each chunk has loaded, thus improving the latency. However, the RTMP needed a dedicated server and the connection could be blocked by the firewall due to its port number being different from HTTP[15], causing the content to not reach its destination.

Later on, ABR[16] takes over the streaming world, with the most common protocols being HLS[17] and DASH[18]. These protocols use the same approach as RTMP, by splitting up the media into smaller segments. The difference is that HLS and DASH are based on HTTP, which solves the firewall problem by using the webserver directly instead of a separate server.

[14] Real-Time Messaging Protocol

[15] Hypertext Transfer Protocol

[16] Adaptive Bitrate Streaming

[17] HTTP Live Streaming

[18] Dynamic Adaptive Streaming over HTTP

However, the small segment sizes of HLS and DASH can cause a high amount of HTTP request traffic, which can cause higher latency. This can be adjusted, and HLS recommends a segment size of 10 seconds, while DASH recommends 6 seconds. Although the ABR protocols have much lower latency than progressive downloading, the latency will still be around 30 seconds, which is way too high for live streaming (Salehi and Li, 2021).

The most recent protocols to improve this issue are low latency ABR protocols, such as Low-Latency HLS (LLHLS), CMAF[19], and WebRTC. LLHLS and CMAF split the segments into even smaller chunks, as small as a few frames, and use chunk transfer encoding technology to encode and deliver the frames. These protocols need support for this encoding technology in the player on the receiving end. This means it plays dynamically as the frames are being received and decoded, reducing latency to less than 5 seconds.

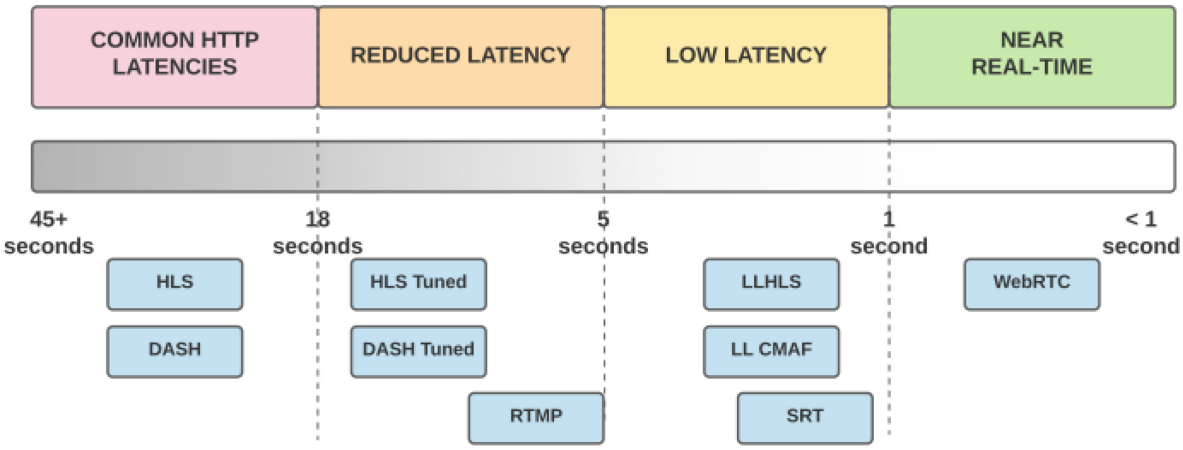


Figure 14: Latency for streaming protocols[20].

According to Salehi and Li (2021), the fastest streaming protocol is WebRTC, which is based on the UDP[21] transport protocol and can stream the media content with a latency of only 1 second or less. This is possible because UDP has no guarantee for delivering the data being sent. This is preferable in real-time streaming because losing some data is better than waiting a long time for the re-sending of the data (Mdn, n.d.). Figure 14 displays the latency comparison of the different protocols, and WebRTC is clearly the best option for low latency.

[19] Common Media Application Format

[20] Latency for streaming protocols

[21] User Datagram Protocol

5.2.2 WebRTC

WebRTC is a technology that allows video, sound, and other general data to be sent directly to the receiving browser or application, without going through a server. The technology is now available in all major browsers and is an open web standard with JavaScript-API[22]s in the most popular browsers. WebRTC is peer-to-peer communication, in which user one's browser connects directly with user two's browser. This means that while the communication itself does not require the usage of a server or API, a signaling server is needed to create the connection. This signal server sends and receives information that allows WebRTC to determine which device or browser to connect to (WebRTC, n.d.).

To create the peer connection WebRTC needs a signaling server to send an offer and create an answer using the ICE[23] framework. The ICE framework makes it possible to bypass firewalls, give a unique address if the device does not have a public IP[24] address, and relay data through a server. These servers are called STUN[25] and TURN[26] servers. NAT[27] is a way of translating the device's private IP address to the router's public IP address to make it accessible. Figure 15 visualizes how the ICE framework are working.

[22] Application Programming Interface

[23] Interactive Connectivity Establishment

[24] Internet Protocol

[25] Session Traversal Utilities for NAT

[26] Traversal Using Relays around NAT

[27] Network Address Translation

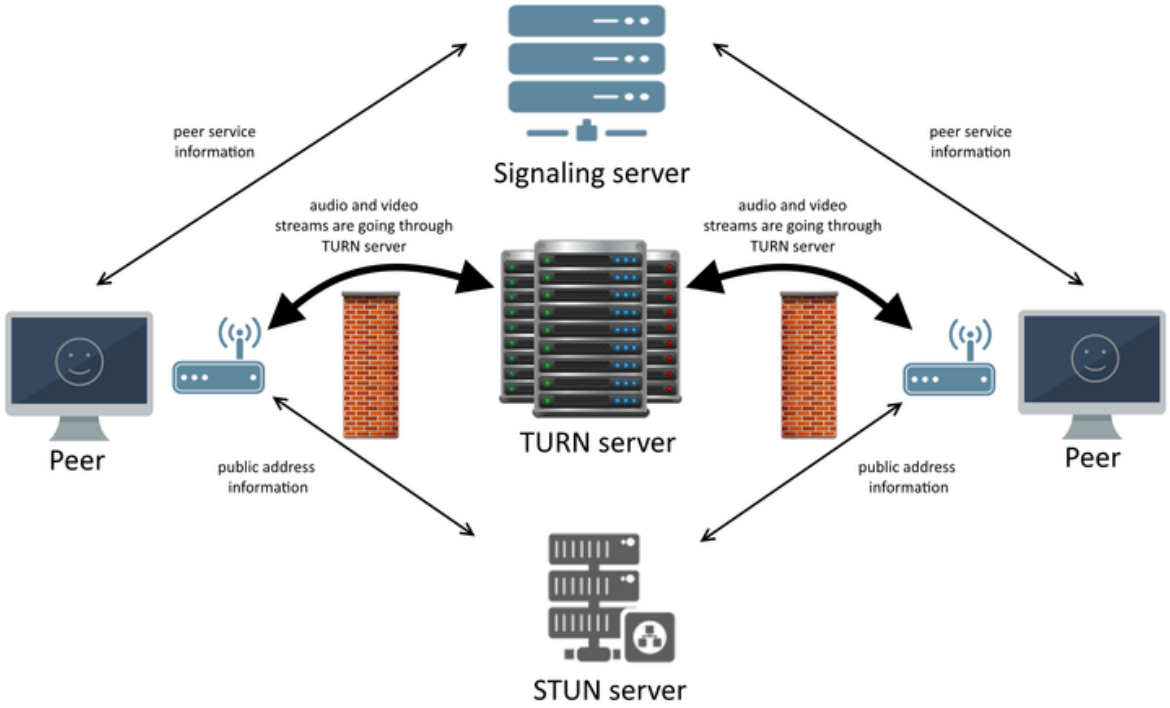


Figure 15: Illustration of the ICE framework, using STUN and TURN servers.[28]

5.2.3 Xirsys

To establish STUN and TURN servers we needed a service to provide these servers. Although companies like Google have some free servers available, we wanted to avoid using these as they are public and open for anyone to use. By researching available options, we found a service called Xirsys which provides free STUN and TURN servers. These are also protected and you need credentials to access them, and this will prevent unwanted access on the servers. The downside of these being free is the maximum bandwidth is set to 500 megabytes.

5.2.4 Socket.IO

Socket.IO is a library that allows for bidirectional, event-based, and low latency communication between a client and a server by creating a data channel. It uses the WebSocket protocol, which allows a two-way interactive communication session between the browser and server. Socket.IO includes HTTP long-polling as an automatic fallback if the WebSocket connection fails. Socket.IO also includes a server-side and client-side library, which makes it easy to configure on the client- and server-side. Socket.IO has a functionality called rooms, which allows clients to join a room and emit messages to all clients currently in the same room (Socket.IO, n.d.).

[28] Illustration of ICE

5.2.5 React

“React is a declarative, efficient, and flexible Javascript library for building user interfaces. It lets you compose complex UIs from small and isolated pieces of code called “components”” (ReactJS, n.d.). Component based development allows for fast troubleshooting and solving. Identifying the different components needed to build the system is important, as it decides how state and context are transferred to where it is needed. In react, state is a way of storing data to be used in components, and context is a way of transferring data, as well as states, across different components (ReactJS, n.d.). The components needed were discussed to get an overview, primarily taken from the final iteration of the high-fidelity prototype.

5.3 Chosen technology

After conducting research on the technologies mentioned previously, the best option regarding latency was WebRTC. Because the application will be used on a variety of devices with little tolerance for lag, the well implemented WebRTC technology was the best fit for our needs.

Xirsys was the best option for STUN and TURN servers, because we will have our own private and protected servers instead of public. This will reduce the traffic and improve the speed. Xirsys is also much simpler than some other options we looked at, which required us to set up our own servers, which would be time consuming and demanding since this technology is new to us. By using Xirsys’ servers, it will be more reliable, time-efficient, and cost-efficient. The downside of choosing these servers is that bandwidth capacity will be limited to 500 megabytes for the free plan. However, we decided that this was preferable to the other options, because 500 megabytes would satisfy our needs during development and testing, and it will be possible to upgrade the servers later, if needed.

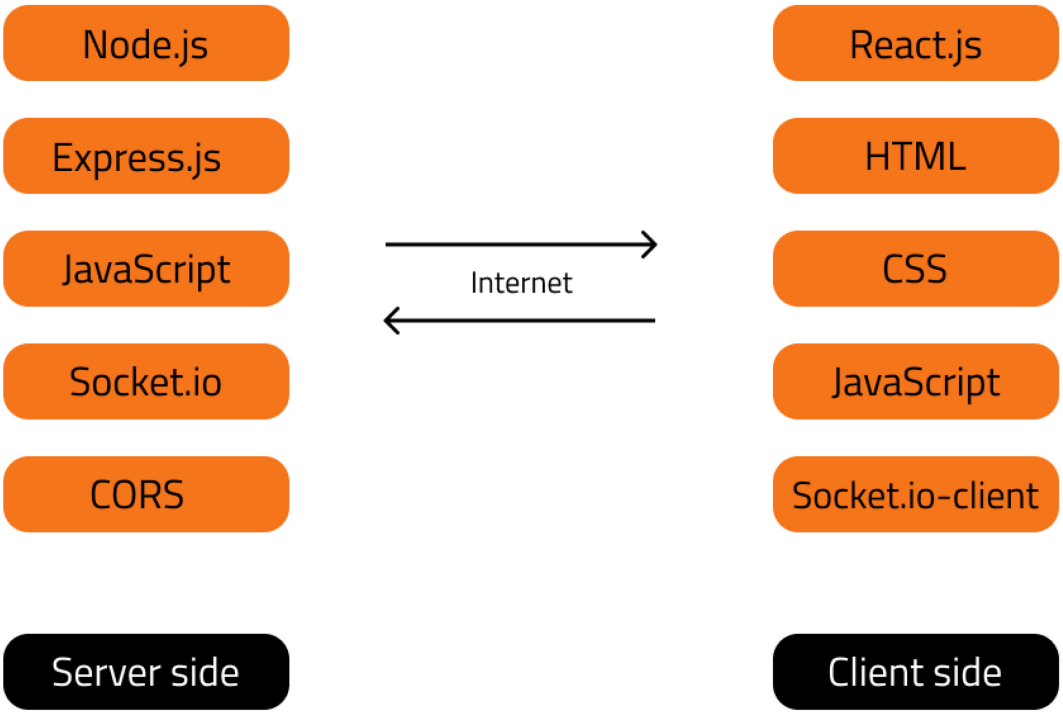


Figure 16: Technology stack of the system.

Figure 16 displays our final technology stack. We chose to stick with programming languages and frameworks we were already familiar with in order to avoid having to start from scratch with new ones. On this basis, we decided to use JavaScript as the programming language and Node.js as the framework for the server-side. It is built up by an express server that uses Socket.io and CORS[29] to communicate with the client-side. CORS is a way of sending requests to other domains, as the server has a different domain than the application. For the client-side, we decided to use React as the framework, with JavaScript, HTML, and CSS, together with Socket.IO-client which is the client-side library for Socket.IO. React was chosen primarily because we have much more experience with it, over other similar frameworks.

5.4 Video quality

From the previous in-depth project, we researched different codecs used to encode and decode the video stream. These codecs varied in their ability to encode and decode efficiently, and their ability to preserve the quality. The codecs have a part to play regarding quality, which is

[29] Cross-Origin Resource Sharing

one of the functional requirements. However, as we discovered in the in-depth project, due to the structure and complexity of the WebRTC technology we did not have the time nor skill to implement a different codec than the WebRTC standard (Bergo, Smaaberg, Teppan, 2021a, s. 19).

5.5 Front-end

To ensure that the application is usable on a variety of screens and devices, and because the group focuses on web development rather than app development, a website will serve as the interactive component from which the application is accessed.

To gain a better understanding of WebRTC's functionality and all of its working parts, we studied some tutorials on how to create WebRTC applications. Using these as an inspiration, we had to develop an application that satisfied our project's requirements and functionalities the employer and informants needed.

5.5.1 Implementation

In the first development phase of this project, our priority was to make the WebRTC technology work by creating a peer connection using Socket.IO. We prioritized this because it is the foundation of the application and without it working, the rest of the application would fall apart. The socket.js file is what allows the user to communicate with the back-end signaling server. It contains a context provider which handles the socket calls from the back-end. It also contains all the functions needed to create a peer connection. By having all communication between the signaling server and the front-end in a context provider, it makes it easy to access the values from all the components in the React app. "Context provides a way to pass data through the component tree without having to pass props down manually at every level" (ReactJS, n.d.). This makes all the information easy to access from all the components in the front-end.

As mentioned in chapter 5.2.2, STUN and TURN servers are needed to bypass the firewalls and allow data to be sent directly to the peer. We created STUN and TURN servers through the free service called Xirsys. This was, as mentioned in chapter 5.3, to avoid using public servers which could have a lot of traffic and negatively affect the performance. The

configuration for these servers is also located in the socket.js file and is used in the creation of the peer connection.

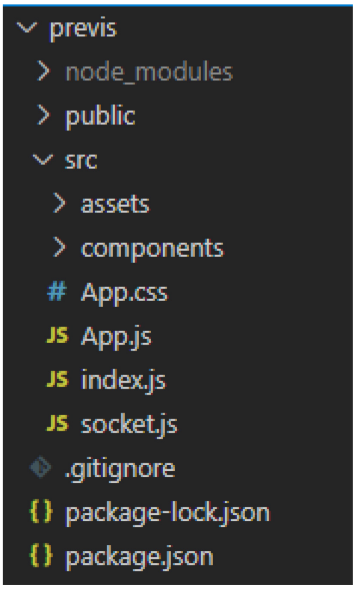


Figure 17: Screenshot of front-end folder structure.

Seen in figure 17 is the structure of the front-end. The public folder contains the index.html file, which is the base skeleton of the website, and where the index.js file is rendered. The index.js file contains the context provider wrapped around the app component. The App.js file contains the app component and this is where the routing is located, to render the different components based on the URL[30] route. The App.css contains all the CSS styling for all the components. The socket.js file contains all the socket listeners, functions for establishing the peer connection, and functions for getting the user video and audio. The package files are related to React and contain information on which modules are used, which react version it is, and metadata about the application.

[30] Uniform Resource Locators

5.5.2 Components

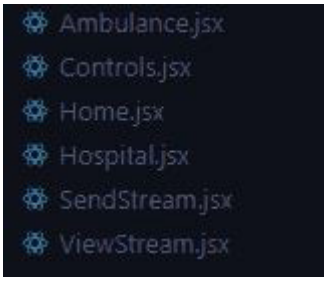


Figure 18: Screenshot of the React components.

Figure 18 displays all the components from the front-end. SendStream.jsx is the component for the ambulance side and ViewStream.jsx is the component for the hospital side. The Controls.jsx component contains the controls for both sides and the displayed controls depend on whether you are on the sending side or the receiving side, which is decided by a state value. The Ambulance.jsx and Hospital.jsx are used to combine the controls with the SendStream and ViewStream components. More importantly, they also use the use effect hook to run the starting functions for both sides, which finds the cameras and sets the correct state depending on whether the user are the ambulance or the hospital. Finally, the Home.jsx component contains the landing page which allows the user to select between ambulance and hospital. In a real-world scenario, this would be static from the different sides, but since we made an application that should work from both sides we thought this was a good temporary solution.

Use effect is a react hook that lets you perform side effects in function components (ReactJS, n.d.). The use effect hook is a function that by default runs after every render, but it is customizable to only run on specific terms. We used it in the rendering of the hospital and ambulance components, by creating the video elements inside the use effect function, and triggering the re-render if a few selected state values are updated.

We needed three state values on the ambulance side causing a re-render. The first is to state that the cameras are found and ready, which will create the video elements from the cameras and display them on the page. The second is to state if the user wishes to share their screen to simulate e.g., ultrasound, which again will create a new video element to display the ultrasound. The third and final state on the ambulance side is to signal when the hospital is

connected, which will trigger the use effect function again, and replace the previous video elements with the shared screen from the hospital side.

On the hospital side, we only needed one state, because this side is not sending any video of themselves, only a stream of their screen which is selected on start-up. We did not think it was necessary to preview this shared screen, and thus there was no need for a re-render. Therefore, the only state is for signaling that the ambulance is connected, which will trigger the function and create the video elements to display each video from the ambulance.

We used Google Chrome as the chosen browser for development and testing. This choice was later deemed important as some functionality regarding the code worked in some browsers, but not others. We therefore chose to primarily develop in Chrome, as this is per April 2022 the most widely used browser on a world basis, with over 64 % of market share (Statcounter, 2022). The main functionality is also tested to work in Mozilla Firefox, Opera, and Edge.

5.6 Back-end

As mentioned previously, the peer connection is what the whole system is based on and the back-end plays a crucial role in the establishment of this connection. The back end server is what allows the peers to find and connect to each other, and are therefore very important.

As mentioned in chapter 5.5, the first development phase involved making the peer connection. In order for the front-end to connect the peers, we had to create the back-end signaling server as a way for the peers to exchange information. We started by making the server file called index.js and initialized Node.js. Then we installed all the dependencies we needed for the server, including express, CORS, and Socket.io.

The next thing we had to make was all the socket listeners needed. The most important are the listeners for the initial connection and the messages being sent from the front-end. In addition we needed a listener for creating the room for the selected callee, to simulate the different locations the ambulance can call to. Lastly, we added a listener to send the name of the ambulance to the receiver, in order for them to know who they are talking to.

The socket technology makes it easy to listen for messages from the front-end and send them to the clients in the room. The back-end makes sure the peer connection is created by sending

an offer in a message through the socket room to the other client. The receiving client sends an answer in the same way, thus creating a successful peer connection.
During the first stages of development and testing, we ran the server locally on our machines. We did this because it was easier to test and troubleshoot the functionality of the back-end when it was running on localhost. When these stages were completed and all the functions worked as we wanted, we deployed the server to Heroku[31] to better simulate the real-world use cases. We were familiar with Heroku from previous projects and chose this because it is free and we know it well. This also gave us a more realistic scenario of connection times.

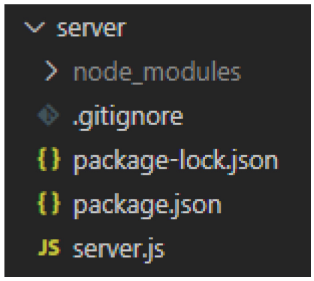


Figure 19: Screenshot of back-end folder structure.

Figure 19 shows the folder structure of the back-end. It consists of the main server folder with the server.js file serving as the signaling server. The package.json and package-lock.json files are automatically created when creating a Node project. The files manage the scripts for deployment and make sure the right dependencies are installed before deploying. The node_modules folder contains the actual dependency files that are used in the server.js file and are for running the server locally. The .gitignore file removes the node_modules folder before pushing the files to GitHub to save space, as it is of considerable size. On deployment, Heroku will automatically install the dependencies and run the start script from the package.json file, which starts the server.

Although the back-end of this system is fairly simple, it is the most important part of the system, because without it there would be no way for the peers to communicate with each other.

[31] Heroku is a platform as a service (PaaS), that allows quick and simple deployment of web development projects.

5.7 Responsiveness

CSS, or “Cascading style sheet” is a simple mechanism for adding style (e.g., fonts, colors, and layout) and spacing to web documents. Using CSS to design in a responsive way means adapting the style to better fit different screen sizes. If adequate responsive design is achieved, it will improve the user experience across more screens, for instance tablets, tv’s and phones. Information gathered from the employer gave us a good idea of what devices would be most used to interact with the system. In the ambulances, touch sensitive 10 inch tablets will be installed, and in hospitals or emergency rooms larger tv screens at 55 inches and 1080p resolution (Stokke, 2021). These devices worked as the foundation for the responsive design in the application.

To achieve responsiveness that satisfies the displays needs, the flexbox layout model is utilized for positioning most of the major components in the application. This is a one-dimensional layout model, and together with the two-dimensional grid layout model represents the two most popular models for CSS layout. One good example of why the flexbox model proved valuable to our project, is its ability to position the video streams logically, and scale well independently of how many video streams the user has added. This is a consideration needed because the amount of cameras and equipment varies based on the needs of the paramedics and doctors. The employer had not decided on a set amount of cameras to use in the final solution, so we made the decision to make the system adapt to this.

5.8 Software testing

“Software testing is the process of evaluating and verifying that a software product or application does what it is supposed to do. The benefits of testing include preventing bugs, reducing development costs and improving performance” (IBM, n.d.). While development costs are not relevant for this project, the rest of the benefits are. To make sure the application works as expected, it was tested in conditions designed to emulate its real life use cases. The testing phase occured after the code was implemented and laid the groundwork for identifying and correcting remaining errors. Due to our infrequent meetings with project participants towards the end of the development phase, we conducted system tests to determine which functionalities needed to be improved.

5.8.1 Functional testing

Functional testing is to check functions by emulating business scenarios, and is based on the functional requirements (IBM, n.d.). In our case, business scenarios were translated to specific use case scenarios, meaning scenarios that emulate how the ambulance and hospital will use the system to interact.

A feature that proved somewhat difficult to implement was the requested back button when a user has pressed one of the video feeds, to only view that specific camera angle or multi-monitor in fullscreen. To add CSS on top of a fullscreen HTML element, a pseudo class named :fullscreen seemed like the obvious choice, as this is the purpose of the pseudo selector (MDN, 2022). But when implementing the feature it proved to be very browser specific, as the feature could work in one browser, but not another. The solution was found in different vendor prefixes, which is made to cover a better range of browsers (CSS-TRICKS, 2021) . The prefixes :-moz-full-screen and :-webkit-full-screen turned out to be a combination of selectors that covered all the browsers we tested in. The button is hidden when the user is not in fullscreen mode, by using the CSS selector display: none. This issue was fixed in the first round of testing.

The main part of functional testing was between members of our group, one simulating the ambulance and another the hospital. Laptops were connected to 4G mobile routers to better simulate the real-world ambulance conditions, which had a visible effect on video quality. We used personal web cameras to simulate the ambulance setup, which has a much lower quality than the 4K cameras that will be used in the ambulance.

5.8.2 Usability testing

Usability testing is used to determine how well a user can complete a task using a web application (IBM, n.d.) One issue discovered during the usability testing was that in order for the ambulance to see what the hospital sees, the hospital user must share the browser tab that is currently being used to view the ambulance videos. We believe this may be confusing for some users and may result in an incorrect tab or window selection, resulting in miscommunication and confusion. In order to avoid the browser popup for selecting the correct browser tab, an attempt was made to select the current tab in the code, so that this

would be handled automatically in the background. However, for user privacy reasons, media cannot be shared without consent and input from the user (MDN, 2022).

5.9 Summary of development phase

After comparing technologies and developing the working application, the application was tested and improved. By relying on some technologies we have previous knowledge of, a skeleton of the application could quickly be implemented. But as none of us had worked with video streaming previously, except when making the small prototype in the in-depth project, it took some time to understand and implement the Socket.IO library and all of the working parts of the WebRTC technology. The final application will be presented in the next chapter, followed by a test guide.

6 Final application

After design, development, and testing, an application has been made to simplify and effectivize the interaction between paramedics and doctors or specialists. By reducing the number of buttons to push and simplifying the layout, paramedics will be able to use the application more quickly and easily in critical situations. Allowing the doctors or specialists to mirror their screen can make interaction and communication clearer, as the ambulance workers can see what areas of a patient the doctor means.

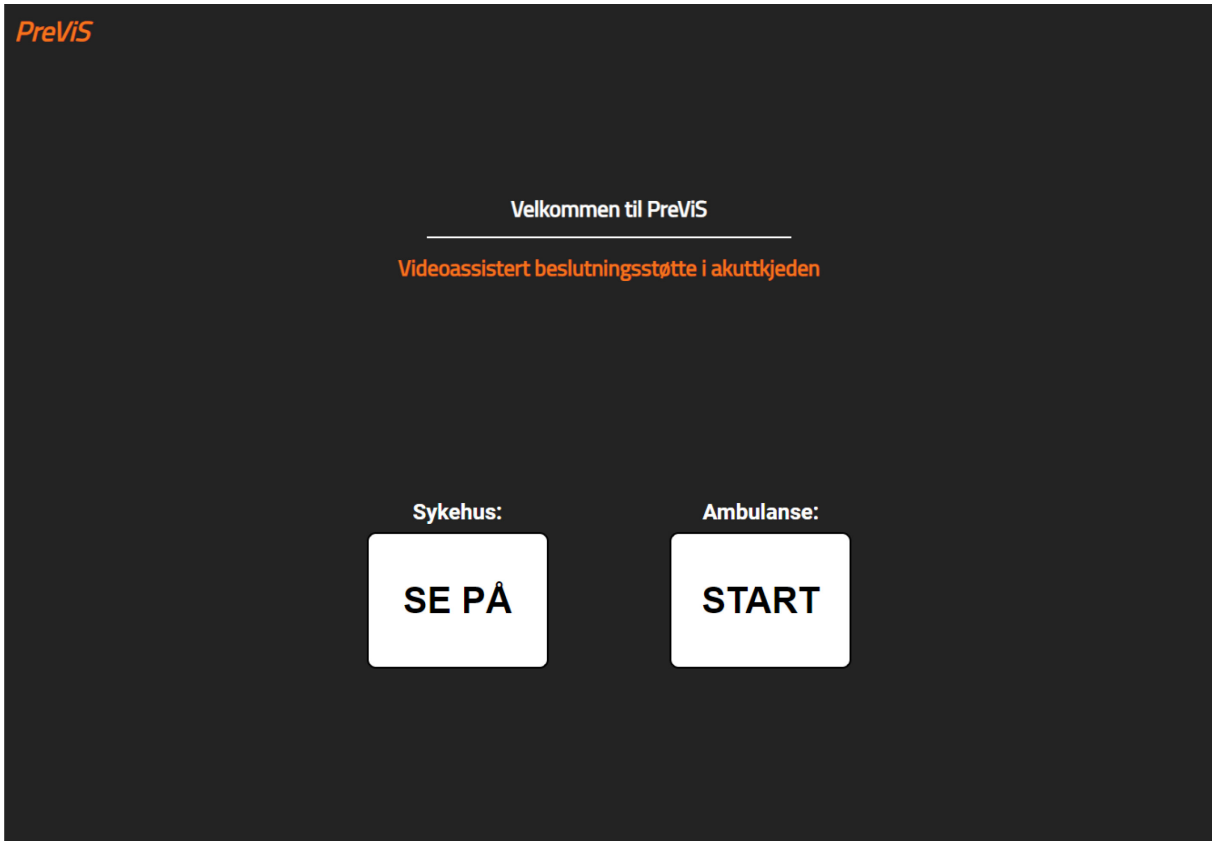


Figure 20: PreViS landing page.

Figure 20 illustrates the completed version's landing page, emphasizing only the elements that are absolutely necessary for the system and omitting everything else. The landing page features two buttons, one for the ambulance and the sending of the stream, and another for the hospital that is receiving the stream. These start their respective sides and their features, as there are different needs and components for them both.

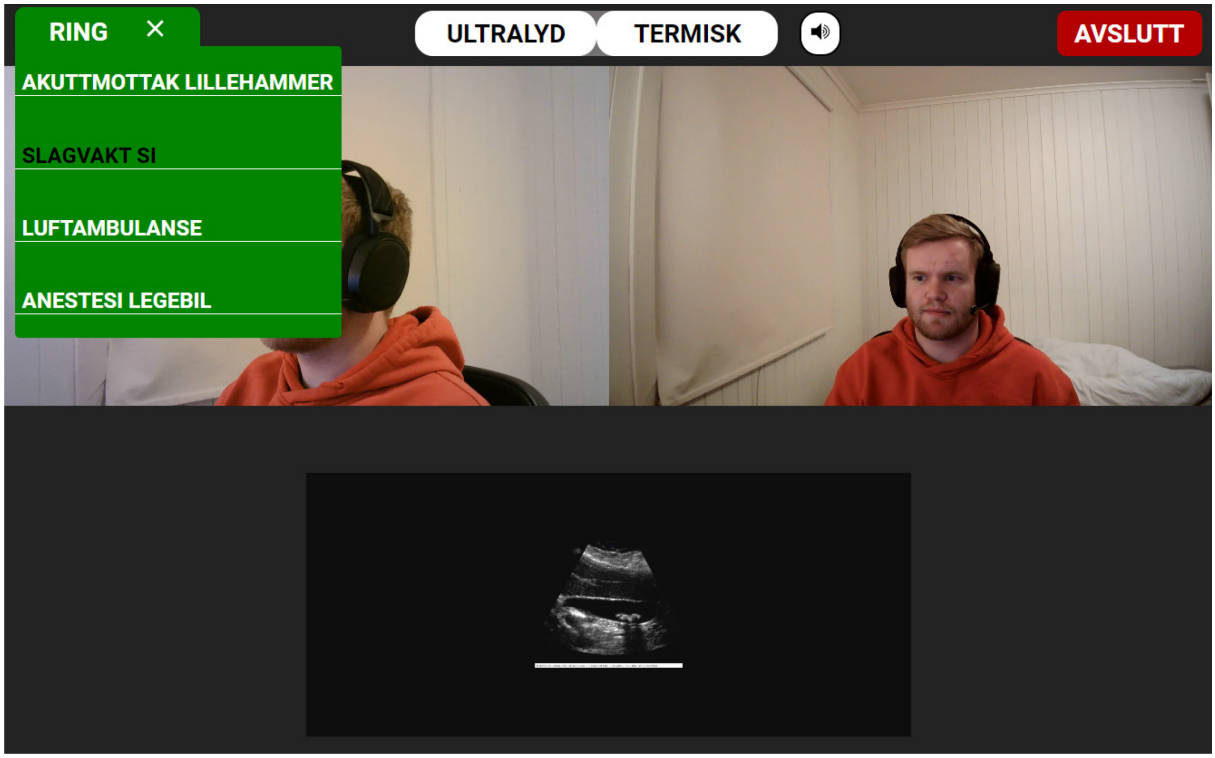


Figure 21: Screenshot of PreViS ambulance side before calling.

Figure 21 shows the ambulance view before connection, which is a preview of the cameras and possibilities of adding ultrasound or thermal camera. The call and end button are color-coded to easily understand their respective functions. When the call button is clicked, a little selection of callees appears. When pressing one of them, a room is created with the same name, and this alerts the hospital that an ambulance has started a session.

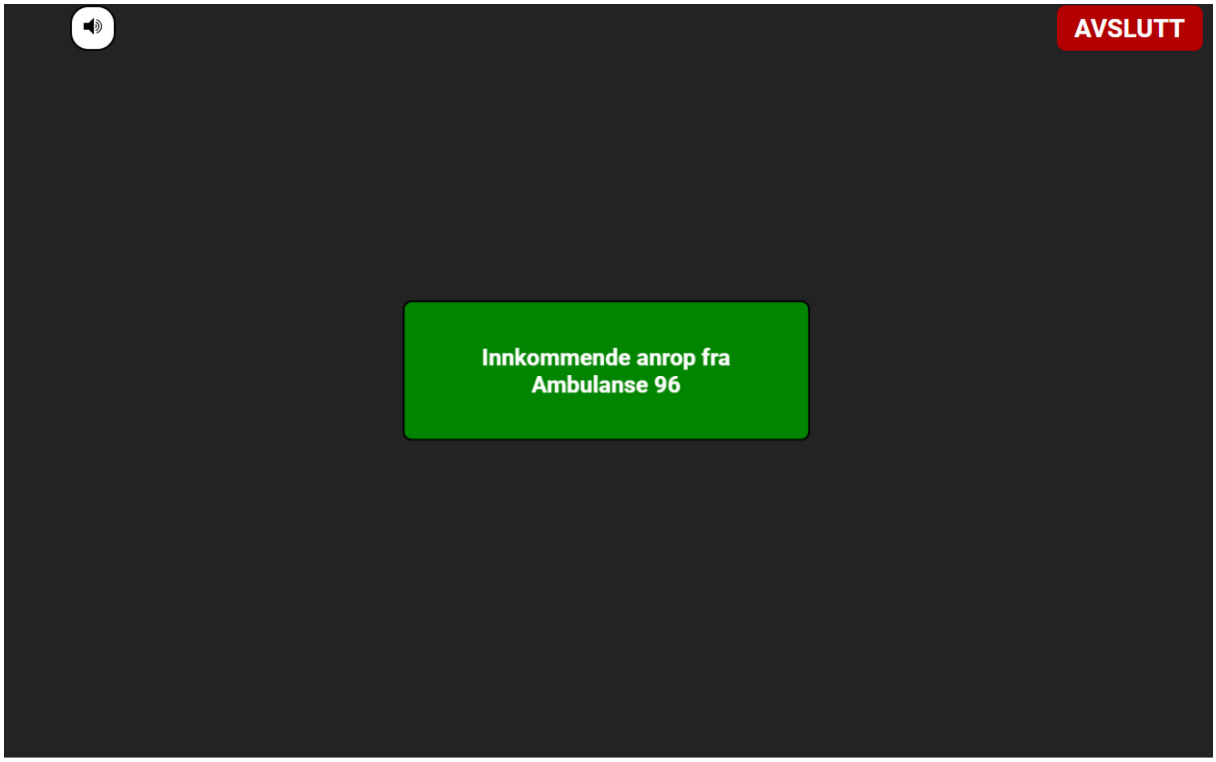


Figure 22: Screenshot of PreViS hospital side on incoming call.

Figure 22 shows the application from the hospital side, which is receiving a call from the ambulance. The hospital answers the call by clicking the button and the peers will connect and exchange streams. We decided to add a ringtone to the application, that plays on repeat until a user answers the call. This gives a clearer way of alerting doctors or specialists that an ambulance is calling.

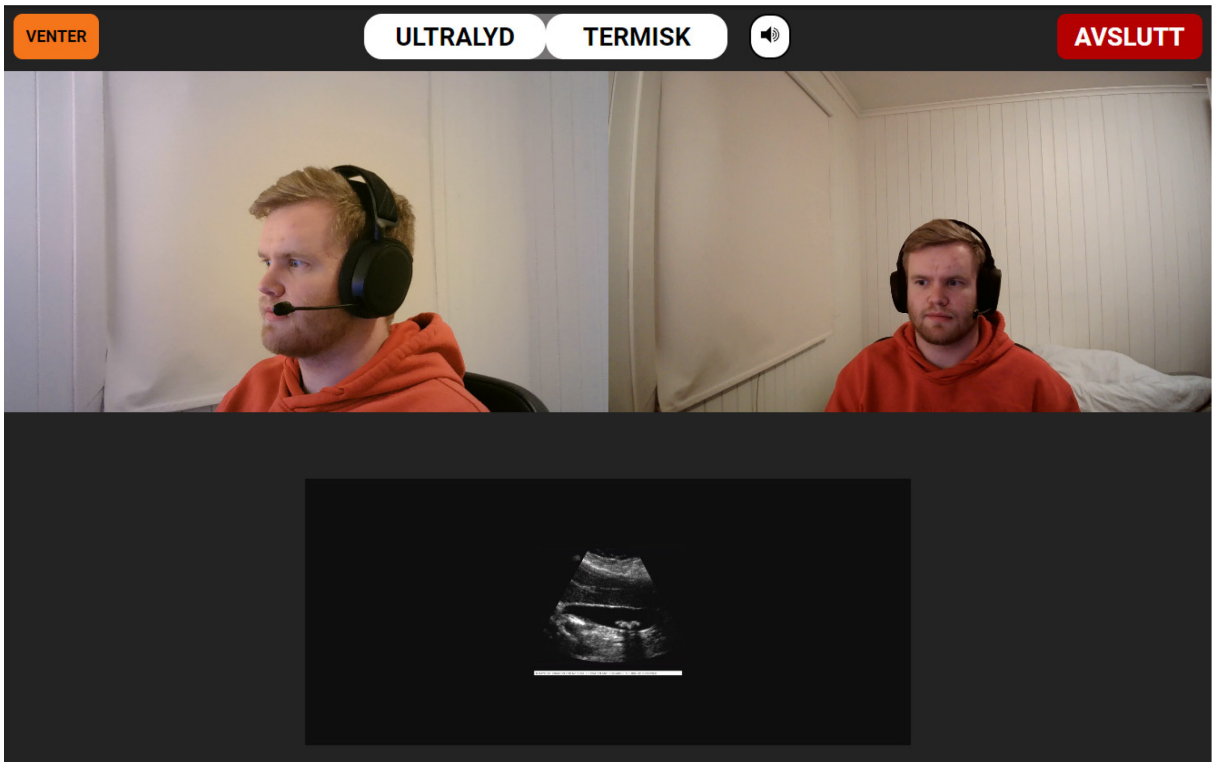


Figure 23: Screenshot of PreViS ambulance side waiting on answer.

After calling the selected hospital or emergency room from the ambulance, the call button is changed to a pending status, as seen in figure 23, until the receiving side answers the call.

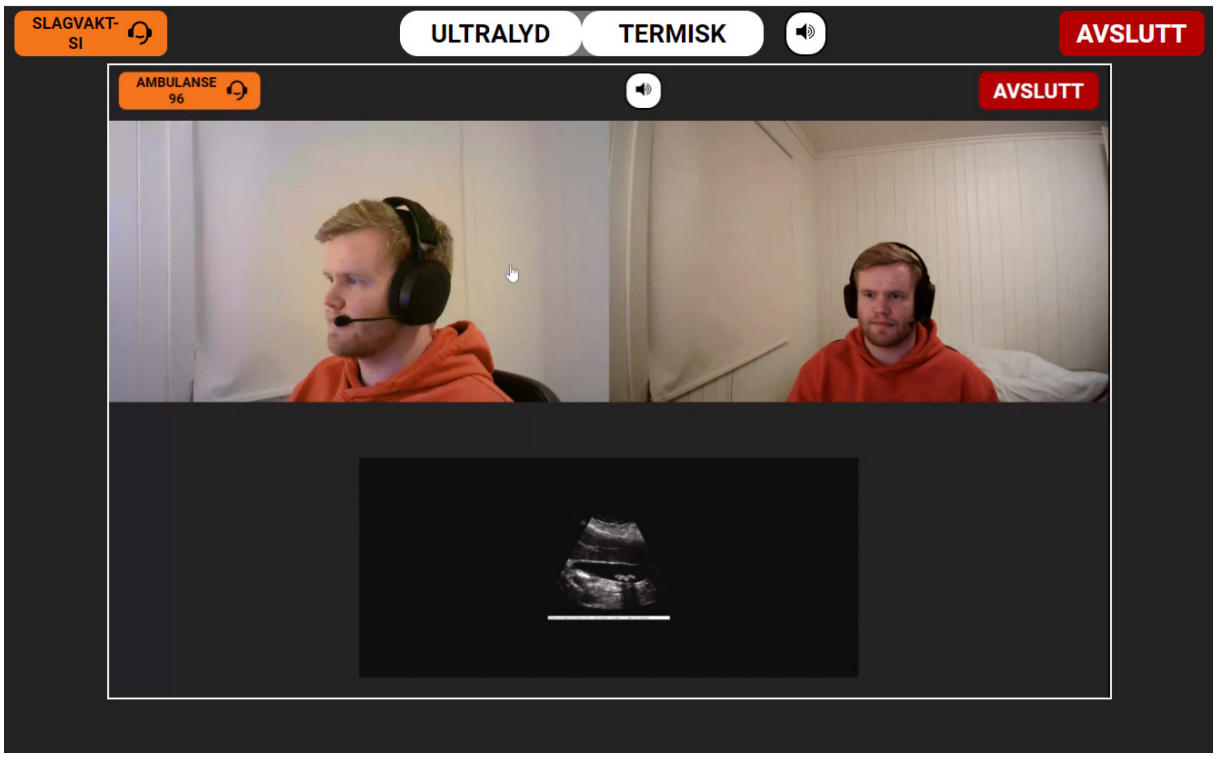


Figure 24: Screenshot of PreViS ambulance side after connection.

After connecting, the ambulance receives a video of the hospital's screen, shown in figure 24, so the ambulance can see what the hospital is looking at. This removes any doubt on which picture is being discussed, which can be critical in medical situations.

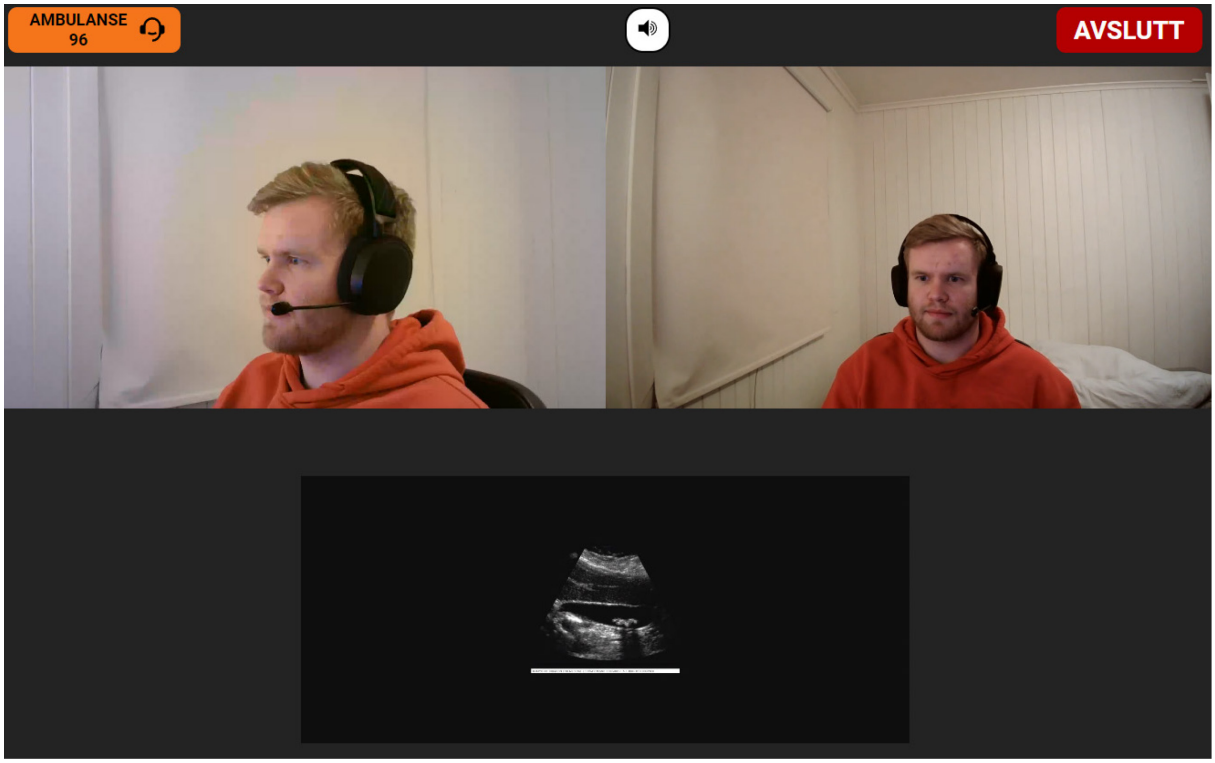


Figure 25: Screenshot of PreViS hospital side after connection.

When the ambulance is connected, the hospital will see their videos as seen in figure 25. The hospital can then choose which video to get a better look at, by clicking them and entering full screen. All the actions the hospital does will also be mirrored in the ambulance.

6.1 Test guide

The system requires some steps in the right order, to work as intended. Unlike traditional two-way video systems, PreViS depends on screen sharing to capture what the doctors or specialists are viewing in the hospital. Depending on the browser used, the users might encounter different screen share options than what is described in this test guide. As mentioned in chapter 5, we chose to use the most popular browser to conduct most of the testing, which as of April 2022 is Google Chrome, with over 64% browser market share on a worldwide basis (Statcounter, 2022). The application is also adapted to work in Mozilla

Firefox, Opera, and Edge. Safari was not a focus as the employer stated they would not use IOS devices in their system.

The intended use flow is that the hospital user connects first, and has their page running to wait for incoming calls from ambulances that suddenly need assistance. The hospital user enters the landing page and presses *SE PÅ*, to enter the hospital page. Here they are prompted with a popup to share content, and choosing the right content is important. The idea is to share the tab you are currently viewing the hospital page in. This is demonstrated below in figure 26, where the user has to select the *chrome-fane* option at the top right and select the tab they currently are in. This needs to be done to send the correct screen share back to the ambulance. The preview window under *chrome-fane* can be used to check if the correct browser tab is selected. As mentioned above this removes any doubt on which picture is being discussed.

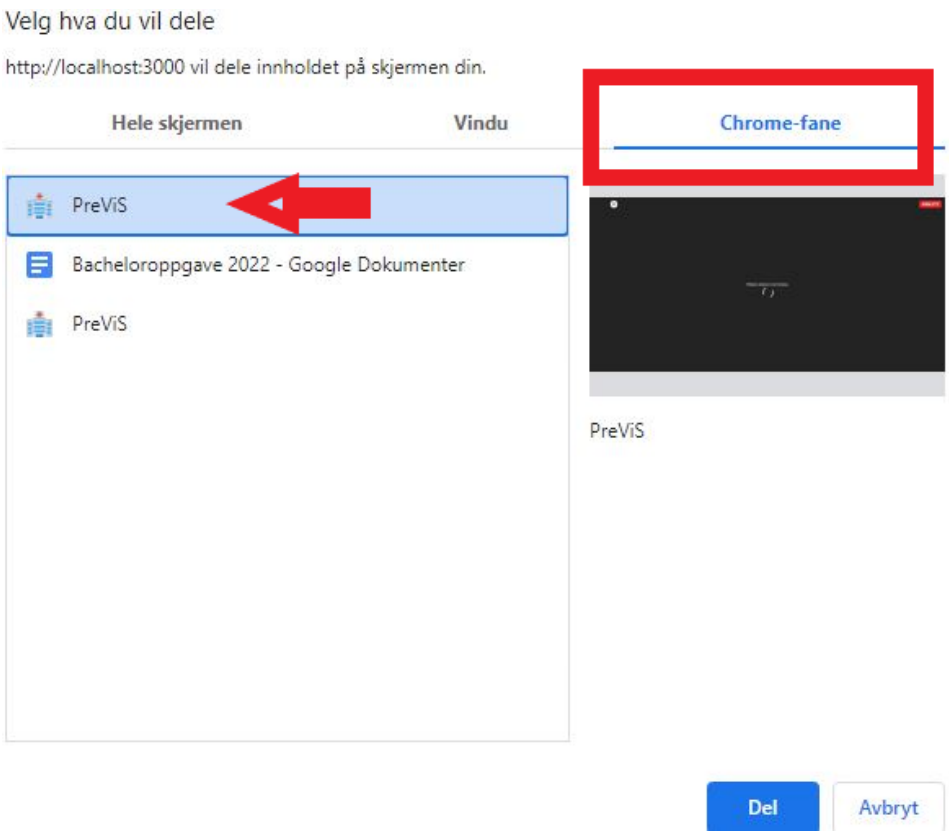


Figure 26: Screenshot of tab selection on hospital side.

The user will be prompted to allow the browser to use their microphone and camera, as shown in figure 27 below. The user needs to give permission in order for the code to get the user media.

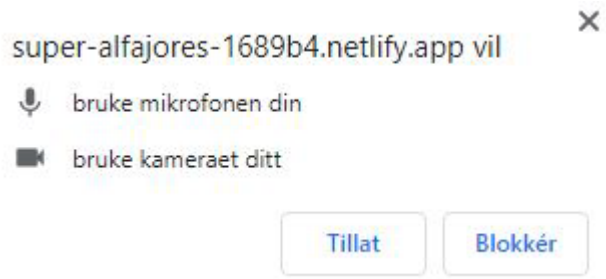


Figure 27: Screenshot of chrome asking for permission for camera and microphone.

When the hospital page is up and running, it is ready to receive calls from an ambulance. The user simulating the ambulance side will press *START* on the landing page, to enter the ambulance page. Here the user is also prompted to allow the browser to use cameras and the microphone and needs to be allowed to start the application. The optimal strategy is to have two web cameras connected to your device, to better simulate the two view angles in the ambulance. The user can then choose to simulate a screen share for ultrasound, thermal camera, or multi-monitor, or proceed to call without. Calling, or entering one of the Socket rooms, is done by selecting *RING* in the top left corner and choosing one of the rooms. The page will then wait in a pending state until someone using the hospital interface answers the call. A call is now in progress and a session can run as long as needed, either side can always stop the call by pressing *AVSLUTT* in the top right corner, which ends the session for both sides. This guide describes how the application is intended to be used by paramedics and doctors or specialists.

The application is deployed and can be tested through this link:
https://capable-klepon-bea761.netlify.app

7 Result

This chapter will examine the outcome of our bachelor project work. The results of user testing, requirement specification, and our applications perceived efficiency will be presented.

7.1 Results from user testing

We chose the System Usability Scale (SUS) to user test the system. We decided to do the SUS test with the project participants who were already participating, such as our employer and two paramedics, who are test pilots for the PreViS project. This was done to receive feedback from both parties who will be using and developing the application. All parties concerned took part in the user testing, which was done on Microsoft Teams. The Figma prototype and the final application ended up having several differences, in terms of usability and intuitivity. To determine if there was improvements from the Figma prototype to the developed application, we calculated the average scores for both rounds of user testing. This made it easier for us to distinguish between the prospective prototype, system modifications, and actual enhancements.

7.1.1 SUS scores

We let the participants test the prototype before we started with the SUS questionnaire. Each participant was given access to Figma, where the high-fidelity prototype were available. The participants were given no restraints and were free to test out the prototype how they wanted. In the second round of testing, the participants were given a demo of how the final application works and were able to test it themselves. We completed two rounds of user testing. The first phase consisted of user testing based on the Figma prototype, while the second round consisted of user testing based on the final application.

The total score of the SUS test for participant 1 after the first round of user testing was 85, which is above average. The SUS test obtained a final score of 95 after the second round of user testing with the first participant.

The SUS test attained a total score of 82.5 during the first round of user testing with participant 2. After the second round of user testing, the final score was 92.5.

The total score of the SUS test after the initial round of user testing with participant 3 was 85. Following the second round of user testing, the final score was 92.5.

7.1.2 Summary of user testing

The average score for the Figma prototype was 84.1 points. According to the table created by Bangor, Kortum and Miller (2008) in chapter 4.3.5.2, the first round of user testing ended up scoring between good and excellent with 84.1 points on the scale.
The average score for the second round was 93.3 points. With 93.3 points, the second round of user testing falls between excellent and best imaginable. The scores can be visualized in figure 28. The three scores on the left represent round one of testing, and the three to the right represent round two.

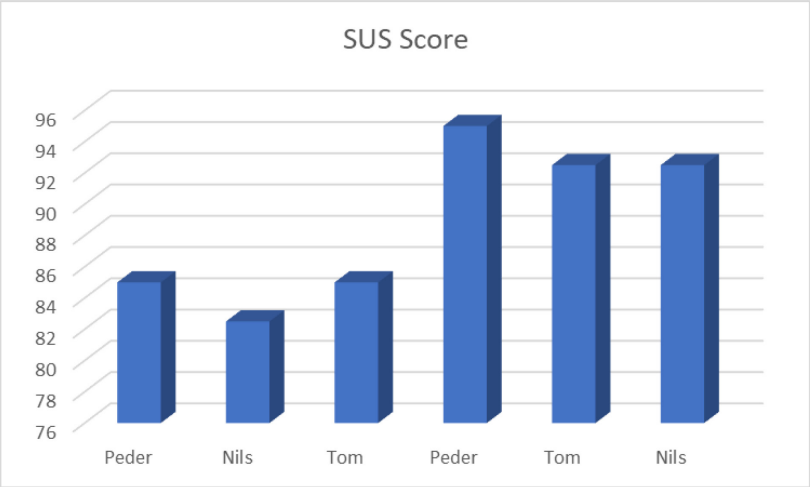


Figure 28: Diagram of the scores for each participant.

Based on this, we recognize that the changes made were indeed improvements, resulting in a considerably higher usability score for the final application than the Figma prototype.

7.2 Requirement specification

To be able to positively answer the thesis statement, and improve on the PreViS team's prototype, we needed to assess whether the functional and non-functional requirements had been met. These were presented in chapter 2.8, and we designed and developed the application with these in mind. The requirements we have qualitative or quantitative data from will be presented here, and the other will be explained in discussion.

7.2.1 Accessible design

To assess whether this requirement is met, we used the Wave tool to check if color and contrast comply with WCAG guidelines. Figure 29 displays the results of checking contrast on the most used elements in the application. These elements are the orange caller id box and the back button, the green call button, and the red end button.

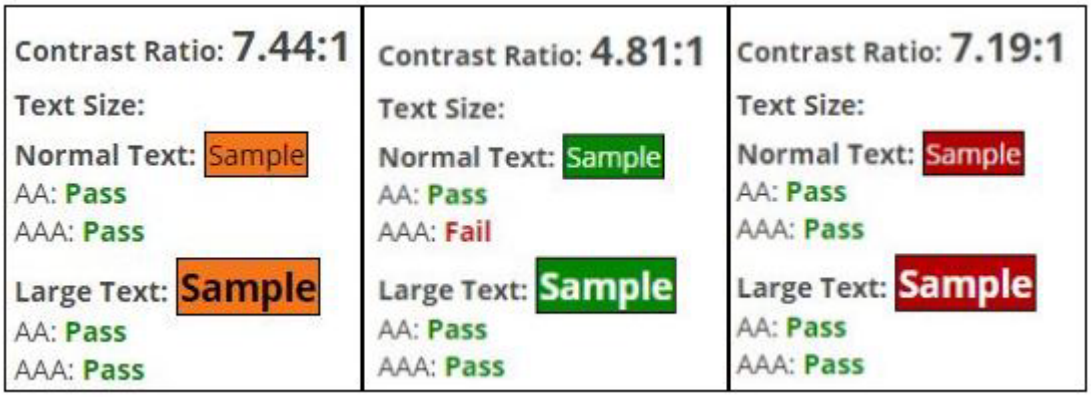


Figure 29: Wave evaluation of contrast ratio.

The only value that fails is white text on our green call button, a fail on AAA normal text. This is not applicable to this project, as we have made sure text in these button elements is classified as large text (WebAIM, n.d.). This shows that all the interactive parts in our application comply with WCAG guidelines.

7.3 Efficiency

Fitt's law was previously introduced in chapter 4.1.3, to give us a readable number on whether our application is more efficient in use. By calculating movement times on both our application and the PreViS teams prototype, an indication can be made to find out if our is

faster, helping answer the problem statement. The equation for Fitt's law is expressed as shown below:

$$MT = a + b\,[\mathrm{Log2}(2D/W)]$$

- Where MT is the average time required to complete the movement.
- *a* and *b* are constants that depend on the type of input device used, e.g mouse, finger on touch screen etc.
- D is the distance between the starting point and the target.
- W is the width of the target object.

Both prototypes were run on the same screen size and format, to give equal test conditions. A Codepen calculator[32] was used to calculate the scores, and constants *a* and *b* were set to the default values. The starting point was in our case set in the middle of the screen, as we assumed users would regularly press the different video feeds to enter fullscreen, as well as use the control buttons. Screen sizes was also set to exactly the same dimensions, to give equal references for measuring distance and width.

The PreViS team's prototype had a distance of 14.2 cm from starting point to target, and a target width of 2.3 cm. This gives a movement time of 1.96 seconds. Our application had a distance of 14.5 cm, and target width of 3.7 cm. This gives us a movement time of 1.7 seconds, just over a quarter of a second faster. Even though this does not sound like a lot, it demonstrates that increasing button size and organizing them better can somewhat improve efficiency. Throughout a call, the doctor viewing might press a multitude of buttons, and the quarter of a second can add up to more time. Below in figures 30 and 31 are examples showing how the distance and width were measured, in this case, the measurement of the end button.

[32] Codepen calculator

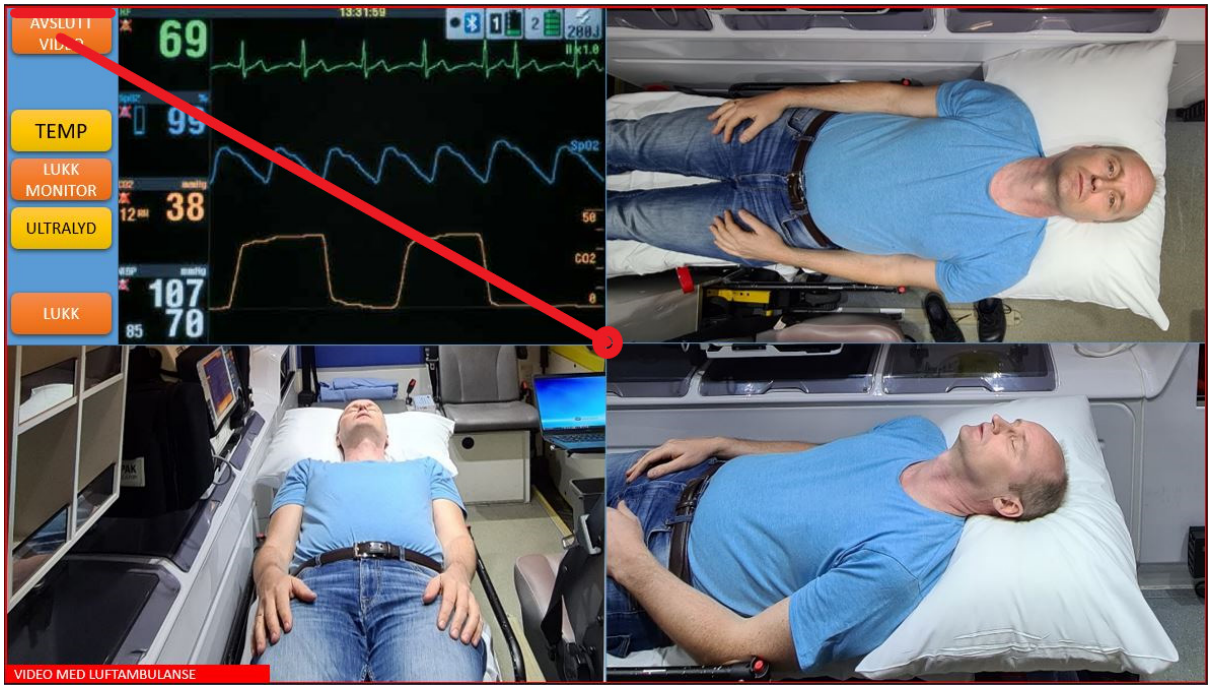


Figure 30: Distance and width measured in the original PreViS prototype.

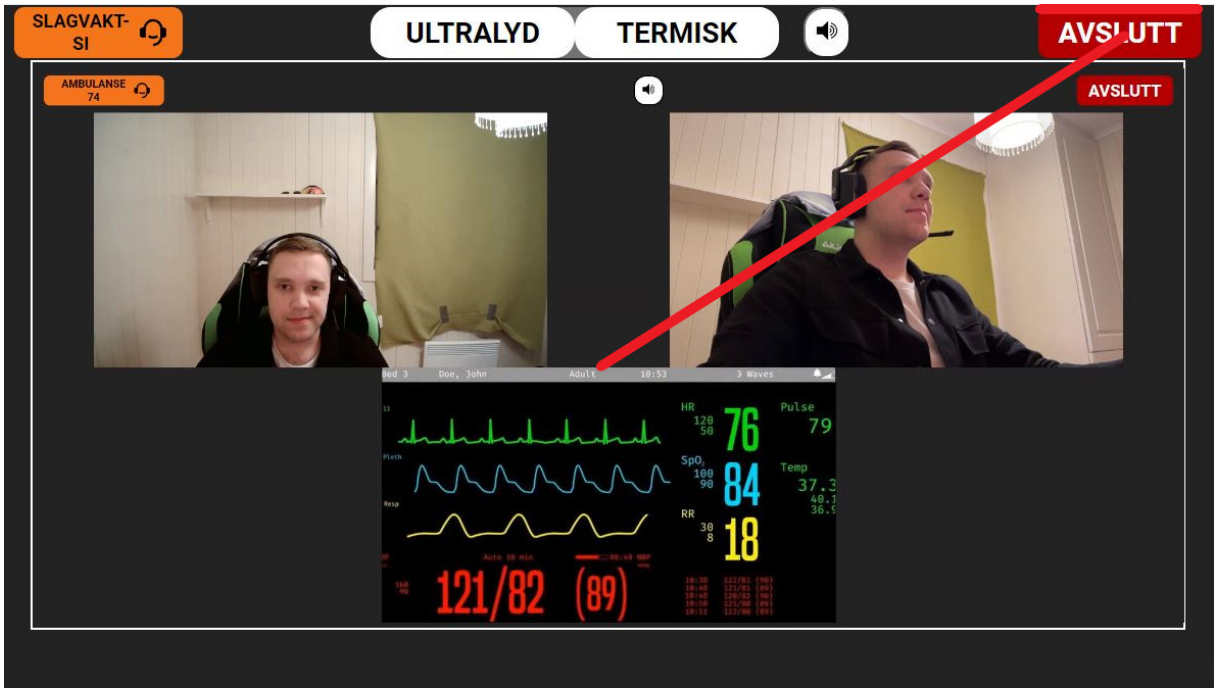


Figure 31: Distance and width measured in our final application.

8 Discussion

In this chapter we will discuss whether the problem statement has been answered. By looking at the theory we have used and the final application developed, the thesis problem statement can be assessed as solved, partially solved, or not solved. The thesis problem statement was:

How can we make the interaction between paramedics and specialists more intuitive and efficient through a web application?

In order to assess whether the problem statement has been answered, we need to rely on the results from chapter 7. This includes data we have gathered during user testing, and tangible data that can be gathered from the prototypes. The following subchapters will discuss whether they prove the problem statement, and will be summarized in the conclusion.

8.1 Answering the problem statement

8.1.1 Efficiency

When planning the project, we had a large-scale test in mind involving our final application, and to test and compare it against the PreViS prototype. When this fell through, we looked at other methods of evaluating the efficiency of our application. Fitt's law allowed us to get indications of how much time is used to press different buttons. This gave us some answers as presented in chapter 7.3, but we should have used Fitt's law from the beginning of the design phase, allowing us to have this in mind when designing all components and buttons. This could have helped improve the efficiency of our application even more. The results for the movement time still show some improvement, and in terms of time used to locate and press the buttons, our application is more effective.

When comparing the PreViS team's prototype with ours, the choices of layout and information architecture create some differences regarding efficiency that is important to highlight. In the PreViS team's prototype, the controls are located at the top left of the interface, on top of one of the video feeds from the ambulance. This control menu has to be closed to properly see the full video feed, as opposed to our control menu, which is placed permanently on top of the interface. Therefore, our application eliminates button presses to

view specific video feeds, and at the same time provides more space between the control buttons, reducing the chance of pressing the wrong ones. These changes make our application more efficient in use than the PreViS prototype.

8.1.2 Intuitiveness

Our application has some areas where user errors can be made, and it can be hard for the user to realize and undo their mistakes. Depending on the user, there may be sections where not everything is intuitive or understandable to those who have never used it before. An example of this is the process of selecting the correct browser tab to share. However, we could not test the application in the ambulance and had to develop it to make it work with web cameras and in browsers. By testing it in the ambulance with the right cameras, we could have prevented some of these usability flaws. As stated in chapter 4, most of the influence comes from interviews with the test users of the current system. Given that the system is closed and its primary focus is to support paramedics in their work, the decision to follow their input as much as possible was made early in the project. We used principles learned in different subjects to add trusted design and development theories, where possible.

Some of the features implemented are features not currently in the original PreViS prototype, such as mirroring of the hospital screen back to the ambulance. This will help the paramedics focus on the patient and at the same time receive feedback from doctors, helping them cooperate to diagnose patients faster. If the hospital user is using a mouse pointer, this will also be mirrored back to the ambulance, allowing the doctor to point at specific areas they want to highlight, which can improve the speed of the assessment. This was also a functional requirement from the requirement specification, based on the requests from the PreViS team. We can therefore argue that our final application has better intuitiveness than the PreViS prototype because we have implemented the mirror-screening functionality.

8.1.3 Requirement specification

As explained in chapter 7, many of the requirements did not yield results that were displayable through data or charts, and they will now be discussed to find out whether or not they were improved or implemented, to help answer the thesis problem statement.

8.1.3.1 Functional requirements

Low latency

One of the functional requirements were low latency, which is crucial to provide near real-time communication between the users. We tried to measure the latency on our application by logging a timestamp when the videos were sent from the ambulance and again when the videos were received. However, the values we got from this method were much higher than the visible latency, meaning the outputs of the latency test were just above 3 seconds, but we could see on the shared screen from the hospital that the actual latency was much lower. This method was proven to be inaccurate, and we could not get an accurate readout on how low the latency was.

Even though we could not measure the latency, it is clear that WebRTC in general is much better than its competitors with a latency of 1 second or less, as stated in chapter 5.2.1. This means we can assume our application has about the same latency, as it uses this technology for streaming the videos. However, in our final application, there are some factors that will affect the latency, such as the stability and speed of the internet connection. This can make the latency higher, but these factors are out of our control. Based on this and the fact that the PreViS prototype also uses WebRTC as a streaming protocol, we can not assess this requirement as solved, since we have not improved this aspect in any known way.

High quality feed

In the in-depth project, we researched different codecs that could be implemented in WebRTC streaming and found alternatives that would increase quality by using better compression techniques. The problem was, as mentioned in chapter 5.4, that this could not easily be changed, and required to build a custom WebRTC streaming protocol to add preferred codecs. This means we have no control over which codec is being used by WebRTC, and how good or bad the video quality is. However, we specified resolution constraints when we retrieved the videos from the user, meaning the WebRTC technology will always try to maintain the specified resolution as long as the internet speed allows it.

It is clear that good video quality overall can provide better grounds for doctors to spot details that they might miss if the quality is low. Being limited by the complexity of WebRTC, we did what we could to improve the quality. The quality of the streams mainly depends on the

codec's ability to efficiently encode and decode, as mentioned in chapter 5.4. This is controlled by internal WebRTC code which we were not able to change. On this basis, we consider this requirement as not solved, because our solution uses the default WebRTC codec, meaning we were not able to improve the quality.

Remote control from hospital

In the PreViS teams prototype, both ambulance and hospital have the same interface, meaning they have not implemented the requested feature of mirroring back what doctors or specialists are doing. We implemented the feature by allowing the hospital interface to share the browser tab where they are viewing the video from the ambulance and sending this back to the ambulance. This means the ambulance will see exactly what the doctor or specialist sees, including the mouse cursor if a computer mouse is used. We therefore assess this requirement as solved.

8.1.3.2 Non-functional requirements

Simple information architecture

This requirement will be answered by feedback we received from one of the informants. When asked about the finished design, one of the informants replied with “Simple, and clear”, and “Nice that the patient is on the top and telemetry below.” (Bakken, 2022). The informant only had time for a short meeting, so no answer was given on whether our application’s information architecture was better or not. We still believe his comments on the final application can give us an indication, but not enough to assess as completely solved. This non-functional requirement is therefore assessed as partially solved.

Accessible design

As described in chapter 4.1.2 about color and contrast, and chapter 7.2 about requirement specification results, we chose colors that the informants had associations with, and that still fulfilled WCAG contrast ratios. It also mentioned that color associations can vary a lot depending on context, culture, and time. We have been aware of this implication and came to the decision that the combination of strong button colors, together with white descriptive text that passes all WCAG contrast ratios as described in chapter 7.2.1, gives more than sufficient readability even for users with color deficiencies. This was deemed important as even doctors can have forms of color blindness, as explained in chapter 4.1.2.1. This also aligns well with the wishes of the informants, meaning that this accessibility aspect matches appropriate

guidelines and insight from direct users of the application. Therefore, the accessibility has been improved overall, based on Wave results in chapter 7.2.1.

Error prevention and usability

As stated in chapter 2.5, the user groups in this project are paramedics, and doctors or specialists. Given this fact, the decision was taken to not optimize the solution regarding normal people outside of the healthcare industry. Based on this, if the user of the system is sufficiently trained, we believe the simple UI and low amount of buttons make it hard to commit errors when navigating the application. The different buttons are also positioned far enough from each other to avoid clicking the wrong button. Figure 21 in chapter 6 can be looked at for reference. For example, the start and end buttons are positioned on each side of the interface. However, there are improvements to be made regarding error prevention in the final application, as it has not been tested in the ambulance with their equipment and cameras. This will most likely create some problems and some errors might occur.

The results from the SUS user testing proved that our prototype improved usability from the design prototype to our final application. This is based on the scores we collected from the two rounds of user testing. The final score of 93.3, is in SUS terms a score between excellent and best imaginable. This shows that the finished application we have developed, has very good usability, given that the score comes from the user groups of the system. Based on this, and the fact that the buttons are placed far from each other to prevent errors, we assess this requirement as partially solved. This is because the usability is good, but there is still some room for improvement regarding error prevention when implementing the application with the ambulance's equipment and cameras.

8.2 Sustainability goal

As stated in chapter 1.3.1, we developed a question related to the United Nations' third sustainability goal. The research question stated:

How can the developed application contribute to improve good health and life quality?

One issue with proving this research question is the fact that our final application will not be implemented in prehospital service before the bachelor project is finished. The PreViS project is still in progress and will not be finished until September 2022. Since the project is

incomplete, it is challenging to formulate a response to our research question. To answer the question definitely, the project must be completed and utilized by the prehospital service, which is not possible at this time.

Because it has not yet been tested on patients, there are no statistics or data regarding the success of this study. This results in little to no information on how this technology aids the prehospital service and what benefits will be obtained following the completion of the project. Therefore we must determine if the sustainability objective has been attained based on the information provided in the official application report for the PreViS project.

There are none of the sub-goals that specifically mention prehospital health, however, the main health goal says “Ensure healthy lives and promote well-being for all at all ages” (UN, n.d.). We argue that if this project gets the results it claims when used in the real world, it *can* better ensure healthy lives and promote well-being for people at all ages. It can achieve this by providing better and more equal prehospital service to people, no matter where they live. Subsequently, it can also help the goal by reducing the number of admissions to hospitals from prehospital support, because more patients are treated where they live, either from emergency personnel or the primary health service. If the project never reaches implementation into active prehospital services, these goals will not be reached. The costs described in chapter 2.3 can also be lowered, if PreViS means fewer unnecessary admissions to the hospital, if the patient can be treated at home. Taking all this into account, we believe that the developed application can contribute to improve good health and life quality, if the project is realized.

8.3 Limitations

Through the project, we have had many meetings with both the client and project participants during the interview process and design phase. Toward the end of the project, we wished there had been more opportunities for meetings in order to establish a stronger foundation for testing and comparing the applications. Due to the infrequency of meetings in the last phases, we were forced to conduct the tests independently. This could have been avoided if we had planned better and given earlier notice to the innovation team. By setting dates earlier in the project time frame for user testing and comparison with the PreViS team’s prototype, we

would have had much better grounds to find quantitative data on whether our application has improved the existing one.

This would also have given us the possibility of testing the application in one of the PreViS team's ambulances, to see if it could have been implemented and worked in their setup. However, it is unlikely that our application would work seamlessly with the technology in the ambulance, as we have not been able to test any of their equipment.

8.4 Reflections

This project has been challenging in many ways for us. Many of the limitations in the project comes from the fact that this is the first application of its kind. Therefore, there are not many existing design and development theorems that provide relevant information. Given that the identified user groups of this project are paramedics, doctors, and specialists, we have had to go through our employer to find relevant test subjects. All of these were already connected to the PreViS project, which in many ways has helped give great and direct input on the application's needs and use cases.

This meant that we ourselves had to follow our intuition, and as a result, many of the decisions taken in regards to design and layout is based on our own ideas and suggestions, together with the informants. On the other side, it has been very rewarding and exciting to be a part of an innovation team that develops something almost never done before. We are very honored by the fact that contributions from this project might help people in need, and even save lives.

8.5 Conclusion

Improving the access to healthcare services is an important step toward equal treatment for everyone, and when the PreViS project is implemented into prehospital services, it can help in delivering equal treatment regardless of location. As stated in chapter 2.7, the PreViS team made it clear that the application's most important areas of attention were simplicity and efficiency. Based on this, we formulated the following thesis problem statement:

How can we make the interaction between paramedics and specialists more intuitive and efficient through a web application?

In this chapter, we have discussed whether we have implemented the requirements, and answered the thesis problem statement. We have solved many of the problem areas the employer wanted us to focus on and added features requested by the informants. Based on the discussion in this chapter, we argue that we have made the interaction between paramedics and specialists more intuitive and efficient. This was achieved by improving the size and placement of the interactive buttons, and enhancing the usability through design principles and the user group's opinions and requests. The results from the SUS testing also displayed that our application has excellent usability. By adding the screen mirroring, we also made the application more intuitive in terms of clearer communication between the users.

8.5.1 Future improvements

Given that the PreViS project is still in the development phase, there is a lot of functionalities and features that could improve our application further. We argued that we have made the interaction and usability better than the PreViS prototype, but there were still many areas to improve on. More iterating on design prototypes with the user groups will give better insight into what is needed. There are also several features that need to be added or improved. One area of the developed application that we did not work out properly was the ability to start the ultrasound or thermal camera after a stream had been started. Allowing the paramedics to turn these features on and of is necessary, as these cameras should not be on all the time. This will also minimize the total amount of data being sent over the network, leaving more of the capacity for the main videos of the patient.

We believe that in general, an app or software application would better suit the needs of the project, rather than a website. Browsers have many safety and privacy features that are good,

but in the case of this project, they make things more complicated, such as the need for user approval to share video and sound, which can complicate the fast use flow needed. Things like this may have been made easier with a software program installed on the computer where the application would be used. As this was one of the features that confused the user testers the most, eliminating the manual approval, and making it happen automatically in the background could help get our application far closer to its most efficient.

The quality of video and low latency was crucial for the success of our application. Knowledge from both the previous in-depth project and this one revealed that not much could be changed in regards to quality. WebRTC aims by default for the highest possible quality based on how good the connection it has available is, and as previously mentioned it can be specified in the code what the desired quality is. However, this does not make noticeable changes in testing. By controlling this in a better way, better quality along with more efficient code might help the latency get even lower. We did not find a solution to lowering the latency either. We assume that both quality and latency might be better in the future. As 4G and now recently 5G improve internet speeds even in remote areas, it can help improve the quality. Streaming technologies are also continuously improving, and the latency in newer and more advanced technologies might be lower. In this way, the PreViS project might be a little ahead of its time, as broadband infrastructure is not yet at a level that allows fast enough speeds in some areas of the county, and streaming technologies can get even better. Better measurements of the latency could also have given an indication of where it can be improved, so this is something we wish to be followed up in future development and testing.

Other features that the employer wanted to implement include a mute button, zooming in on a video feed when it is focused in fullscreen mode, and increasing the brightness only when the ultrasound tool is in fullscreen mode, as this tool requires more light to be properly used by the doctors. We tried to implement a mute button in our application, but we were only able to make a deafening button, which deafens the incoming audio rather than muting the outgoing audio. The perfect scenario for this is that the paramedics have earpieces so that the audio from the doctor does not play out loud in the ambulance, which might upset the patient if the doctor has bad news about their condition. The mute function is important on the hospital side because when they mute themselves, it gives the paramedics more focus on what is going on in the ambulance, and if needed the hospital can unmute and talk.

The zoom feature was attempted but only enlarged the video feed, making it grainy and not very detailed. The PreViS team mentioned new cameras with optical zoom would be installed. This allows for much better details even when zoomed far in on small areas of the patient, and will give the user better help to diagnose and triage the patient efficiently.

Increasing the brightness only on ultrasound in fullscreen mode was another wish from the employer. We tried to implement this in our application, but it was challenging to only set the brightness on one of the video elements as they are created dynamically and the ultrasound will not always have the same value. Optimally we needed to test or connect the ultrasound equipment to our application to make this work properly. Adding this feature would further help the doctors or specialists to see the ultrasound clearly.

Another future improvement is to improve the speed and capacity of the STUN and TURN servers. As mentioned in chapter 5.2.3, the STUN and TURN servers were made through Xirsys' free hosting plan, which limits the bandwidth to 500 MB. This can quickly become a problem if the application is used by many different ambulances, creating a lot of traffic on the servers. This problem can be solved by upgrading the current servers to a paid plan with better bandwidth, or by switching to another STUN and TURN provider, depending on the need of the application.

The current mirroring from the hospital screen uses a standard JavaScript function for getting the displayed media. The problem with this function is when you select the desired browser tab for sharing, it shows the whole tab including the hospital's control buttons. This is not a big issue, but ideally, we only want to send the video elements that are being viewed by the hospital. To achieve this we would have to define how much of the screen we want to share, but this is not possible when using the JavaScript function for getting display media. A future improvement to this issue could be to use another method for getting the display media, where we can define exactly what part of the screen we want to mirror.

We wished to have conducted additional testing with the PreViS team. It would have been preferable to test the system in the ambulance using multi-monitors and ultrasound. This would have allowed us to observe the performance of the system in a real-world environment. Implementation with the PreViS team was something we desired but did not accomplish. A future improvement would be to test our application in the ambulance with the correct

cameras and equipment. The implementation of feedback to the application's users is also something to work on in the future and would have been advantageous. For instance, if an error is made or occurs, the user receives informative feedback indicating that something has gone wrong, and in which process the error occurred.

9 Reference list

Bakken, N. (2022). Meeting with paramedic. Microsoft Teams, 02.05.2022. (Accessed: 2nd May 2022).

Bangor, A., Kortum, P.T. and Miller, J.T. (2008). *An Empirical Evaluation of the System Usability Scale*. Available at: https://www.tandfonline.com/doi/pdf/10.1080/10447310802205776?needAccess=true (Accessed: 2nd May 2022).

Bergo, S. Smaaberg, S. Teppan, K. (2022a) *Project outline.* Available at: https://docs.google.com/document/d/1p1yxjCYBnsqzGZEI_D3poN64WArna3beI-FeRBmAWhg/edit?usp=sharing (Accessed: 14th Feb. 2022).

Bergo, S. Smaaberg, S. Teppan, K. (2021b) *IDG3101 In-depth project report.* Available at: https://docs.google.com/document/d/19W9dL10SJoxh_G5ZciZZSMayeHiK5jiLE-0RulzZf3M/edit?usp=sharing (Accessed: 14th Feb. 2022).

Brooke, J. (1995). *(PDF) SUS: A quick and dirty usability scale*. Available at: https://www.researchgate.net/publication/228593520_SUS_A_quick_and_dirty_usability_scale (Accessed: 28th Feb. 2022).

Brooke, J. (2013). SUS: A Retrospective. *Journal of Usability Studies*. Available at: https://uxpajournal.org/wp-content/uploads/sites/7/pdf/JUS_Brooke_February_2013.pdf (Accessed: 9th May 2022).

Codepen (n.d.). *Fitts' Law Formulation*. Available at: https://codepen.io/gsus/full/QyKBGy (Accessed: 10th May 2022).

CSS-TRICKS (2021). *:fullscreen*. Available at: https://css-tricks.com/almanac/selectors/f/fullscreen/ (Accessed: 22nd Apr. 2022).

Dalheim, H.A., Grøtheim, F. and Solberg Vågseter, S. (2021). *Kostnader, produktivitet og økonomisk status i spesialisthelsetjenesten*. Available at: https://www.helsedirektoratet.no/statistikk/samdata-spesialisthelsetjenesten/analysenotater-samdata-spesialisthelsetjenesten/IS-

3011%20Kostnader%20produktivitet%20og%20%C3%B8konomisk%20status%20i%20spesialisthelsetjenesten.pdf/_/attachment/inline/0661d028-f1ac-4ac8-b50d-69d2b920051f:ae1e77d45a842fd0db55e30a88e8bed09cc9cb56/IS-3011%20Kostnader%20produktivitet%20og%20%C3%B8konomisk%20status%20i%20spesialisthelsetjenesten.pdf (Accessed: 28th Apr. 2022).

Dalheim, H.A. and Solberg Vågseter, S. (2020). *Kostnader i spesialisthelsetjenesten*. Available at: https://www.helsedirektoratet.no/rapporter/kostnader-i-spesialisthelsetjenesten-2019/Kostnader%20i%20spesialisthelsetjenesten%202019%20-%20IS%202949.pdf/_/attachment/inline/7b171ed2-9ef8-4834-8143-397a0d3f6d75:32a12747d28e19778a4946a467e19d1d6bc1e3eb/Kostnader%20i%20spesialisthelsetjenesten%202019%20-%20IS%202949.pdf (Accessed: 28th Apr. 2022).

Discord (n.d.). *What is Discord | A Guide for Parents and Educators*. Available at: https://discord.com/safety/360044149331-What-is-Discord (Accessed: 1st May 2022).

Figma (n.d.). *A Free, Online UI Design Tool For Teams | Figma*. Available at: https://www.figma.com/ui-design-tool/ (Accessed: 1st May 2022).

FN-SAMBANDET (n.d.). *God helse og livskvalitet*. Available at: https://www.fn.no/om-fn/fns-baerekraftsmaal/god-helse-og-livskvalitet (Accessed: 2nd May 2022).

Førland, O., Zakariassen, E. and Hunskår, S. (2009). Samhandling mellom ambulansearbeider og legevaktlege, *Tidsskrift for Den norske legeforening*, (11). doi:10.4045/tidsskr.08.0501. (Accessed: 14th Apr. 2022).

GitHub (n.d.). *Build software better, together*. Available at: https://github.com/about (Accessed: 1st May 2022).

Google Docs Editors Help (n.d.). *How to use Google Docs - Computer - Docs Editors Help*. Available at: https://support.google.com/docs/answer/7068618?hl=en&co=GENIE.Platform%3DDesktop (Accessed: 1st May 2022).

Google Workspace Learning Center (n.d.). *What can you do with Docs? - Google Workspace Learning Center*. Available at: https://support.google.com/a/users/answer/9300503?hl=en (Accessed: 1st May 2022).

Gordon, K. (2020). *5 Principles of Visual Design in UX*. Available at: https://www.nngroup.com/articles/principles-visual-design/?lm=color-enhance-design&pt=article (Accessed: 9th May 2022).

Gordon, K. (2021). *Using Color to Enhance Your Design*. Available at: https://www.nngroup.com/articles/color-enhance-design/ (Accessed: 9th May 2022).

Gryting, N.H. (2022). Interview with paramedic. Microsoft Teams, 30.03.2022. (Accessed: 30th Mar. 2022).

Helse- og omsorgsdepartementet (n.d.). *Nasjonal helse- og sykehusplan 2020-2023*. Available at: https://www.regjeringen.no/contentassets/e353a5d022d84deabd969a5fe043783e/no/pdfs/i-1194_b_kortversjon_nasjonal_helse.pdf (Accessed: 2nd Apr. 2022).

Helsedirektoratet (2019). *Kostnader i spesialisthelsetjenesten*. Available at: https://www.helsedirektoratet.no/rapporter/is-2847-kostnader-i-spesialisthelsetjenesten/IS-2847%20Kostnader%20i%20spesialisthelsetjenesten.pdf/_/attachment/inline/f0e87f77-220b-4440-b552-fce996ede52a:83a989edfcbc748d0a970e4919e1105b83d27377/IS-2847%20Kostnader%20i%20spesialisthelsetjenesten.pdf (Accessed: 28th Apr. 2022).

Hem, E. (2004). Fargeblinde leger – er det noe problem? *Tidsskrift for Den norske legeforening*, (2). Available at: https://tidsskriftet.no/2004/01/kronikk/fargeblinde-leger-er-det-noe-problem (Accessed: 9th May 2022).

Heroku (2020). *Cloud Application Platform | Heroku*. Available at: https://www.heroku.com/ (Accessed: 4th May 2022).

Høvding, G. (2020). *rød-grønn fargeblindhet – Store medisinske leksikon*. Available at: https://sml.snl.no/r%C3%B8d-gr%C3%B8nn_fargeblindhet (Accessed: 9th May 2022).

IBM (2019). *What is software testing?* Available at: https://www.ibm.com/topics/software-testing (Accessed: 6th May 2022).

Interaction Design Foundation (n.d.). *What is Fitts' Law?* Available at: https://www.interaction-design.org/literature/topics/fitts-law (Accessed: 5th May 2022).

Langhelle, A. *Et al.* (2004). *International EMS Systems: the Nordic countries.* Resuscitation, 61(1). doi:10.1016/j.resuscitation.2003.12.008. (Accessed: 14th Apr. 2022).

Mdn web docs. (n.d.). *Introduction to WebRTC protocols - Web APIs | MDN*. Available at: https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API/Protocols (Accessed: 7th Feb. 2022).

Mdn web docs (n.d.). *MediaDevices.getDisplayMedia() - Web APIs | MDN*. Available at: https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getDisplayMedia (Accessed: 11th May 2022).

Mdn web docs (n.d.). *MediaDevices.getUserMedia() - Web APIs | MDN*. Available at: https://developer.mozilla.org/en-US/docs/Web/API/MediaDevices/getUserMedia#parameters (Accessed: 13th Apr. 2022).

Mdn web docs (n.d.) *UDP (User Datagram Protocol).* Available at: https://developer.mozilla.org/en-US/docs/Glossary/UDP (Accessed: 6th May 2022).

Microsoft (n.d.). *Record a meeting in Teams*. Available at: https://support.microsoft.com/en-us/office/record-a-meeting-in-teams-34dfbe7f-b07d-4a27-b4c6-de62f1348c24 (Accessed: 1st May 2022).

Mizrachi, D. (n.d.). *What is monday.com?* Available at: https://support.monday.com/hc/en-us/articles/115005310945-What-is-monday-com- (Accessed: 1st May 2022).

Morales, J. (2021). *Mobile First Design Strategy: The When + Why | Adobe XD Ideas*. Available at: https://xd.adobe.com/ideas/process/ui-design/what-is-mobile-first-design/ (Accessed: 2nd May 2022).

NOU 2015: 17 (2015). *Først og fremst: Et helhetlig system for håndtering av akutte sykdommer og skader utenfor sykehus*. Oslo: Departementenes sikkerhets- og serviceorganisasjon, Informasjonsforvaltning. Available at: https://www.regjeringen.no/contentassets/477c27aa89d645e09ece350eaf93fedf/no/pdfs/nou201520150017000dddpdfs.pdf (Accessed: 10th May 2022).

Office for National Statistics (n.d.). *Using red and green – Style.ONS*. Available at: https://style.ons.gov.uk/data-visualisation/using-colours/using-red-and-green/ (Accessed: 8th May 2022).

Pouncey, I. and York, R. (2011). *Cascading Style Sheets for Web Design*. Available at: https://web.p.ebscohost.com/ehost/ebookviewer/ebook?sid=ed49b55f-c925-4ddc-bee6-19402979bd8b%40redis&ppid=pp_ix&vid=0&format=EB (Accessed: 4th May 2022).

ReactJS (n.d.). *Context – React*. Available at: https://reactjs.org/docs/context.html (Accessed: 19th Apr. 2022).

ReactJS (n.d.) *Instance properties.* Available at: https://reactjs.org/docs/react-component.html#state (Accessed: 19th Apr. 2022).

ReactJS (n.d.). *Using the Effect Hook – React*. Available at: https://reactjs.org/docs/hooks-effect.html (Accessed: 19th Apr. 2022).

Salehi, A.M. and Li, X. (2021). *Low-latency Delivery Networks for Multimedia Streaming*. Springer. Available at: https://link.springer.com/chapter/10.1007/978-3-030-88451-2_7?noAccess=true (Accessed: 5th May 2022).

Smith, J. (2012). *Applying Fitts' Law To Mobile Interface Design*. Available at: https://webdesign.tutsplus.com/articles/applying-fitts-law-to-mobile-interface-design--webdesign-6919 (Accessed: 5th May 2022).

Socket.IO (n.d.). *Introduction | Socket.IO*. Available at: https://socket.io/docs/v4/#what-socketio-is (Accessed: 30th Apr. 2022).

St. Meld. 7 (2008-2009) (2009) *Et nyskapende og bærekraftig Norge.* Available at: https://www.regjeringen.no/no/dokumenter/stmeld-nr-7-2008-2009-/id538010/?ch=1 (Accessed: 7th Feb. 2022).

St.meld. nr. 47 (2008-2009) *Samhandlingsreformen: Rett behandling - rett sted - til rett tid.* Oslo: Det kongelige helse- og omsorgsdepartement. Available at: https://www.regjeringen.no/no/dokumenter/stmeld-nr-47-2008-2009-/id567201/?ch=1 (Accessed: 9th Feb. 2022).

Statcounter Globalstats (n.d.). *Browser Market Share Worldwide*. Available at: https://gs.statcounter.com/browser-market-share#monthly-202004-202204 (Accessed: 3rd May 2022).

Statens legemiddelverk (2020). *Dokumentasjon av enhetskostnader*. Available at: https://legemiddelverket.no/Documents/Offentlig%20finansiering%20og%20pris/Dokumentasjon%20til%20metodevurdering/Dokumentasjon%20av%20enhetskostnader%20V1.pdf (Accessed: 2nd May 2022).

Steane, J. (2018). *The Principles and Processes Of Interactive Design*. London Bloomsbury Visual Arts. (Accessed: 6th Apr. 2022).

Stokke, P. A. (2022). Interview with employer. Microsoft Teams, 27.10.2021. (Accessed: 3rd Apr. 2022).

Tjernshaugen, A. *Et al.* (2022). *koronapandemien*. Available at: https://sml.snl.no/koronapandemien (Accessed: 14th Mar. 2022).

Tomitsch, M. *Et al.* (2018). *Design. Think. Make. Break. Repeat a handbook of methods*. Amsterdam Bis Publishers B.V. (Accessed: 6th Apr. 2022).

Usability.gov (2019) *Usability Evaluation Basics | Usability.gov*. Available at: https://www.usability.gov/what-and-why/usability-evaluation.html (Accessed: 27th Apr. 2022).

Visual Studio Code (n.d.). *Documentation for Visual Studio Code*. Available at: https://code.visualstudio.com/docs (Accessed: 1st May 2022).

W3 Schools (n.d.). *What is GitHub*. Available at: https://www.w3schools.com/whatis/whatis_github.asp (Accessed: 1st May 2022).

W3C (2018). *Web Content Accessibility Guidelines (WCAG) 2.1*. Available at: https://www.w3.org/TR/WCAG21/#contrast-minimum (Accessed: 9th May 2022).

WebAIM (n.d.). *WAVE Web Accessibility Tool*. Available at: https://wave.webaim.org/ (Accessed: 9th May 2022).

WebAIM (n.d.). *WebAIM: Contrast Checker*. Available at: https://webaim.org/resources/contrastchecker/ (Accessed: 9th May 2022).

WebRTC. (n.d.). *WebRTC*. Available at: https://webrtc.org (Accessed: 23rd Feb. 2022).

10 Figure list

11 Table list

12 Attachments

Appendix 1: Internal application Sweden-Norway Interreg.

Appendix 2: IDG3101 In-depth report.

Appendix 3: Project outline.

Appendix 4: Link to Figma prototype including wireframes.

Appendix 5: Project agreement with employer.

Appendix 6: SUS scores.

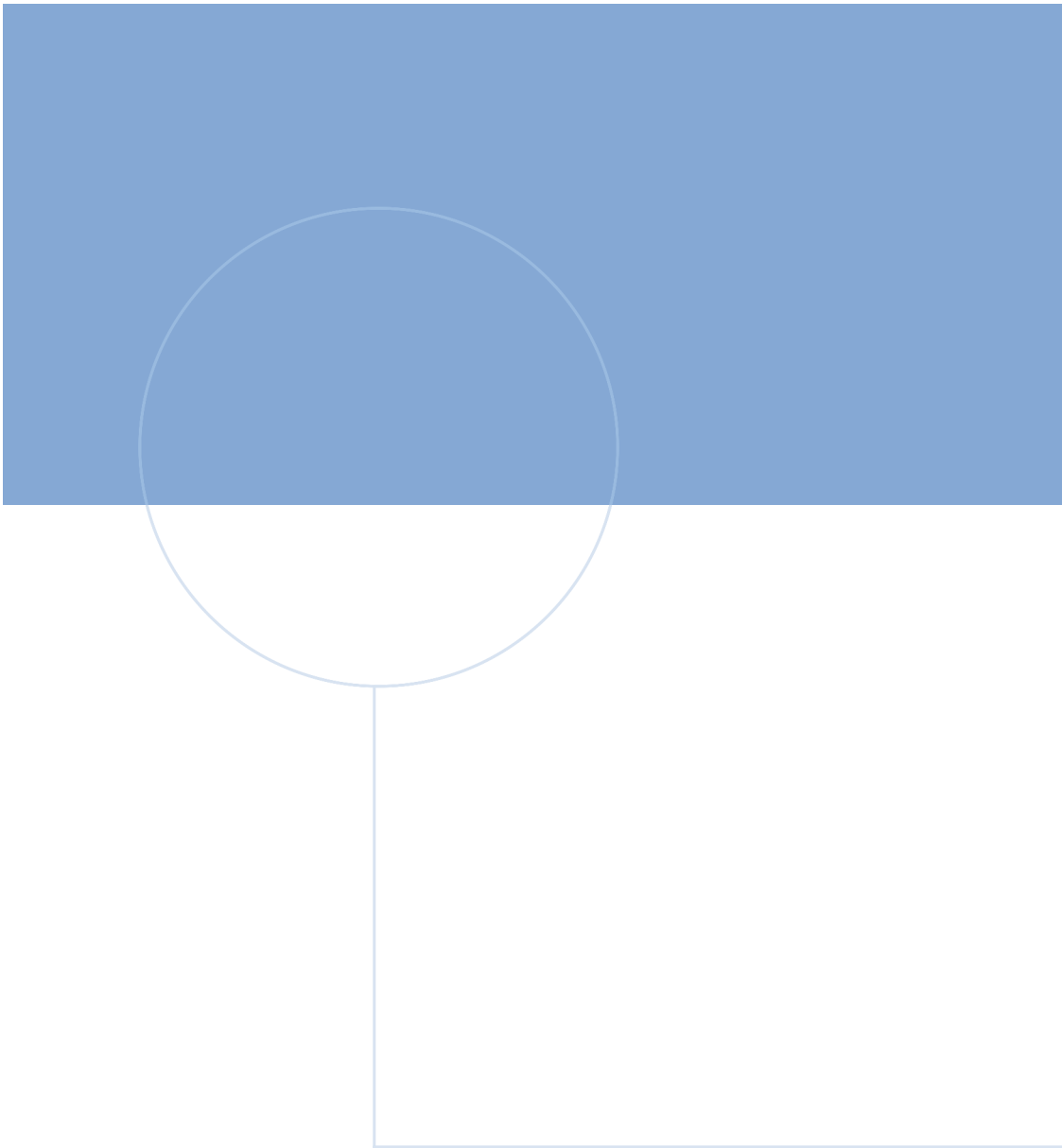
Appendix 7: Transcribed interview with Nils H. Gryting, paramedic.

Appendix 8: Transcribed interview with Tom Bakken, paramedic.

Appendix 9: Transcribed interview with Roy H. Otterholt Bekkeseth.

Appendix 10: Interview guide.

Appendix 11: Source code.



NTNU

Norwegian University of
Science and Technology