Collaborative user interface design of a remote device control system

Master of Science (Tech) Thesis
University of Turku
Department of Computing
Software Engineering
2024
Miika Toivanen

Supervisors:
Seppo Helle (University of Turku)
Jouni Smed (University of Turku)

UNIVERSITY OF TURKU
Department of Computing

Miika Toivanen: Collaborative user interface design of a remote device control system

Master of Science (Tech) Thesis, 51 p.
Software Engineering
2024

This thesis explores how the user interface of a remote device control system can be re-designed and re-implemented in collaboration with the users of the current system. First, a literature review is conducted to identify appropriate research methods for involving users in the design and implementation process. This is followed by exploring the evaluation methods that are used to collect numerical data about the current and the new user interface. The chosen methods for user involvement were focus groups and semi-structured interviews, and for evaluation methods System Usability Score and time-on-task were selected. The tools used to conduct the research are also introduced. The thesis then discussed the technical background and introduces more technical tools that are used.

The re-design process is done in collaboration with the users by using the data collected from the users as a foundation for the design. Figma prototypes are used to facilitate iterative design refinement in order to validate assumptions. The thesis then briefly discussed the implementation phase and the limitations related to it.

The current and the new user interface were compared against each other by using the selected evaluation methods. The new user interface was successful in improving both the SUS score and the time measured from time-on-task tasks. The average measured SUS score increased by 65%, transitioning from a grade of D to a grade of A+. Time spent on tasks was improved by 9%, although the change was smaller than anticipated.

Keywords: user interface design, remote device control system, human-computer interaction, UI, visual design, user-centered design

Contents

List of Figures

List of Tables

1 Introduction

User interfaces are an interesting topic as they define how humans can interact with machines. A good user interface is pleasant to use, whereas a poorly designed or implemented user interface can make simple tasks complicated and cause frustration. Although we can easily distinguish a poor user interface when we use one, it is much more difficult or near impossible to design and implement the most optimal one for a given system.

The human-machine interaction is also interesting because humans are different. Some users may also face limitations, including inability to use pointing devices such as a mouse, or visual impairments. On top of that, people approach the same tasks from multiple viewpoints and take different paths to complete them. This means that the designers can not just design for themselves, but instead they have to design for users. Therefore, in order to design a good, usable user interface, we need to communicate with the users and find out how they see the system and what kind of approaches they take to complete actions so those can be optimized.

The aim of this thesis is to design and implement a new user interface for an existing remote device management system in collaboration with the users of the system. The first initiative for the need of a new user interface came from the users, as they requested new features and bug fixes on a regular basis. For instance, the user interface did not provide enough feedback and as a result the users found themselves confused if an action they initiated had actually done anything. As the existing user interface was already aged, implementing new features and fixes for it along with necessary updates and code changes was deemed to be as time-consuming as designing and implementing a new user interface from scratch. Redesigning and implementing the user interface from scratch also gives a bit more flexibility as there are no existing constraints that could possibly make some design choices impossible.

The redesign and implementation was done in collaboration with six participants that use the system actively and have different profiles and varying lengths of experience with the system.

The goal of this thesis is to answer the following research questions:

> *RQ 1: How can the user interface be improved in collaboration with current users?*
>
> *RQ 2: What are the metrics that can be used to assess the current and the new implementation of the user interface?*
>
> *RQ 3: How can the end result of the improvement process be validated using the metrics chosen?*

Chapter 2 introduces the fundamentals of user-centered design (UCD) and how it is used in the process of re-designing the user interface. Following this, the chapter showcases the research and evaluation methods that are applied in the thesis. Finally, the chapter briefly explains the tools, guidelines and platforms utilized.

Chapter 3 explains the technical background of the thesis by introducing the relevant technologies used, including JavaScript, TypeScript, React, Docker, Google Cloud Platform (GCP), and GitHub.

Chapter 4 introduces the remote device management system and explains the use case for it. The chapter then goes on to discover the usability problems associated with the current implementation. The chapter also discusses how focus group and interviews were arranged and what kind of feature requests the participants requested for.

Chapter 5 focuses on how the new user interface was designed and implemented, and how the users were involved in the process.

Chapter 6 presents the key findings and results obtained from the Chapters 4 and 5, providing an in-depth analysis and discussions of the results. The chapter also discusses the possible explanations for the results obtained.

Chapter 7 serves as the conclusion chapter, summarizing the key findings and concepts presented throughout the previous chapters. In addition, the chapter answers the research questions from Chapter 1.

2 Background

This chapter aims at answering RQ 1 by exploring user-centered design as well as research and evaluation methods. It also introduces tools, guidelines and platforms used.

2.1 User-centered design

User-centered design (UCD) is a design process that focuses primarily on the users and their needs. Users are involved in the design process in each phase via different research and design methods in order to create a product that suits their needs. [1]

In UCD, different research and generative methods such as surveys, interviews and brainstorming are used to develop a deeper understanding of the user's needs. The process is iterative, as the design is evaluated against user needs until a satisfactory level is achieved. [1]

If applied to full extent, the UCD process involves experts from various fields, such as psychologists, ethnographers and software engineers. In this thesis, no other experts were used. The primary reason for this is the project's constraints in terms of both resources and time.

In this thesis, the role of the users is to provide ideas and features requests that they hope to see in the end product. Furthermore, users are expected to communicate any problems found in the current implementation. As in this case the users are not designers or programmers, it is up to the designer to make those ideas and requests viable and further improve them so they can be implemented as well as provide solutions to the problems found.

2.2 Research methods

There are numerous different methods that researchers can choose from. The following research methods were chosen based on literature and their suitability for the research process. Each chosen method can be applied to a small group of participants and require no immediate presence between the researcher and the participant, which was crucial due to the nature of the project.

In the project, I took the roles of researcher, designer and coder. This allowed me to target the research on the problematic areas, make design choices that supported the actual coding process as well as made the iteration of the product design and implementation faster.

All of the six participants are real users of the system with different roles and experience levels. The participants consist of:

- 2 developers, who also use the system to perform maintenance tasks.
- 1 product owner.
- 3 system users, who use the system regularly for its intended purpose.

Participants that were not the typical users of the system, such as the product owner and the developers, were selected in order to get feedback representing less experienced and new users. By selecting only the typical users, the results could have been different but less inclusive at the same time. Including more participants with different profiles was not feasible at the time of the research due to the scope of the project.

2.2.1 Semi-structured interview

Semi-structured interview is an interview method that combines both predefined questions and ad-hoc questions that give the interviewer the ability to explore emerging topics or issues. Semi-structured interviews can be used to gather opinions and facts from the users, as well as give the users ability to raise new issues in scenarios where the relevant issues have already been identified. The length of semi-structured interviews can vary from several minutes to several hours. Whereas short interviews might not provide enough data, too long interviews can reduce the number of qualified participants willing to participate. [2]

The strengths of semi-structured interview include the ability to discover previously unknown issues, mechanism to redirect conversation back to the relevant topic as well as the ability to address complex topics. Interviews can also be broadly compared with other interviews as they share the same base set of questions. [2]

Although semi-structured interview has many strengths, semi-structured interview can suffer from what is known as the 'interviewer effect' where the background of the interviewer might affect the answers. There is also a risk that the interviewer might guide the participant to give a particular answer or put words into the participant's mouth. [2] As the risks associated with semi-structured interview techniques were known before the actual interviews made it possible to avoid them as well.

2.2.2 Nielsen heuristics

One way to identify underlying design problems in user interfaces is to use Jakob Nielsen's 10 usability heuristics [3]. Nielsen's heuristics are not specific guidelines, but more of rules of thumb in evaluating the usability of a user interface design. These heuristics are useful early in the design process as they require no user input to identify problems and they can also be used in collaboration with user research to better target certain areas during testing. [4] These heuristics were used as the foundation for the semi-structured interview questions later on in the thesis. Nielsen's 10 heuristics are:

1. *Visibility of system status*

   Users should always be informed of the system status through feedback within a reasonable amount of time. This increases trust in the product and helps the user to determine the next steps.

2. *Match between system and the real world*

   The design should use concepts that the users are familiar with, as well as display information in natural and logical order. By doing so, the product becomes easier to learn and feels more intuitive to use.

3. *User control and freedom*

   The users should be able to quickly leave unwanted actions because people make mistakes and it can be frustrating for the user to get stuck without a clear exit.

4. *Consistency and standards*

   The users should not have to think if different words or actions mean the same thing. Platform and industry conventions should be followed. Consistency lowers cognitive load when users move between using different products.

5. *Error prevention*

   The design should prevent the users from making errors by eliminating error prone conditions and presenting users with confirmation option. Slips caused by inattention can be prevented with constraints and defaults, whereas mistakes can be prevented with warnings and undo support.

6. *Recognition rather than recall*

   Elements of the UI should be visible and the amount of information that the users have to remember should be minimal. Doing so helps to lower the cognitive load.

7. *Flexibility and efficiency of use*

   More advanced users should be presented with shortcuts and customization options. This helps to make the product suit both novice and advanced users.

8. *Aesthetic and minimalist design*

   The UI should not contain irrelevant or rarely used information. Any extra unit of information directly competes with the relevant units of information, making them less visible.

9. *Help users recognize, diagnose, and recover from errors*

   Error messages should be clear and suggest a solution for the error instead of showing error codes for example. Visuals for the errors should promote their visibility and recognition.

10. *Help and documentation*

    If the system needs explanation, it should be focused on the user's task and easy to search. Concrete steps to take should be listed.

Focus groups

Focus groups are small groups of users that provide feedback about a certain topic. Focus groups are limited in size, as no more than 12 participants should participate in a single focus group session. In order to involve more users, multiple focus group sessions can be held with different participants. In focus group sessions, the moderator asks open-ended questions about the product in order to collect ideas about how the users use the product as well as what they expect the product to be capable of doing. Therefore, focus groups can be a valuable tool when defining features for existing and new products. The goal of focus groups is to provide multiple opinions and viewpoints that can be used in the design process. [5]

2.3 Evaluation methods

Evaluation methods are methods that are used to collect numerical data in order to compare two implementations of the user interface. Whereas the System Usability Scale provides qualitative data that captures the users' perception of usability, Time-on-task provides quantitative data on users' efficiency and performance on certain tasks.

System Usability Scale

The System Usability Scale (SUS), developed by John Brooke in 1986, is a scale that gives a general view of subjective assessments of usability. The SUS is typically used before any discussion or debriefing, after the respondent has used the system being evaluated. [6]

The System Usability Scale is made up of ten statements that alternate between positive and negative statements. The respondent has to then indicate the degree of agreement to each statement on a 5-point scale. The overall usability score can be calculated by subtracting the user's score from 5 for even-numbered questions and subtracting 1 from the user's score for odd-numbered questions. The adjusted scores are then summed and the sum is multiplied by 2.5. SUS score is measured from 0 to 100. [6]

Evaluating SUS score differs a bit from source to another, but a common conception is that the average score for a system is 68. Scores above 68 are considered to be above average, whereas scores below 68 are considered to be below average [7].

Table 2.1: SUS score evaluation, modified from [8]

Grade	SUS score	Percentile range	Adjective
A+	84.1 - 100	96-100	Best imaginable
A	80.8 - 84.0	90-95	Excellent
A-	78.9 - 80.7	85-89	
B+	77.2 - 78.8	80-84	
B	74.1 – 77.1	70 – 79	
B-	72.6 – 74.0	65 – 69	
C+	71.1 – 72.5	60 – 64	Good
C	65.0 – 71.0	41 – 59	
C-	62.7 – 64.9	35 – 40	
D	51.7 – 62.6	15 – 34	OK
F	<= 51.6	< 15	

Instead of a linear progression, the SUS score follows a curved pattern. This curved pattern can be expressed as a table with grades for score ranges [8] as shown in Table 2.1.

It is also important to notice that the score of a single item is not meaningful on its own [6], and therefore no conclusions can be drawn from the answers to a single question.

Time-on-task

Time-on-task, otherwise known as task completion time, is a way to measure the efficiency of a product. Time-on-task is relatively easy to measure, as it is the time elapsed between the start and the end of a certain task. Typically, time a task takes is measured in seconds or minutes. For products where the user repeatedly performs tasks, a lower time on task usually means a more efficient product, although this might not be true for all products. [9] When measuring the time spent on a single task, it is important that both the participant and the timekeeper know the conditions when a specific task is completed to prevent a case where for example the participants thinks they have completed the task and stop.

2.4 Tools, guidelines and platforms

This section showcases the primary tools used in both pre-design and design phases. Tools related to the implementation phase are introduced in Chapter 3.

Figma

Figma is a cloud-based user interface design tool developed by Figma, Inc. and released in 2015, that can be used to create designs as well as clickable prototypes. Figma offers both free and paid plans. [10] Figma was used in this project to create designs as well as prototypes that made it possible to iterate the design before any actual code was written.

Figma also supports third-party plugins that make it possible to turn Figma designs into code, but the quality and maintainability of the code these plugins produce can not be verified and for that reason those plugins were not used.

There are multiple Figma alternatives available, most well-known being Adobe XD. The reason for choosing Figma as the design tool was because of my experience with the software and access to a paid subscription, which provides more features, such as advanced prototyping.

Microsoft Teams

Microsoft Teams is an online communication software developed by Microsoft. Microsoft Teams offers a wide variety of features, including video meetings. It also supports other Microsoft applications, such as Microsoft Whiteboard, a collaborative online whiteboard. Microsoft Teams offers both free and paid plans, although the free plan is intended for non-business use only. [11]

Microsoft Teams was used in this project because the customer's organization is using mainly Teams and the features that Teams offers are sufficient for this project. Other video meeting solutions, such as Google Meet would have most likely been sufficient as well.

WCAG 2

Web Content Accessibility Guidelines (WCAG) 2 is a technical standard that defines how websites and applications can be made accessible to users with disabilities. The latest version of the standard is 2.2, published on October 5, 2023. [12] Although the standard defines a lot of accessibility features, not all are relevant in this context. As the user pool is fairly well known and the system will be in limited use instead of being available to the public, some features, such as support for blind users can be overlooked. However, if the system were to be adopted for use by a larger user pool, these accessibility features and the need for them would need to be re-evaluated.

As WCAG is an open standard, there exists a large number of online tools that can be used to verify that a web application follows the guidelines. One useful tool that was used was a contrast checker by WebAIM [13] to verify that text stayed readable in all cases, even when in dark mode. WebAIM also provides another tool called *WAVE* (web accessibility evaluation tool) that was used to verify applicable accessibility features.

Material Design

Material Design is an open-source design system created by Google and widely adopted across Android, iOS and the web. Material design provides a comprehensive set of UI components, including buttons, menus, cards and more. Each component has specific design guidelines and usage recommendations. [14] By utilizing an already existing design system the system is already tested and the time needed for design is significantly reduced.

3 Technical background

The technical aspects of the project help to clarify the project's environment as well as to justify design choices that were made. As a web application, the project is limited by the functionality that modern web technologies offer.

In the same way as the current implementation, the new implementation was also implemented as a web application instead of a more traditional desktop software that requires to be installed on the end user's machine. The main reasons for this were the ease of use as well as the suitability of the web technologies available. The product had to be easily distributed to a large number of users using both desktop and mobile platforms and the implementation did not require large amounts of computational power or access to native application programming interfaces (APIs). Therefore, implementing the product as a web application was seen as the most viable option, as it made the distribution of the product easy for both desktop and mobile platforms, required no initial setup from the end users and made applying updates to the application much easier. The following tools introduced are the tools that were used in the thesis.

3.1 JavaScript

JavaScript is a single-threaded, dynamically typed, just-in-time compiled programming language that supports first-class functions. It is best known for its use in the web, but JavaScript is supported by non-browser environments as well, such as

Node.js. [15]

JavaScript is standardized as ECMAScript (ECMA262), and the first version was adopted in June of 1997. [16]

As JavaScript is dynamically typed, variables in JavaScript can hold values of any type. For example, in Listing 3.1 it is evident that the function `hello()` can be called with a variable of any type. This can lead to a variety of problems if not taken into consideration during coding.

```
const hello = (name) => {
    return `Hello, ${name}!`
}
console.log(hello("Joe")) // Output "Hello, Joe!"
console.log(hello(123)) // Output "Hello, 123!"
```

Listing 3.1: Example JavaScript code

As of 2024, according to a study by W3Techs, 98.8% of websites globally use JavaScript as a client-side programming language [17]. Stack Overflow's survey marks JavaScript as the most popular programming language for the eleventh year in a row [18], reasoning the use of JavaScript.

3.2 TypeScript

TypeScript builds on top of JavaScript by adding types and type checking to improve developer experience as well as to catch errors early on. TypeScript works everywhere where JavaScript does, as TypeScript code is converted to JavaScript.

The example from Listing 3.1 can be re-implemented in TypeScript to show how the typing system can prevent errors early on. It can be seen from Listing 3.2 that assigning values with types other than `string` to the variable `name` raises an error.

```
const hello = (name: string): string => {
    return `Hello, ${name}!`
}
console.log(hello("Joe")) // Output "Hello, Joe!"
console.log(hello(123)) // Error: Argument of type 'number' is not
    assignable to parameter of type 'string'.
```

Listing 3.2: Example TypeScript code

3.3 React

React is an open-source JavaScript library developed and maintained by Meta with the help of voluntary contributors around the world. React can be used to build user interfaces from components by combining both HTML and JavaScript (or TypeScript) in a syntax known as JSX (if using TypeScript, TSX), [19] that is demonstrated in the Listing 3.3, showcasing a basic React component that renders the results of `hello()` in a HTML paragraph.

```
const HelloComponent = () => {
  const hello = (name: string): string => {
    return `Hello, ${name}!`
  }
  return (
    <div>
      <p>{hello("Joe")}</p>
    </div>
  )
}
```

Listing 3.3: Example React component

React's initial public release was in 2013, and has since seen 18 major versions, the newest one being version *18.2.0* released in June of 2022. [20] According to the 2023 Stack Overflow's survey, React was the most used web technology amongst professional developers. [18] React's widespread adoption has created an ecosystem of open-source libraries that simplify the process of creating complex functionalities within React applications. While features like internationalization (i18n) and authentication can be complex to implement, these libraries significantly simplify the process. As a drawback, these libraries limit transparency and potentially make customization more difficult.

There are also frameworks for React, such as Next.js, that enhance the core React library by providing features such as server-side rendering (SSR) and routing. These frameworks come with the drawback of increased complexity and potential bias, often making it difficult to switch from a framework to another without major refactoring.

3.4 Other tools

Docker

Docker is a platform that enables applications to be separated from the infrastructure they run on. Docker allows applications to be packaged into containers, which are loosely isolated environments that contain everything needed to run the application. [21] By using a platform such as Docker, the application can be packaged and deployed on almost any platform, as long as it supports Docker. As the deployment is not tied to a certain provider it allows organizational changes, such as migrating from one cloud platform to another, to take place.

Google Cloud Platform

Google Cloud Platform (GCP) is a collection of cloud computing services provided by Google. [22] One of the services available, *Compute Engine*, allows the creation and management of virtual machines on the platform. These virtual machines can then be scaled up and down based on requirements, ensuring performance and cost-efficiency. There are other alternatives to GCP, such as Amazon Web Services

(AWS), Microsoft Azure or more traditional on-premise hosting. The reason for selecting GCP was that the customer's organization was already using it and the other alternatives did not provide any advantages.

Git & GitHub

Git is an open-source distributed version control system developed by Linus Torvalds in 2005. Git works by recording changes known as commits to allow access to prior versions of the source code. GitHub on the other hand is a developer platform owned by Microsoft that utilizes Git. GitHub offers a variety of features, such as hosting Git repositories, automated CI/CD (continuous integration / continuous development) pipelines, container registries (packages) and so on. GitHub offers both free and paid subscriptions. [23] By utilizing Git, code changes can be pushed to the code repository hosted by GitHub. After certain conditions, such as code being pushed to a certain branch, an automated workflow will verify the code quality and build the code into an application on the cloud. By utilizing this workflow, code can be built and packaged into Docker images. GitHub hosts these images on their servers, making it easy to deploy them on Docker running on the GCP virtual machine. There exists alternatives for both Git and GitHub, although Git is the most used version control system. GitHub has multiple alternatives, GitLab being the most popular. These alternatives do not typically offer any advantages, and the reasoning for choosing Git and GitHub was based on preference.

Material UI

Material UI is an open-source React component library, developed by a company called MUI, that implements Google's Material Design. The library consists of prebuilt React components that can be used in production straight away and customized easily. Material UI is based on Google's Material Design 2 and support for

Material Design 3 is already planned. By utilizing Material UI, components such as buttons and modals are already tested and ready to use. Material UI also provides a theming system, allowing colors and fonts to be predefined in the code for later use. [14]

4 Assessment of the current implementation

This chapter focuses on explaining the remote device management system, goes through the process of discovering current problems as well as discusses feature requests.

4.1 Remote device management system

The current implementation in use is a remote device management system that allows the users to control and manage different kinds of devices remotely. Devices managed from the system are for example ticket vending kiosks and other devices that require constant monitoring and remote management. The current system is designed to support hundreds of devices, as well as user-specific access privileges with designated roles and organizations.

4.2 Discovering current problems

The process of discovering the current usability issues of the remote device management system started with a focus group that had six participants in total. The focus group was held online via Microsoft Teams, as there was great geographical distance between the participants.

The focus group began by introduction to the topic which was to assess the usability problems with the current implementation and ideas for the new implementation. The participants were instructed to think and write down what aspects of the UI had usability issues, what parts of the UI did work and what features the participants would hope to have in the future version of the application. This part of the workshop utilized Microsoft Teams' Whiteboard, which allowed the participants to see what others were writing down in real-time. After the participants were done writing notes, each note was discussed and the writer of the note had a chance to elaborate the note further.

The focus group was successful in discovering usability issues related to the real-time functionality of the application, as well as multiple issues related to the Nielsen heuristics. One major issue that contradicted the Nielsen heuristics was that the visibility of the system status was often insufficient. The user interface was also considered to be complicated and the functionalities were placed illogically.

4.2.1 Interviews

After the focus group, I interviewed each participant personally in order to get a better understanding of the usability problems with the current implementation as well as map out missing features. The interviews were held as semi-structured, and they included ten questions related to usability and design. The participants were also asked to grade the current implementation on a scale 1–10 along with a reasoning for the grade to gain a deeper understanding of the issues they were experiencing.

The questions the participants were mostly asked in the semi-structured interviews were the following:

- Q1: What aspects of the UI catch your eye in the first place?
- Q2: Do you feel that the UI gives appropriate feedback?

- Q3: Does the UI have features that you never use?
- Q4: Do you feel like you can find all the relevant information easily in the UI?
- Q5: Do you feel that the language used in the UI is clear and uniform?
- Q6: Do you feel that the UI is modern? How do you feel about the placement of the elements and the use of colors?
- Q7: Would you like to use the UI in other languages?
- Q8: Have you faced bugs or other defects that make the use of the UI difficult?
- Q9: Have you used the UI with a mobile device such as a phone or a tablet?
- Q10: Do you feel like the UI is missing features that it should absolutely have?

As typical for a semi-structured interview, some of the questions were not asked if the participant had already answered them in a way or another, and some of the answers led to more in-depth questions.

The interviews verified the initial problems which rose from the workshop. Participants felt that the usability problems hindered the usability and user experience, but did not make the current implementation inoperable. As Figure 4.1 suggests, the grades given for the current implementation averaged a grade of 6.6 out of 10. While the grade itself was not either bad or great, asking the participants to give a grade with reasoning discovered even more aspects that hindered the usability.

During the interviews, participants highlighted issues impacting their work and largely overlooked those not relevant to them. This highlights the importance of diverse participants to uncover the widest range of issues.

4.2.2 Usability issues

One of the usability issues that was identified from the interviews was that *managing hundreds of devices from a list view was troublesome.* Users expressed that they

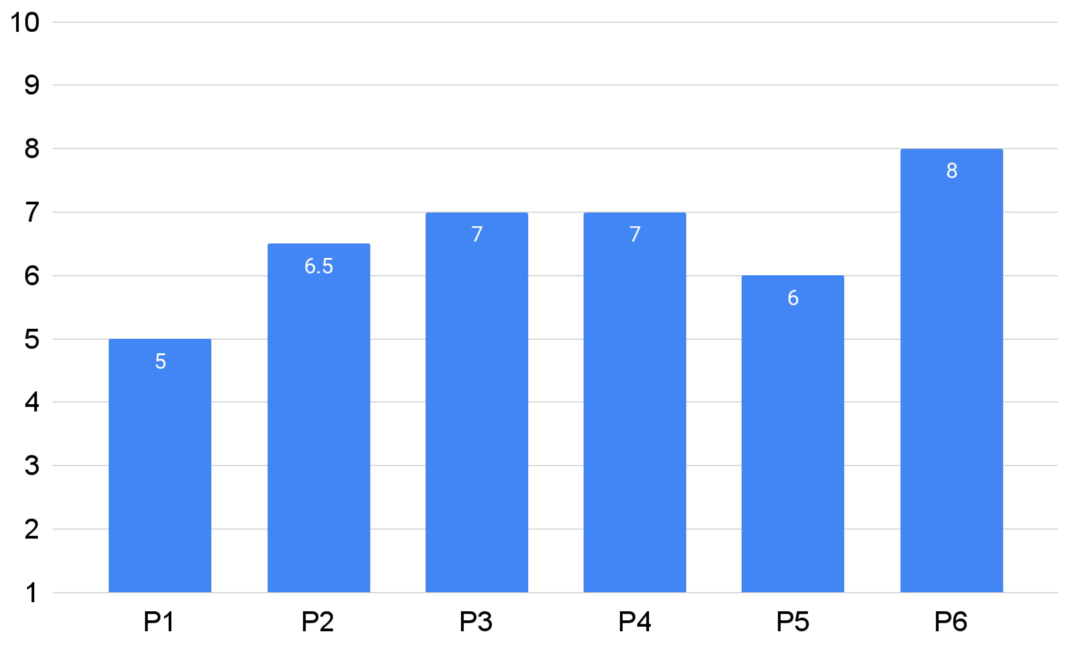
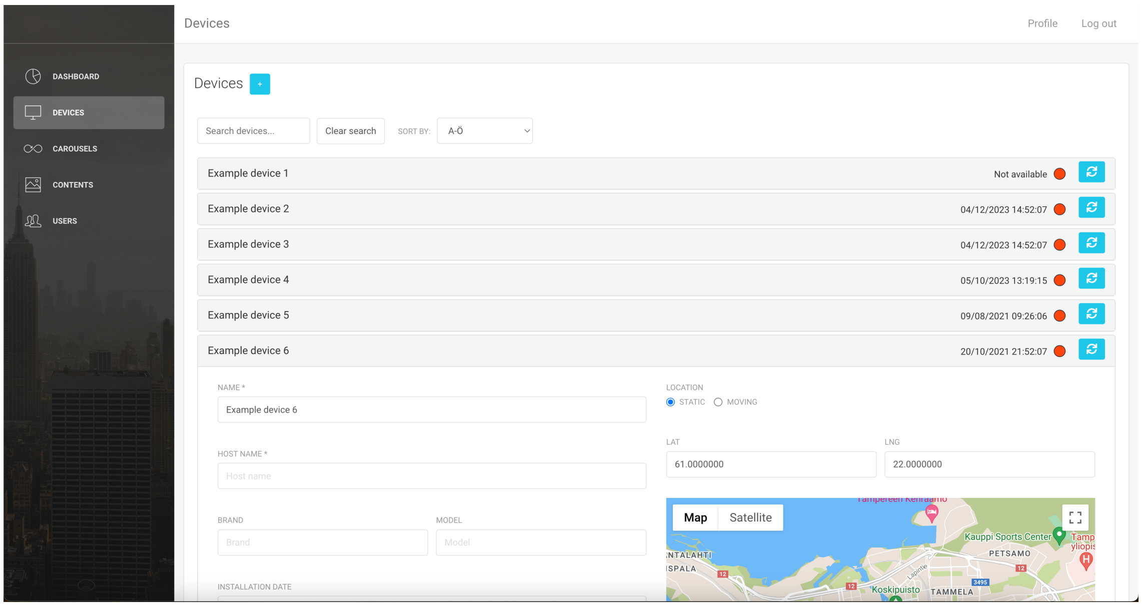


Figure 4.1: Grades for the current implementation

needed a way to group devices as they saw fit and filter the device list view based on those groups. Filtering devices only by device name was not sufficient enough, as not all of the devices have a common naming convention.

Viewing a single device's details was also considered to be troublesome as each device in the list acted like an accordion as can be seen from Figure 4.2. Opening a single device would shift the other devices on the list downwards such as they were no longer visible on the screen. Depending on the device's position on the list, the user would often have to scroll downwards after opening a device to see any relevant information about the device. In some cases, almost no information became visible after opening a device. The UI did not also provide a way to open a single device to its own browser tab and made comparing different devices troublesome.

Insufficient visibility of system status hindered the usability in many parts of the UI. When performing actions, the users typically had to rely on prior experience



SORT BY: A-Ö

Example device 1	Not available
Example device 2	04/12/2023 14:52:07
Example device 3	04/12/2023 14:52:07
Example device 4	05/10/2023 13:19:15
Example device 5	09/08/2021 09:26:06
Example device 6	20/10/2021 21:52:07

NAME *: Example device 6

LOCATION: [✓ STATIC] MOVING

HOST NAME *: Host name

LAT: 61.0000000

LNG: 22.0000000

BRAND: Brand

MODEL: Model

INSTALLATION DATE

Figure 4.2: Device listing with a device opened

rather than recognition. For example, restarting a device did not provide any feedback for the user until the device came back online and updated its status. The lack of visibility of the system status shows particularly well in the time-on-task results seen in Figure 4.3 where the difference between fastest and slowest time was over sevenfold.

In some scenarios the login screen did not provide any feedback to the user, meaning that if the user inserted the wrong credentials, the UI would not respond in any way.

Use of certain colors in varying ways made the purpose of the colors unclear for the users. For example, the color yellow was used to indicate both destructive and non-destructive actions. This clearly violates Nielsen's heuristic of consistency and standards, and can cause additional cognitive load and confusion especially for the less experienced users.

The mobile usability and responsiveness of the application was moderate at best. With different screen sizes, the elements of the UI switch places and often end up

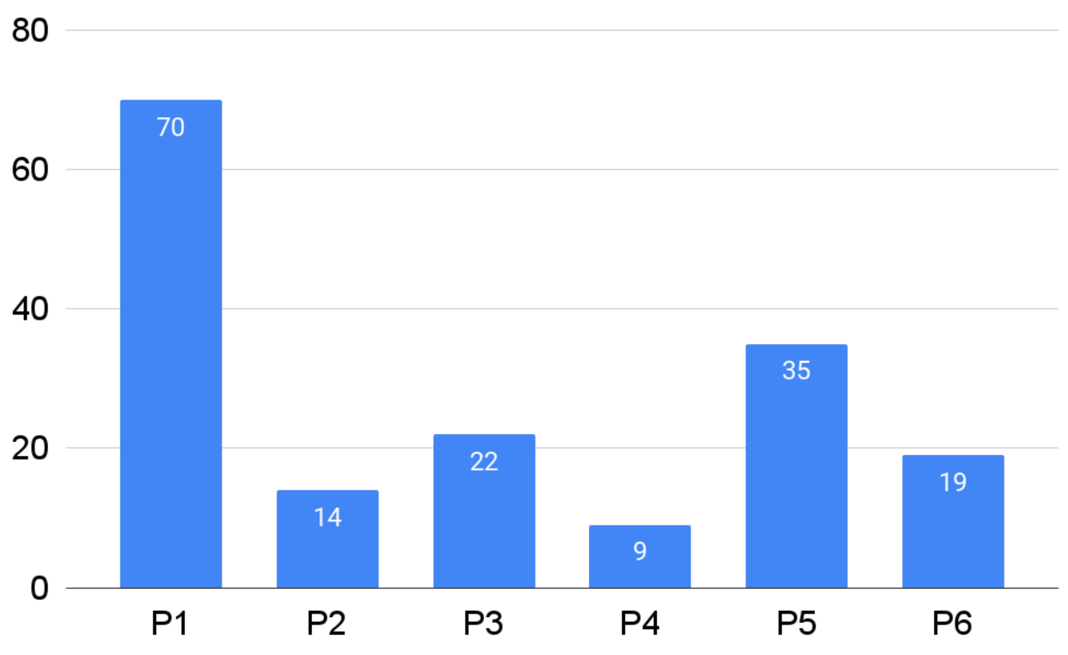


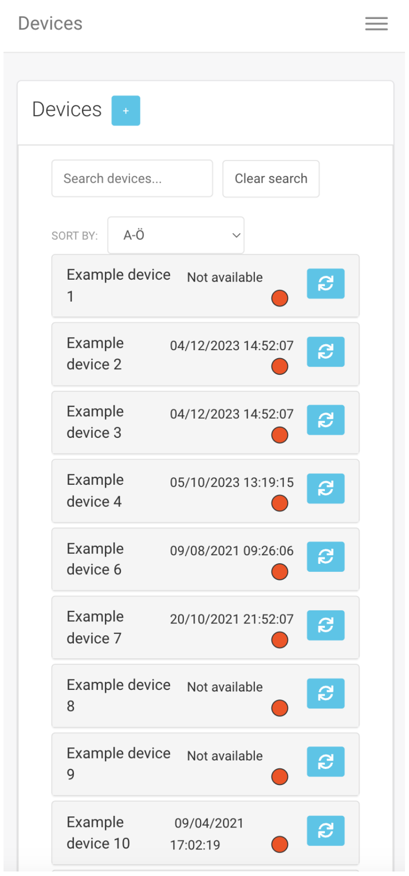
Figure 4.3: Time-on-task task #3, completion times in seconds.

overflowing as can be seen from the Figure 4.4. The mobile view was also crowded with too much information and actions being displayed within a relatively limited screen space. Users suggested that the mobile version should focus more clearly in making the needed data visible rather than allowing for complex operations, such as data modification.

4.3 Feature requests

During the workshop and interviews the participants were asked to present feature requests for the new UI. The scope of the requests was not limited, and the requests that could not be fulfilled in the scope of this project could be selected for further development or at least be documented.

The requests were categorized by implementability and by the impact on the



Devices

Devices +

Search devices... | Clear search

SORT BY: A-Ö

Example device 1	Not available	●
Example device 2	04/12/2023 14:52:07	●
Example device 3	04/12/2023 14:52:07	●
Example device 4	05/10/2023 13:19:15	●
Example device 6	09/08/2021 09:26:06	●
Example device 7	20/10/2021 21:52:07	●
Example device 8	Not available	●
Example device 9	Not available	●
Example device 10	09/04/2021 17:02:19	●

Figure 4.4: Device listing in mobile view

usability of the system, with as many features as possible chosen to be implemented in this project rather than later on. Some of the requests that did not fit the scope of the project were simply too large or required changes in other parts of the system, such as in the backend or the networking layer.

One such feature request that was deemed to be too large for the scope of this project was that the users would have wanted to monitor the device's display in real-time. In the current implementation the users can only take screenshots which can in some situations be hard to interpret. For example, if the device is frozen a screenshot does not communicate that to the user. Although real-time monitoring would have been an useful feature, it would have required too many changes to other parts of the system, such as the backend and the software running on the devices.

The list of feature requests did also contain a lot of requests that could be fulfilled in the scope of the project. Users requested that the device list should update itself automatically so that monitoring the devices would not require constant interaction from the user. Users also wanted the ability to group devices to groups so that managing different types of devices would be more straightforward. Device grouping turned out to be one of the largest new features to be implemented in the new UI.

The current implementation did suffer from a high learning curve, as the users had to know what each button actually did. One of the participants requested that the new implementation should have tooltips and a dedicated help page to assist more inexperienced users with the usage of the system. These features were estimated to be fast and simple to implement but at the same time they were estimated to provide a large increase in the usability of the system and therefore they were given a high priority.

5 New implementation

Based on the data collected from the workshop and interviews discussed in Chapter 4, the new implementation of the user interface aimed to solve the usability problems, as well as improve the overall usability and implement features requested by the participants. Due to the number of feature requests and their complexity, some features requested could not be implemented in the scope of the project, but rather had to be left for later development. The chapter goes through the design, prototyping and implementation phases and discussed how the usability issues discovered earlier were solved.

5.1 Design process

The design process started by establishing basic elements of the new user interface at first. A brand-specific graphical guideline (Figure 5.1) for the colors and typography was already in place, although the color palette needed to be refined as it only included five colors mainly in the shades of purple.

The graphic guidelines dictate that headlines should use *Prompt SemiBold* font while body text should be in *Nunito Sans* font. Although fonts can make or break a design, in this case the provided fonts were neutral and readable. For the font color I decided to choose one of the brand colors, a deep dark purple, as many sources suggest that using pure black on white can cause eyestrain and research has shown that in some cases the difference in contrast can even cause myopia [24].

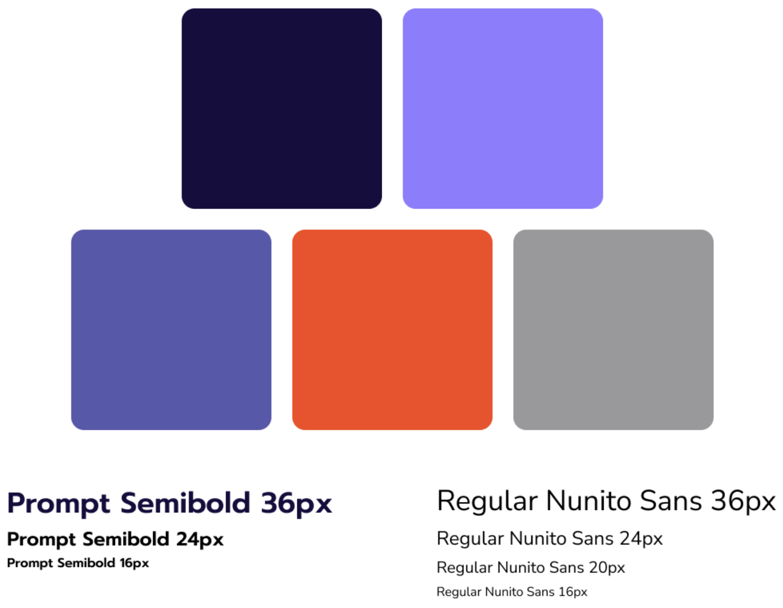


Figure 5.1: Brand colors and fonts used in the project

Other than the graphical guidelines, there were no other constraints for the design set by the customer. As the application is intended for professional use, I decided to steer away from visual appeal and focus the design more on the usability aspect. This decision was backed by the interviews, as many of the participants expressed that the most important aspect of the UI is that it gets the work done efficiently and concisely.

For the design language, I decided to loosely follow Google's *Material Design 2* [25]. This decision was based on the fact that there exists a highly-popular component library for React, MUI [14], that implements Material Design 2. Choosing the newer *Material Design 3* as the design language was also feasible, but the implementation would have been much more time consuming as I would have needed to implement each component in React. MUI not only speeds up the implementation, but it also has a multitude of accessibility features, such as keyboard-only usage, already built in.

After establishing a solid foundation for the design, the first step was to design wireframes that would be shown to the participants by using Figma. The goal was to collect user feedback about the basic layout of the application before going more deeply into other details. The participants were presented with a few different options to choose from and the participants could also suggest improvements based on the wireframes. For example, the participants were presented with a login screen wireframe similar to the current implementation, as well as a different one seen in Figure 5.2. Although the change was not major, the participants liked the idea of seeing relevant notifications that could be of interest to the user logging in.

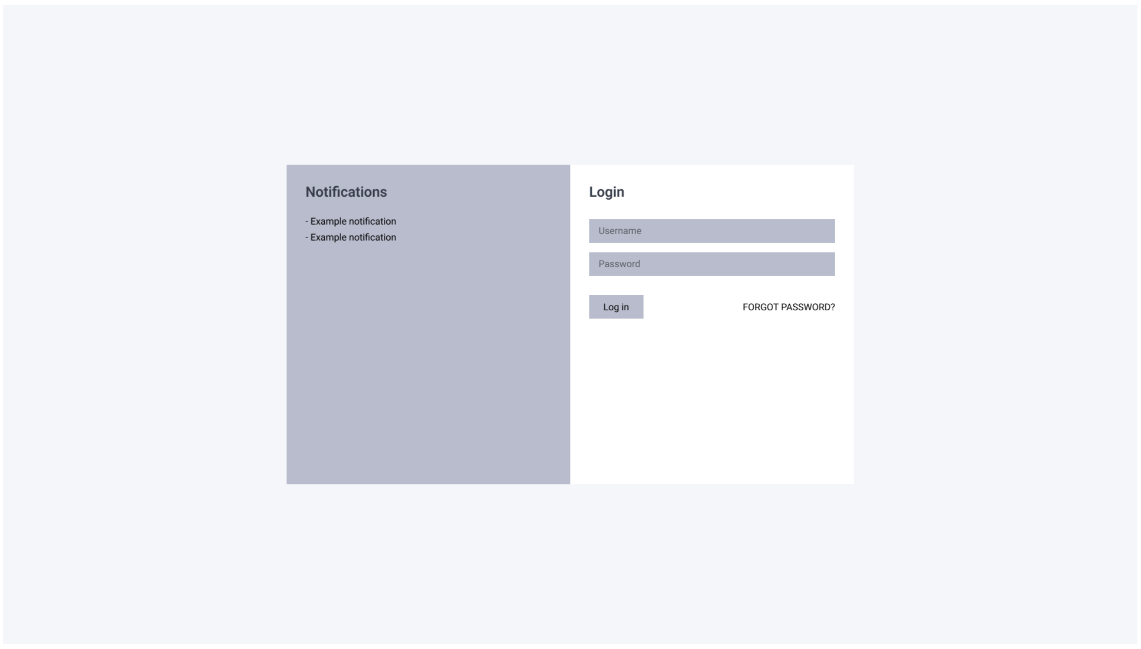


Figure 5.2: Example wireframe of the login page

For the view listing the devices the participants were shown multiple wireframes to choose from, reassembling both wireframes that were identical to the current implementation, as well as wireframes based on the data collected from the interviews.

Figure 5.3 shows an example wireframe of the device listing that the participants felt was able to utilize the available screen space more efficiently as well as display more of the relevant information to the user. One major change was that the device listing stays visible even when interacting with a particular device.

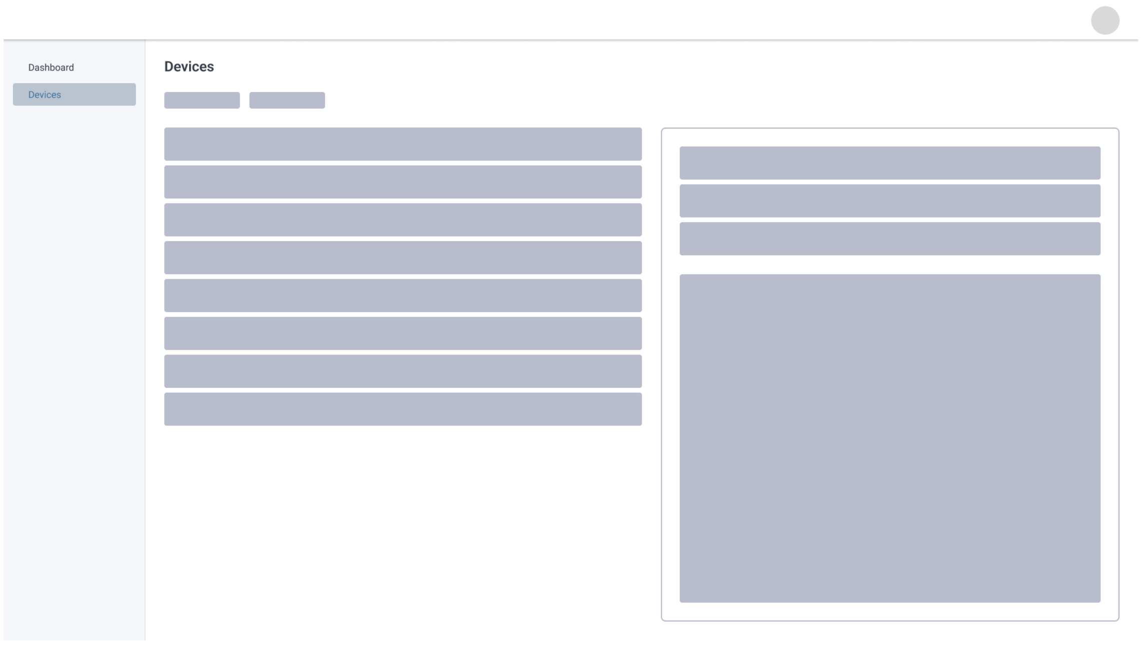
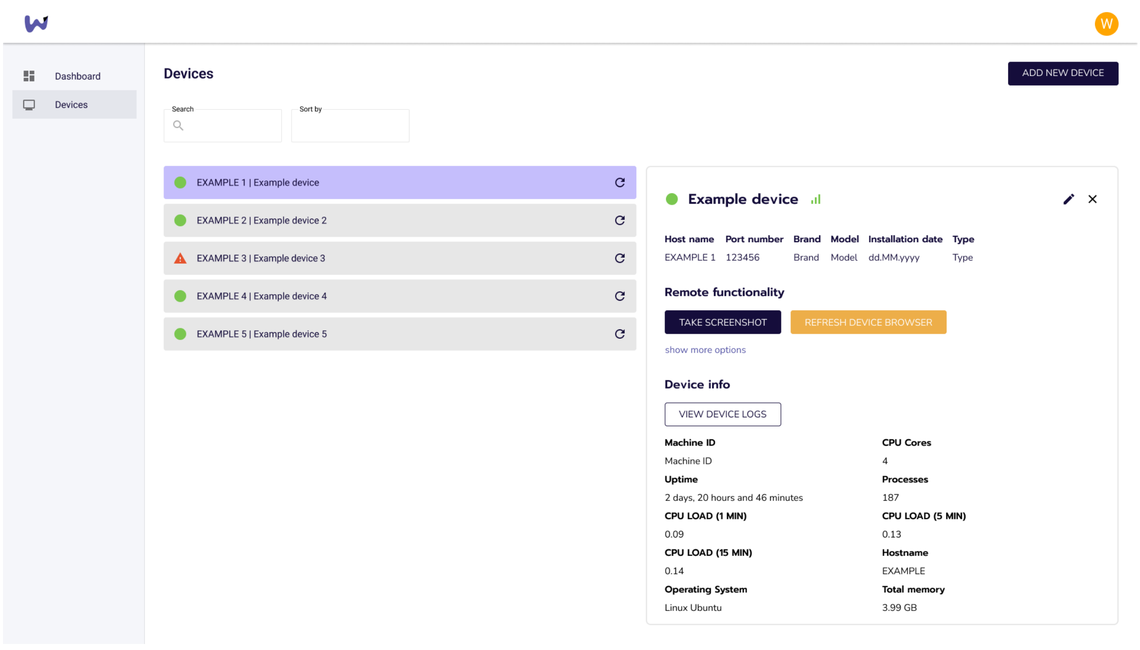


Figure 5.3: Example wireframe of the device listing

Based on the feedback received from the wireframes, more detailed designs were made for the participants to further evaluate. For example, the wireframe from Figure 5.3 was transformed into a more comprehensive design that displayed multiple different features and their respective positions on the UI that can be seen on Figure 5.4.

5.2 Prototyping

The more detailed designs were also used to build a working Figma prototype that mimicked how the user interface would function when actually implemented for the



● EXAMPLE 1 | Example device
● EXAMPLE 2 | Example device 2
⚠ EXAMPLE 3 | Example device 3
● EXAMPLE 4 | Example device 4
● EXAMPLE 5 | Example device 5

● Example device

Host name	Port number	Brand	Model	Installation date	Type
EXAMPLE 1	123456	Brand	Model	dd.MM.yyyy	Type

Remote functionality

Device info

Machine ID	**CPU Cores**
Machine ID	4
Uptime	**Processes**
2 days, 20 hours and 46 minutes	187
CPU LOAD (1 MIN)	**CPU LOAD (5 MIN)**
0.09	0.13
CPU LOAD (15 MIN)	**Hostname**
0.14	EXAMPLE
Operating System	**Total memory**
Linux Ubuntu	3.99 GB

Figure 5.4: Example design of the device listing

browser environment. The prototype consisted of both desktop and mobile versions of the UI and the prototype included basic functionality such as interacting with different devices. For the participants, the user experience was almost browser-like, although some of the elements visible were dummy elements. Under the hood, the prototype consisted of multiple different Figma frames that were linked as a chain with some transitions between the frames. This visualization of how the different frames are linked together in Figma can be seen from Figure 5.5, where each blue line represents a connection between two frames.

The prototype was easily shared to the participants via a link, and the participants could explore the prototype freely when they had the time. The participants were provided with the link, simple instructions of how to use the prototype as well as a request to give feedback when they had explored the prototype.

The prototyping phase was simple to conduct, as the prototype was fast to build and it worked reliably. The feedback received for the prototype did provide some

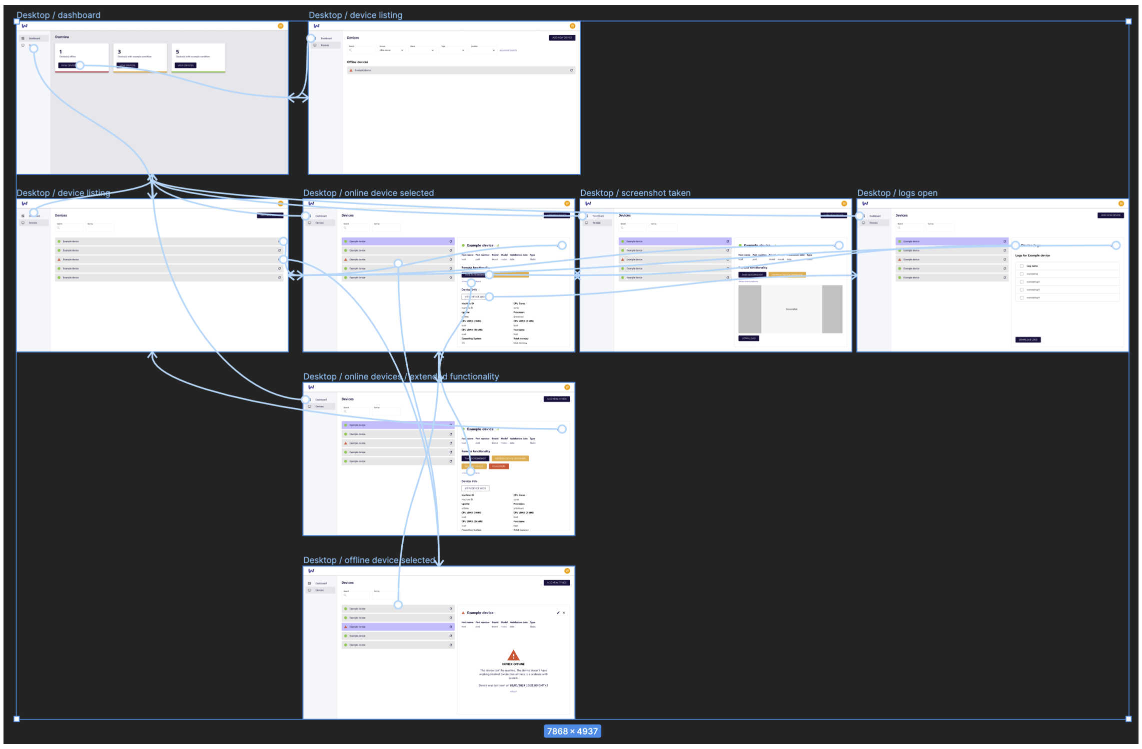

Desktop / dashboard
Desktop / device listing
Desktop / device listing
Desktop / online device selected
Desktop / screenshot taken
Desktop / logs open
Desktop / online devices / extended functionality
Desktop / offline device selected
7868 × 4937

Figure 5.5: Example of Figma prototyping

minor improvement ideas, and more importantly no major usability issues were discovered. The feedback however validated that the prototype was functional and usable and moving on to the implementation phase was justified.

Prototyping presents itself as a valuable tool within the design process by enabling the creation of realistic representations of the end product, facilitating user interaction and providing feedback before beginning of the coding phase. Prototyping can also help test features that depend on changes to other system components, such as the backend, saving valuable time.

5.3 Implementation

Following the completion of the design and prototyping phases, the implementation phase was proceeded to. Utilizing NextJS, a framework for React, made the project setup process straightforward, handling both the initial configuration and related integrations, such as the TypeScript integration. NextJS also has a lot of built-in tooling, such as a compiler and a routing system, that simplify the implementation process.

The backend for the system was already in place, but it required some changes to support new features, such as the device groups. The backend was known to cause at least some difficulties with the new user interface as the limitations of the backend could not be solved just by re-implementing the user interface. Support for device groups was the sole modification to the backend and everything else remained unchanged.

The limitations of backend made the implementation of some features more difficult than anticipated at first. For example, a lot of related data needs to be fetched from multiple endpoints, and the separation of responsibilities is at times frontend-heavy.

5.4 Usability improvements

The four areas of usability issues discovered earlier in Chapter 4 were the following:

1. Managing large amounts of devices from a list view
2. Displaying device details
3. Insufficient visibility of the system status
4. Use of colors in misleading ways
5. Mobile usability and responsiveness

First of all, *the management of a large number of devices* was tried to be solved by both refining the list view as well as implementing a new group feature, which allows users to create both private groups only they can view as well as groups that everyone in the organization can view and manage. Users can add both devices and other groups as subgroups into a group, providing a versatile way of managing a large amount of devices.

Individual devices in the list no longer act like accordions, but instead the view is split in half horizontally, allowing both the device and the device list to remain visible. The user interface remains more stable, as the layout shifts caused by the accordions are no longer an issue.

The current implementation had a lot of issues in regards to *the visibility of the system status.* As the system did not have many of the necessary visual indicators, the users were left waiting for if something was going to happen. To solve the issue, the new implementation introduced two types of visual cues for the user. First, a *skeleton* was used in the parts of the UI that had not yet finished loading the data, as shown in Figure 5.6. By replacing the data with an animated placeholder sized the same as the assumed data, the user is given a cue that something is happening and the system is not frozen. This leads to less frustration while waiting, as the system can be perceived to be responding immediately. The skeletons are typically animated with a pulsating effect to reinforce the user that the system is not frozen. Accurate skeleton sizing is important, as it allows the users to anticipate specific elements to load correctly and minimize layout changes for improved usability and user experience.

The other improvement to increase the visibility of the system status was the use of toast notifications. Toast notifications are notification boxes that appear on top of the UI, typically in one of the corners. Toast notifications can change state as seen in Figure 5.7, so when the user, for example, performs an action, the notification first appears to inform the user that an action is being executed and after the action is executed, the state changes to inform the user about the outcome of the action.

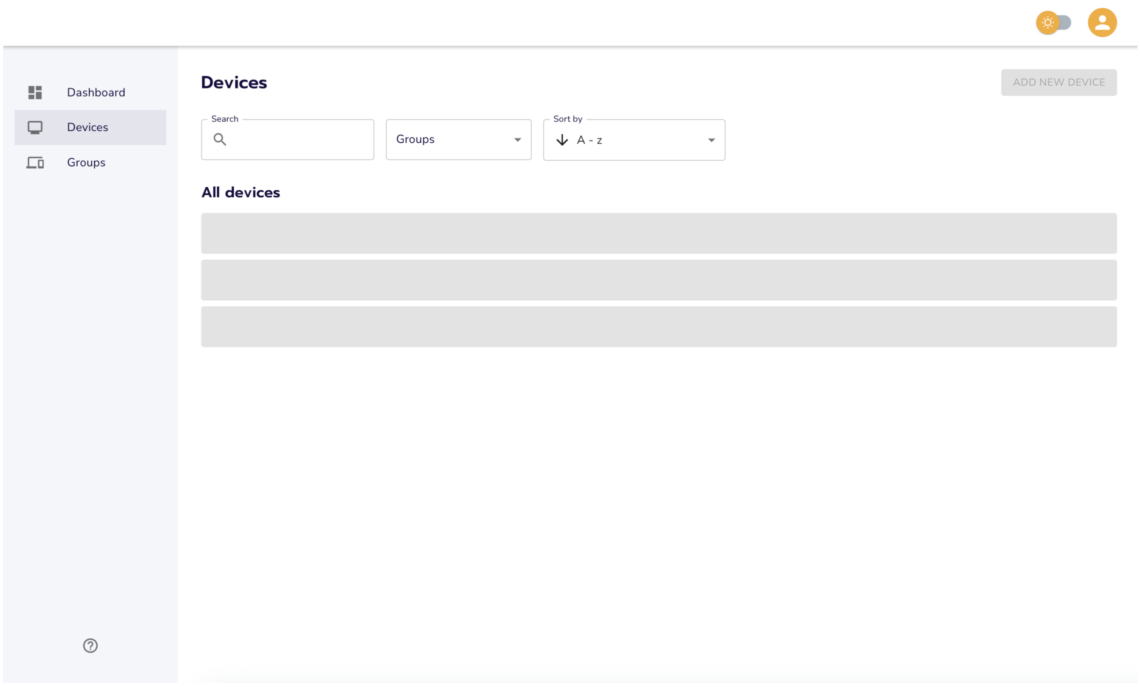


Figure 5.6: Example of multiple skeletons indicating a loading state of the assumed device list

Toast notifications can contain text as well as other components, such as icons, to clearly indicate the outcome to the user. The visibility of the notifications can also be configured, for example, an error notification can remain on the screen until the user interacts with it, whereas a success notification can disappear after a certain time period.

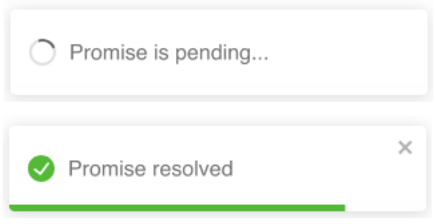


Figure 5.7: Example of a toast notification

To solve the issue with colors, the new UI was designed with clear use cases for different colors to prevent mismatch between the UI and the user's mental model. For example, the color red is reserved for errors and negative actions, whereas the color yellow is used to indicate actions that are non-destructive but still require user attention.

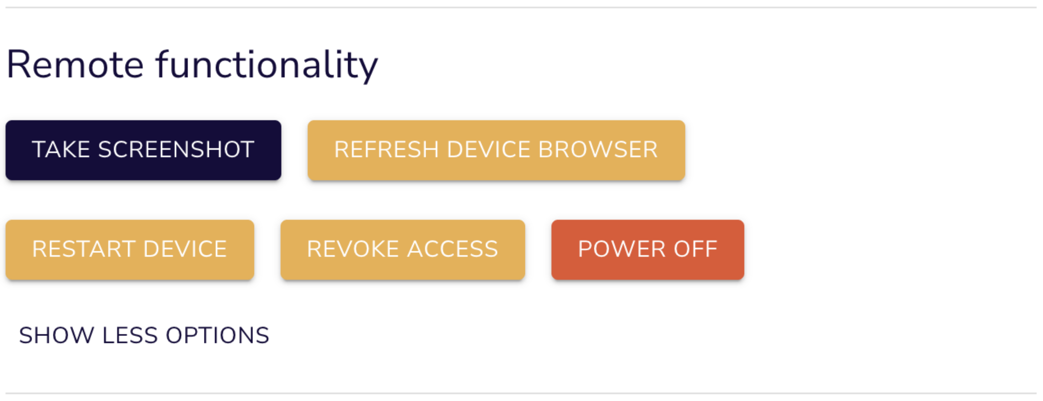
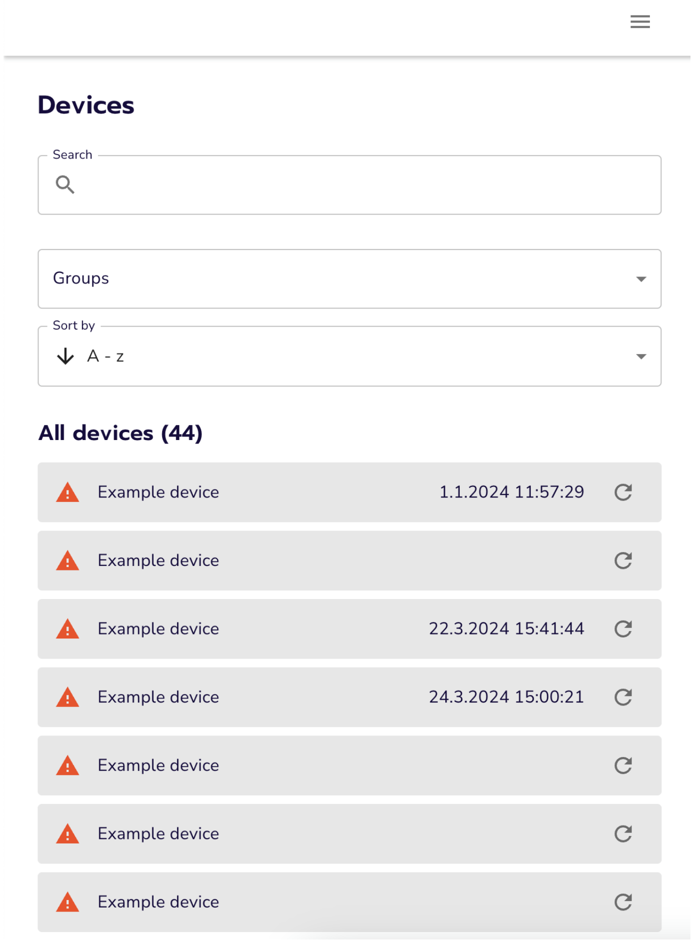


Figure 5.8: Example of color usage in the new implementation

Figure 5.8 demonstrates how color is used to indicate the severity of each action for the users. In this example, taking a screenshot does not have any effect outside the system or affect other users or devices. Buttons colored in yellow initiate actions that can have effects for others as well, for example, restarting the device will cause the device to become temporarily unavailable both in the field as well as in the system. Powering off the device is colored red because in this case it will turn the device off completely and require physical presence to turn the device back on. In addition to indicating the severity of actions with color, some of the options were hidden so that users have to take extra steps in order to reveal them.

Mobile usability was a major usability issue in the current implementation. Even though the application is intended to be used primarily on desktop, there exists a number of use cases for mobile as discovered earlier. In order to make the application usable on both desktop and mobile environments, many parts of the UI had to work differently based on the screen size.


Devices

Search

Groups

Sort by A - z

All devices (44)

Example device 1.1.2024 11:57:29

Example device

Example device 22.3.2024 15:41:44

Example device 24.3.2024 15:00:21

Example device

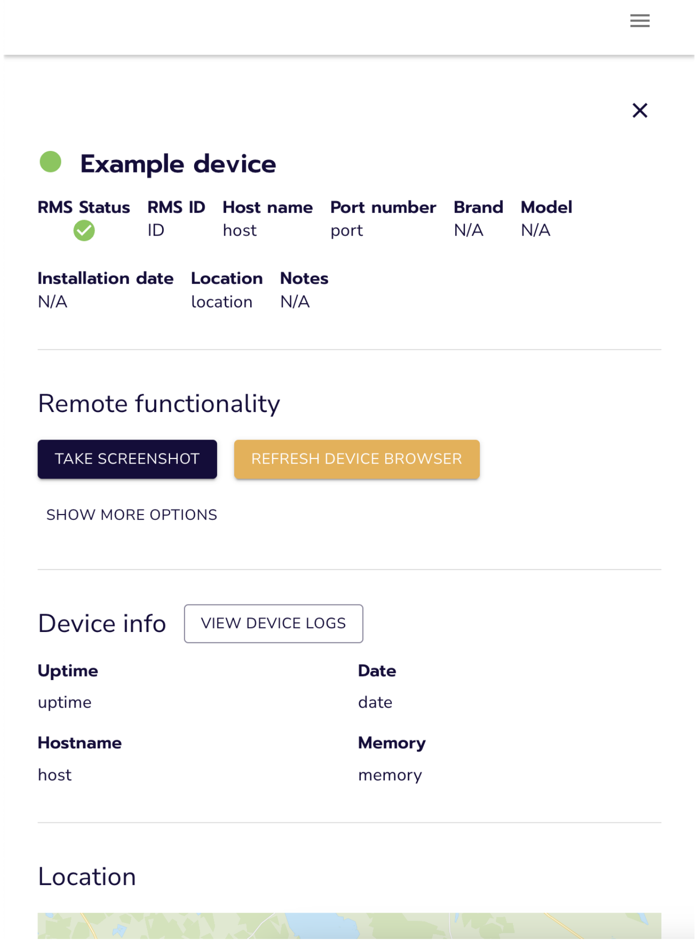
Example device

Example device

Figure 5.9: Example of mobile responsiveness in the new implementation

For example, showing both the device list and the device itself side-by-side on a mobile size screen was not feasible. Optimizing the screen space usage is possible because the desktop and mobile versions differ from each other feature-wise. Figure 5.9 shows the new mobile-optimized device list view. Compared to the earlier implementation, the elements of the UI no longer shift to wrong places and the elements are sized so that they can be more easily controlled by touch. The filters are distinguished from the device list by whitespace, making them stand out and easier to work with.

Instead of showing the device list and device details side-by-side when a device is open, opening a device on mobile hides the device list and opens the device details in full screen as seen in Figure 5.10. This way, the available screen space is optimized to show only the necessary information, rather than attempting to cram everything onto a limited screen.



Example device

RMS Status	RMS ID	Host name	Port number	Brand	Model
✓	ID	host	port	N/A	N/A

Installation date	Location	Notes
N/A	location	N/A

Remote functionality

Device info

Uptime: uptime
Date: date
Hostname: host
Memory: memory

Location

Figure 5.10: Example of mobile responsiveness in the new implementation

The mobile version of the application is designed for on-the-field usage, primarily to control a single device. Therefore, the design had to be simple without compromising essential features. The mobile version works in the browser just like the desktop version and the screen size of the device dictates which version the user is served.

Overall, the UI on the mobile version was designed and implemented so that the user is presented with only the necessary information and features. Scrolling was limited to the y-axis to make the usage experience similar to other mobile applications. Some of the features available on the desktop version, such as adding a new device, are disabled on the mobile version, as they were deemed to be too error-prone on a limited screen. Implementing only the features that will most likely be used on the mobile version also saved time on design and implementation phases and contributed towards the mobile UI being as clean and easy to use as possible.

6 Analysis

In order to validate the usability improvements of the new implementation, the participants were asked to answer the System Usability Scale questionnaire twice, once for both the current and the new implementation. Additionally, the participants were asked to complete four tasks by using both implementations.

For the current implementation, the participants were asked to complete the timed tasks and the SUS questionnaire was given to the participants as a link to be filled at a convenient time. It was discovered later on that this caused extra work, as participants usually forgot to fill out the questionnaire and multiple reminders had to be sent out. Therefore when scheduling times for assessing the new implementation, the time for filling the questionnaire was accounted for. It is worth noting that answering the SUS questionnaire took participants less than five minutes.

The participants were asked to answer the ten statements of the System Usability Scale for both of the implementations. The results for both the current implementation and the new implementation are listed on Table 6.1.

It can be seen from Table 6.1 that the weighted SUS score for the current implementation has a major variation between participants. For example, the highest individual score was 95, meaning best imaginable usability, whereas the lowest score was 25, indicating major usability issues. The difference between the highest and the lowest score was 70 points. The lowest score deviated from the average with 31.7 points, whereas the highest score deviated from the average with 38.3 points.

Table 6.1: Weighted SUS score for both implementations

Participant	Current weighted score	Grade	New weighted score	Grade
P1	72.5	C+	85	A+
P2	55	D	95	A+
P3	95	A+	97.5	A+
P4	35	F	100	A+
P5	25	F	90	A+
P6	57.5	D	95	A+
AVERAGE	56.7	D	93.75	A+

The reason for the variation was not evidently clear. One reason could be that participants who have used the system for longer periods of time have familiarized themselves with the system and overlook the most common usability issues. The data also suggests that using System Usability Scale for assessing already existing systems might cause variation, as SUS is mainly intended to assess systems that users have limited experience with, although this was an acknowledged drawback.

The average SUS score for the current implementation was 56.7, meaning it is 11.3 points below the SUS average of 68. The score is in line with the data gathered from the interviews indicating that there exists usability issues within the system.

The new implementation had much more uniform scoring between the participants. Even the lowest score for the new implementation was 85, surpassing the SUS average of 68, while the highest score was 100. The difference between the highest and the lowest score was only 15 points. Deviations from the average were 8.75 points for the lowest and 6.25 for the highest score. All of the participants had used the system approximately the same time before answering the questionnaire.

The average SUS score improved from 56.7 to 93.75, meaning an increase of

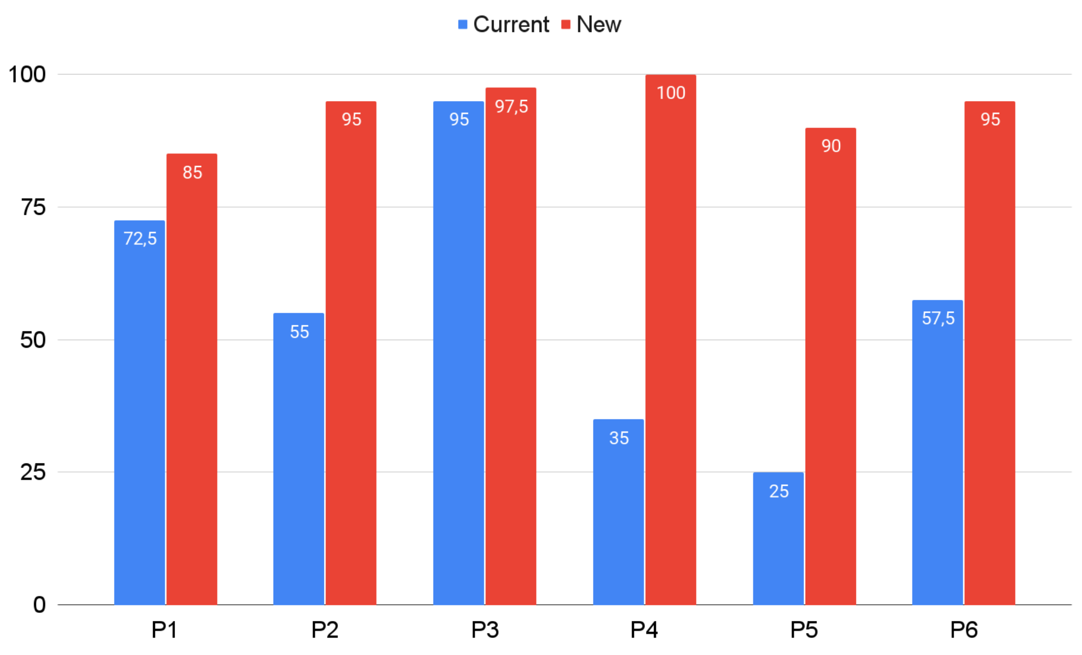


Figure 6.1: Comparison of SUS score between the current and the new implementation

37.05 points or 65% respectively. When comparing against the SUS average of 68, the current implementation was 11.3 points below, whereas the new implementation was assessed to be 25.75 points above the average. These results indicate a major increase in the perceived usability of the system when using the new implementation. The current implementation had an average SUS grade of D, whereas the new implementation had an average grade of A+. The new implementation received A+ from all of the six participants, whereas the current implementation received only one A+ grade and the others were C+ or worse.

It can be seen from Figure 6.1 that participants P2, P4 and P5 had the lowest SUS score to begin with. Participants P2, P4 and P5 were in fact the 3 system users mentioned earlier. These participants also use the system the most so it is logical that usability issues hinder the usage experience of those who use the system a lot.

Conversely, improvements to the system also increase the perceived usability most amongst those participants.

Altogether the recorded SUS scores were uniform with the results gathered from the interviews. The current implementation had a number of usability issues that the new implementation aimed to solve, and the change was directly reflected in the SUS score.

The participants also completed four timed tasks within the system in a controlled environment. The task execution was monitored through Microsoft Teams by both voice and screen share. The tasks the participants were asked to perform were the following:

- Task 1: Check a given device's last seen timestamp. The task ends when the participant reads the timestamp out loud.
- Task 2: Download a given log file from a given device. The task ends when the browser indicates a file is being downloaded.
- Task 3: Restart a given device. The task ends when the participant indicates a restart is done.
- Task 4: Report the number of devices that belong to a given group. The task ends when the participant reports the correct answer.

The tasks were timed on a timekeeping software, and the results were rounded to the nearest second to account for any delay as the tasks were monitored through Microsoft Teams.

Figure 6.2 shows the time-on-task results for the current implementation. From this figure it is visible that the time each of the tasks took to complete by different participants had major variation. To analyze the results more in-depth, the results are displayed in Table 6.2

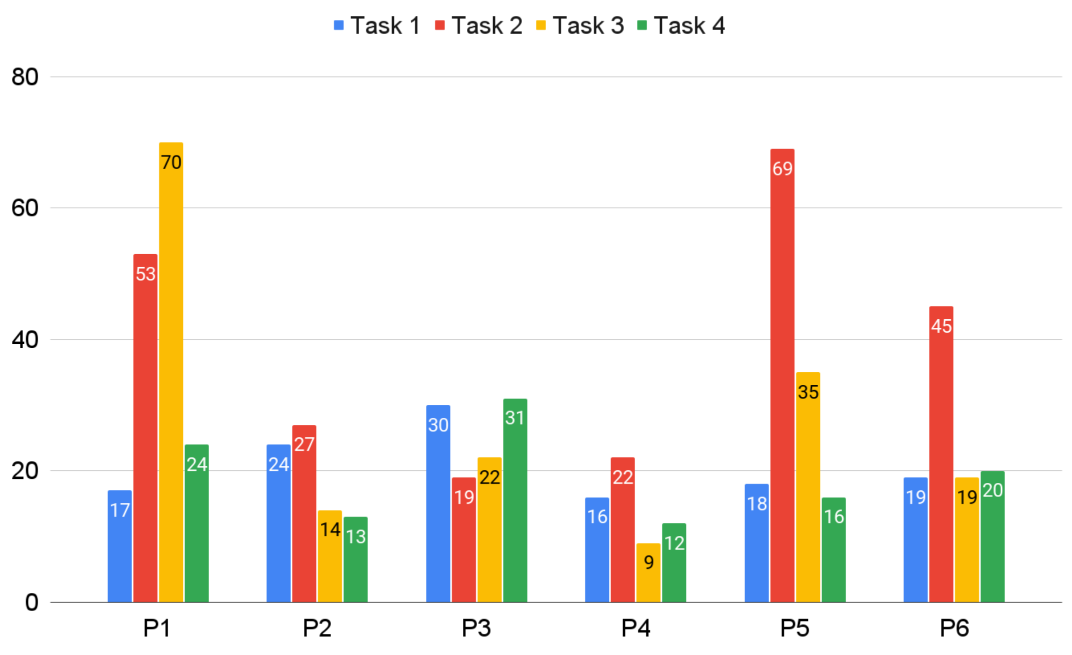


Figure 6.2: Time-on-task distribution for the current implementation. Completion times in seconds.

The time-on-task data for the new implementation shown in the Figure 6.3 shows a more unified completion time distribution, although some tasks still had some variation. It is worth mentioning here that the participants had a limited experience with the new implementation.

For task 1, the average time actually increased by 5 seconds or 24%. In task 1 the users had to report the last seen timestamp for a device, and the data was displayed in approximately the same place in both implementations. In hindsight, the task description should have been more clear, as many of the participants first proceeded to the device details searching for a field named 'last seen' before actually reporting the timestamp from the device listing.

When looking at the data, participant 5 (P5) stands out from the rest, as the completion time for P5 is considerably slower (43 seconds) than for other partici-

Table 6.2: Completion times for the current implementation (seconds). Averages are rounded to the nearest second.

Participant	Task 1	Task 2	Task 3	Task 4	TOTAL
P1	17	53	70	24	164
P2	24	27	14	13	78
P3	30	19	22	31	102
P4	16	22	9	12	59
P5	18	69	35	16	138
P6	19	45	19	20	103
AVERAGE	21	39	28	19	107

Table 6.3: Completion times for the new implementation (seconds). Averages are rounded to the nearest second.

Participant	Task 1	Task 2	Task 3	Task 4	TOTAL
P1	34	36	19	15	104
P2	26	24	17	22	89
P3	24	20	21	21	86
P4	15	16	13	27	71
P5	43	63	32	18	156
P6	11	28	17	20	76
AVERAGE	26	31	20	21	97

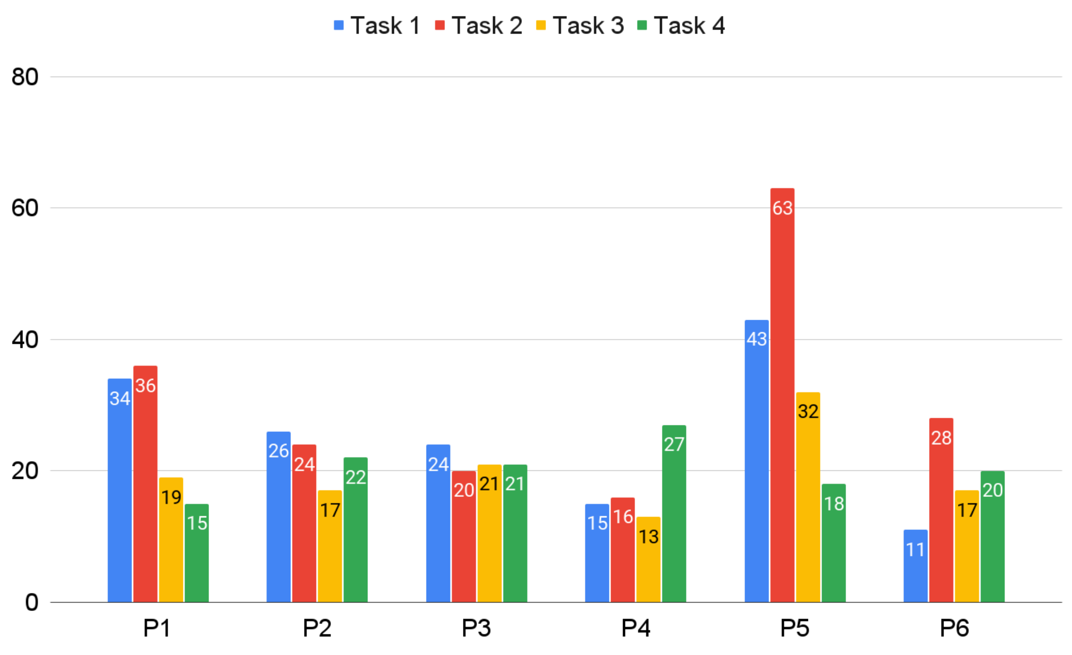


Figure 6.3: Time-on-task distribution for the new implementation. Completion times in seconds.

pants. If the P5 result were excluded when calculating the average for task 1, the average would be 22 seconds, meaning only a 1 second increase, which could have been explained by margin of error.

For task 2 on the other hand the average time improved by 8 seconds or 21%. Although the task completion time did improve, a bigger improvement was expected due to the fact that the current implementation lacked search functionality, whereas the new implementation had it. It was observed though that not all of the users actually used the search functionality, which might explain why there was only a slight improvement.

In task 3, an improvement of 8 seconds or 29% was seen, with the average decreasing to 20 seconds from 28 seconds. The reason for the improvement was that the new implementation clearly indicated to the user that the given device was restarted. From the data it can be seen that the current implementation had a slowest completion time of 70 seconds, whereas the new implementation had the slowest completion time of only 32 seconds. This means that the slowest time was improved by 38 seconds or 54%.

In task 4 a slight increase of 2 seconds or 11% was seen in the completion time. The new implementation had the functionality to filter devices based on their group, but as this was new functionality, it was observed that none of the users actually took advantage of it.

For total task completion times, the average improved by 10 seconds or 9% as the average completion time decreased from 107 seconds to 97 seconds. In the current implementation, four of the participants had a total completion time exceeding 100 seconds, whereas in the new implementation only two of the participants exceeded 100 seconds.

As a whole, the results from the timed tasks were not as good as expected, although they still indicate that the usability and the efficiency has increased as the total completion time did improve. This indicates that the usability issues of the current implementation were not as severe as the users initially suggested. One of the reasons why the results did not improve more than 9% was that the tasks the users typically perform in the system are quite simple, and as seen in the results the tasks take on average less than a minute to complete. One possible option would have been to artificially increase the task length by grouping multiple tasks into a single task, but that was not seen as a valid option. The tasks that the participants completed for this thesis are tasks that they perform on a regular basis when using the system. Therefore, it can be concluded that the time-on-task does provide useful results but the analysis can be difficult to conduct especially if the completion times are short to begin with. As the sample size was relatively small, only consisting of six participant, the results must be viewed as an overview and it is difficult to draw any statistical conclusions from the results.

7 Conclusion

Chapter 1 discussed the reasons why this thesis was conducted and why the topic of user interface improvement in collaboration with users was chosen. Chapter also contained the research questions that the thesis wanted to find answers for. Finally, the structure of the thesis was presented.

Chapter 2 briefly introduced user-centered design (UCD) and both the research and evaluation methods used in the thesis. Focus groups and semi-structured interviews were used to collect both qualitative and quantitative data from the participants and Nielsen heuristics were used as a foundation to discover usability issues when conducting the focus group and interviews.

Finally, the chapter briefly introduced the tools, guidelines and platforms that were used in various parts of the thesis. Figma was utilized as the main design and prototyping tool, whereas Microsoft Teams was used to conduct interviews and focus groups remotely.

In Chapter 3 the technical aspects related to the development of the new system were explored. The use of JavaScript, its superset TypeScript, and React was justified. TypeScript and React were used due to their popularity in modern web development, as well as their support for third party libraries that simplify the implementation of complex functionalities. Chapter 3 also discussed the use of Docker, a container application allowing the deployment of applications virtually on any platform. Finally, the chapter explained how Google Cloud Platform, Git and GitHub can be used together with Docker to deploy the application to the cloud.

In Chapter 4 the current implementation was introduced briefly and the research was done to discover the usability issues with the implementation. The six participants were interviewed one-by-one in semi-structured interviews where the participants were asked 10 questions related to the user interface. Based on these interviews, a list of usability issues that were to be solved was formed. The participants were also given a possibility to provide feature requests for features they would like to see in the new implementation.

Chapter 5 went through the design and implementation processes, as well as discussed how the usability issues discovered in Chapter 4 were solved.

Chapter 6 consisted of analysis of the results gathered from Chapter 4 and Chapter 5. Both the System Usability Scale and time-on-task results were compared between implementations and the end results were presented. The chapter also discussed possible reasons why the results were not in some parts as expected.

7.1 Answers to research questions

Three research questions were addressed in the thesis:

> *RQ 1: How can the user interface be improved in collaboration with current users?*

Research question 1 was answered in Chapter 2, in which the semi-structured interviews and focus groups were used as means to improve the user interface in collaboration with users. Chapter 4 discussed how these methods were used to gather the data needed and also how some of the feature requests had to be left out. Chapter 5 explored how the users can be involved in the actual design process through the means of prototyping.

Focus groups and semi-structured interviews turned out to be successful methods to involve users in the process. The methods required no special skills or preparation from the participants, were able to be conducted remotely and provided useful data for improving the user interface. Focus groups helped to discover usability issues, whereas the interviews provided more in-depth descriptions of the issues experienced.

Prototyping provided the necessary feedback early-on in the design process. The feedback helped save time, as wireframes were relatively quick to design and present to the participants. This approach ensured that layouts and features that the participants disliked or did not need were filtered out before too much time was used to design them further.

RQ 2: What are the metrics that can be used to assess the current and the new implementation of the user interface?

Metrics chosen to assess the current and new implementation were the System Usability Scale (SUS) and time-on-task that both provide quantitative data that is suitable for analysis. SUS provided data about the perceived usability of the system, whereas the time-on-task provided data about how quickly different tasks can be performed on the system. The tasks were relatively simple and quick to complete, which proved to make the analysis more difficult.

RQ 3: How can the end result of the improvement process be validated using the metrics chosen?

For research question 3 it can be concluded that the improvement was validated by the chosen metrics. Improvements in the SUS score indicate that the perceived usability has risen drastically whereas improved time-on-task scores suggest that the efficiency of performing tasks has increased. Therefore it can be determined that the system was more usable after the new user interface was implemented and brought into operation. The perceived usability score measured by SUS improved by 65%, whereas with time-on-task a total completion time improvement of 9% was seen.

7.2 Limitations and future research

While the results needed were collected, the project's scope and participant pool were limited. One of the issues that emerged during the interview phase was scheduling interviews with the participants, which proved to be a time-consuming process. For instance, finding a suitable time for an interview took weeks or months from the initial contact. Other limitations existed due to limited resources, such as time, which meant not all user requests could be fulfilled.

The small sample size of six participants presents an opportunity for future research to explore whether increasing the participant pool size would yield different results or introduce new challenges not present in this thesis.

As of now the research was conducted with the participants having limited experience with the new implementation. Possible future research could be done to see if the results change over time when the users familiarize the new user interface.

References

[1] *What is User Centered Design? — updated 2023 — interaction-design.org*, `https://www.interaction-design.org/literature/topics/user-centered-design`, [Accessed 11-12-2023].

[2] C. Wilson, *Interview Techniques for UX Practitioners.* Elsevier, 2014. DOI: `10.1016/c2012-0-06209-6`. [Online]. Available: `https://doi.org/10.1016/c2012-0-06209-6`.

[3] *10 Usability Heuristics for User Interface Design — nngroup.com*, `https://www.nngroup.com/articles/ten-usability-heuristics/`, [Accessed 21-09-2023].

[4] *Heuristic Evaluations: How to Conduct — nngroup.com*, `https://www.nngroup.com/articles/how-to-conduct-a-heuristic-evaluation/`, [Accessed 21-09-2023].

[5] B. Still and K. Crane, *Fundamentals of User-Centered Design.* CRC Press, Aug. 2017. DOI: `10.4324/9781315200927`. [Online]. Available: `https://doi.org/10.4324/9781315200927`.

[6] J. Brooke, “Sus: A quick and dirty usability scale”, *Usability Eval. Ind.*, vol. 189, Nov. 1995.

[7] A. S. for Public Affairs, *System Usability Scale (SUS) | Usability.gov — usability.gov*, `https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html`, [Accessed 27-11-2023].

[8] P. Jeff Sauro, *5 Ways to Interpret a SUS Score &x2013; MeasuringU — measuringu.com*, `https://measuringu.com/interpret-sus-score/`, [Accessed 01-06-2024].

[9] T. Tullis and B. Albert, *Measuring the user experience: Collecting, analyzing, and presenting usability metrics*. Elsevier/Morgan Kaufmann, 2013.

[10] *Figma: The Collaborative Interface Design Tool — figma.com*, `figma.com`, [Accessed 12-11-2023].

[11] *Video Conferencing, Meetings, Calling | Microsoft Teams — microsoft.com*, `https://www.microsoft.com/en-us/microsoft-teams/group-chat-software`, [Accessed 12-11-2023].

[12] W. W. A. I. (WAI), *WCAG 2 Overview — w3.org*, `https://www.w3.org/WAI/standards-guidelines/wcag/`, [Accessed 28-11-2023].

[13] *WebAIM: Contrast Checker — webaim.org*, `https://webaim.org/resources/contrastchecker/`, [Accessed 28-11-2023].

[14] *MUI: The React component library you always wanted — mui.com*, `https://mui.com/`, [Accessed 01-12-2023].

[15] *JavaScript | MDN — developer.mozilla.org*, `https://developer.mozilla.org/en-US/docs/Web/JavaScript`, [Accessed 12-12-2023].

[16] *ECMAScript 2023 Language Specification — 262.ecma-international.org*, `https://262.ecma-international.org/14.0/`, [Accessed 12-02-2024].

[17] *Usage Statistics of JavaScript as Client-side Programming Language on Websites, January 2024 — w3techs.com*, `https://w3techs.com/technologies/details/cp-javascript`, [Accessed 14-01-2024].

[18] *Stack Overflow Developer Survey 2023 — survey.stackoverflow.co*, `https://survey.stackoverflow.co/2023`, [Accessed 21-09-2023].

[19] *React — react.dev*, `https://react.dev/`, [Accessed 11-12-2023].

[20] *GitHub - facebook/react: The library for web and native user interfaces. — github.com*, `https://github.com/facebook/react/`, [Accessed 12-02-2024].

[21] *Docker documentation*, `docs.docker.com`, [Accessed 3-1-2024].

[22] *Cloud Computing Services | Google Cloud — cloud.google.com*, `cloud.google.com`, [Accessed 13-12-2023].

[23] *GitHub: Let's build from here — github.com*, `https://github.com/`, [Accessed 05-03-2024].

[24] A. C. Aleman, M. Wang, and F. Schaeffel, "Reading and myopia: Contrast polarity matters", *Scientific Reports*, vol. 8, no. 1, Jul. 2018, ISSN: 2045-2322. DOI: `10.1038/s41598-018-28904-x`. [Online]. Available: `http://dx.doi.org/10.1038/s41598-018-28904-x`.

[25] *Material Design — m2.material.io*, `https://m2.material.io/`, [Accessed 01-12-2023].