Isak R. Gjeitsund

Menntor

Navigating internal documentation

Bachelor's thesis in Web Development
Supervisor: Gioele Barabucci
May 2023

Bachelor's thesis

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

NTNU
Norwegian University of
Science and Technology

Isak R. Gjeitsund

Menntor

Navigating internal documentation

Bachelor's thesis in Web Development
Supervisor: Gioele Barabucci
May 2023

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

NTNU
Norwegian University of
Science and Technology

ABSTRACT

Title: Menntor: Navigating internal documentation

Date: 15.05.2023

Participants: Isak R. Gjeitsund

Supervisor: Gioele Barabucci

Employer: Mennt AS

Keywords: Web development, design thinking, sustainability, universal design, component-driven development, documentation, information architecture

Number of pages: 109

Number of attachments: 9

The project described in this thesis aims to create a comprehensive documentation system for Mennt AS, a web development company based in Elverum, Norway. The employees at the company had some issues with managing and organizing their internal documentation, which led to time-consuming searches for information, redundant work, and decreased productivity. The project team was tasked with building a custom organization system that streamlines documentation management and retrieval. Through the use of the Design Thinking framework, the research in this thesis seeks answers to how we can ensure usability, efficiency, and sustainability in the web solution.

PREFACE

I would like to thank Mennt AS for letting me work on this project and giving me the freedom to choose the direction of the project. I would also like to thank my design team at the company for the work they put into it.

My partner, friends, and family have been an incredible support in these stressful months. I appreciate all the help, kind words, and proofreading.

Lastly, I would like to thank my thesis advisor, Gioele Barabucci, for his patience and helpful guidance.

Drammen, 14th of May 2023.

ISAKR.G.

TABLE OF CONTENTS

1 INTRODUCTION

1.1 Project description

Building a company can be challenging, and only 26.5% of all newly established companies in Norway survive for more than 5 years (Statistisk Sentralbyrå, 2019). Many companies struggle to efficiently manage and organize their internal documentation, which can lead to time-consuming searches for information, redundant work, and decreased productivity.

The web development project described in this thesis aims to address these issues by creating a comprehensive documentation system for the web development company Mennt AS that is easily accessible, searchable, and user-friendly. The system will streamline documentation management and retrieval, reducing the amount of time and effort required to find critical information and enabling employees to focus on more productive tasks.

1.1.1 BACKGROUND

Mennt AS is a company based in Elverum that designs and delivers web solutions to other businesses, mainly eCommerce websites built using the DynamicWeb Business Suite. The company was founded in 2020 and is still a fairly small company, but they are growing quickly. Their staff currently includes marketers, designers, and developers.

The author of this thesis had worked with Mennt previously in a part-time position and heard some of the other employees expressing discontent with the company's current solutions for keeping track of the time they spend on projects and other work. The systems in place were described as complicated, time-consuming, and annoying. The manager of the company was contacted to see if there was something that could be done to help them with the challenges that the employees were having with the system. Mennt agreed to let me work on this project, and together we set up a project team that would collaborate on figuring out a solution.

Mennt usually has several client projects happening at the same time, and being a newly founded company, they have limited amounts of time and resources to get everything done. They asked the project team to build a custom organization system for their employees. The main focus at the start of this project was to find a good solution to the issues the employees were having with timekeeping and keeping track of all the different systems, however, the system should be expandable, so that Mennt can add more features to it later.

1.1.2 SOLUTION

After the project team did some research within the company, we found that the company is frequently testing new software solutions, and having to adapt to new technologies was often frustrating for their employees. They spend a lot of time asking other employees at the company for help – wasting valuable time that could be spent elsewhere.

The company was using two different systems for tracking the hours spent on projects. They use several software solutions created by Microsoft, including Azure DevOps, Teams, and Outlook. In Azure DevOps, they have set up 7pace Timetracker to allow the employees to track the time they use on projects that are hosted there. Some employees also used ClickUp, a project management system, to register the hours they work.

Most of the company's internal and external meetings are planned and held using Teams, and most of the task planning happens in Azure DevOps Boards. In addition to this, the designers and developers use software and technologies like Figma, and there are a handful of technologies being used in the development of the projects they work on for their clients.

At the start of the project, the project team was highly focused on the timekeeping issue that the company was having. Our ideas and thoughts revolved around creating a new timekeeping system for the company. After we did some research to empathize with the issues the users were having, we found that creating another timekeeping solution, that already exists in so many forms, would probably not solve the issue at hand. The employees at the company did not only ask for help with the timekeeping but seemed to have difficulties with a lot of the other systems that were being used as well.

Our project team came up with the idea of creating a custom internal documentation system, that would be a place for the management and employees to organize and look up information they need in their work. This would make the information more accessible to the employees and could save valuable time.

The internal documentation system will in this thesis be referred to as a "guide", or by the name "Menntor". The name Menntor came from a combination of the company name "Mennt" with the idea of a mentor as someone who provides guidance and knowledge.

1.1.2.1 Target user group

Our user group for this project was anyone who works for Mennt AS. This included the current management and employees, as well as anyone that could potentially work for the company in the future. Future employees of Mennt were included because one of the project's focus areas was to develop a sustainable solution.

1.2 Problem statement

The main design and development problem statement in this thesis is:

"How can the issues that Mennt is having due to a growing number of internal software systems be solved through the development of a web solution?"

This question changed during the project. It started as "*How can the project team develop a user-friendly web solution for Mennt AS that makes it easier for the employees to record their working hours and provides less frustration in the workday for everyone in the company?*". This initial question was based on the understanding of the problem at the beginning of the project. After some more research was done, the project team realized that we needed to shift our focus from timekeeping to a bigger picture, so the problem statement had to be changed.

Some more specific research questions were also chosen. Keeping the end-users in mind, I posed research question 1:

"How can the web solution be designed in a way that makes it user-friendly and ensures efficiency for the user?"

Additionally, I wanted to focus on meeting the goals on sustainability that the United Nations has put forward, so I posed research question 2:

"What measures can be taken to make the web solution sustainable?"

In this thesis, I will explain how I refined and expanded upon these research questions. At the end of the project, a minimum viable product that is usable for Mennt AS has been created, based on the research that was done.

1.3 Project organization

When choosing a development framework, the needs of the project and the research questions were considered. There were some specific goals I wanted to achieve, but from experience, I knew that the things I wanted to do would likely change during the time of the project. The research conducted throughout a project can quickly change the course of the project, which meant that there was a need to stay flexible.

For this reason, I chose to follow the Design Thinking framework to help come up with a good solution. This framework is very flexible, and I felt that it would be a good fit because it would help the team focus on the usability of the final design. An explanation of this framework can be read in the Design Thinking subchapter in Chapter 2.

Some milestones were also planned out for the project because it was useful to have concrete to move towards that would keep the project on track, considering the limited timeframe.

The milestones were:

- Finish project setup
- Gather requirements
- Finalize concept
- Finish low-fidelity prototyping
- Finish high-fidelity prototyping
- Implement system
- Finish testing system
- Finish writing report
- and Deliver project

The milestones were based on the activities I thought we needed to get done during the time of the project, but the tasks in between each of the milestones were adjusted as the project developed. The initial tasks were planned out and put into a Gantt chart, which will be explained in the next section.

1.3.1 TEAM ORGANIZATION AND COLLABORATION

The project described in this thesis has been carried out primarily by me, Isak Gjeitsund, as part of my final semester at the Norwegian University of Science and Technology. In some of the tasks, I have been assisted by two UX designers from Mennt AS, for example in conducting interviews, analyzing results, and brainstorming ideas for concept solutions.

Throughout this thesis, “the project team”, “the team”, or “we” will refer to the team as a whole. When we only refer to the two Mennt employees that worked on the project, it will be specified in the text. “Mennt” by itself will refer to the company, with all its managers and employees.

The project started in 2022, while the COVID-19 pandemic was still highly relevant, so the project team mostly organized our meetings and work digitally. The project was then postponed until the spring of 2023, but we kept meeting digitally since we were based in different locations.

A shared server on the social platform Discord was created for the team. Here, we could chat, share links and files, and meet up via video or voice calls where we had the option to share our screens. This was a great tool that helped us collaborate when we did not have the option to meet up in person. To keep our files and notes organized, we also used a shared Google Drive folder with a folder system and Google Keep for easy notetaking. A shared Google Calendar was also created to keep track of our meeting times, and the agenda for the meetings.

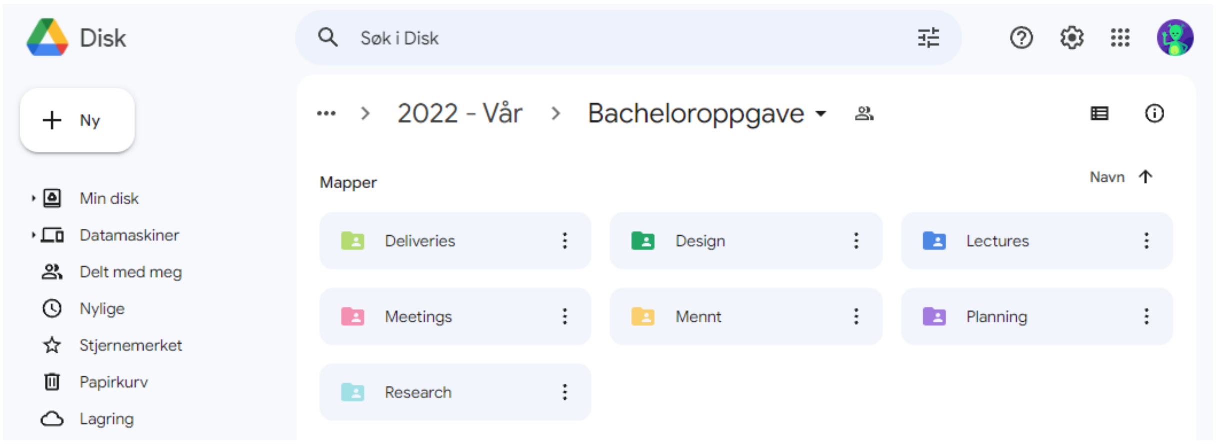
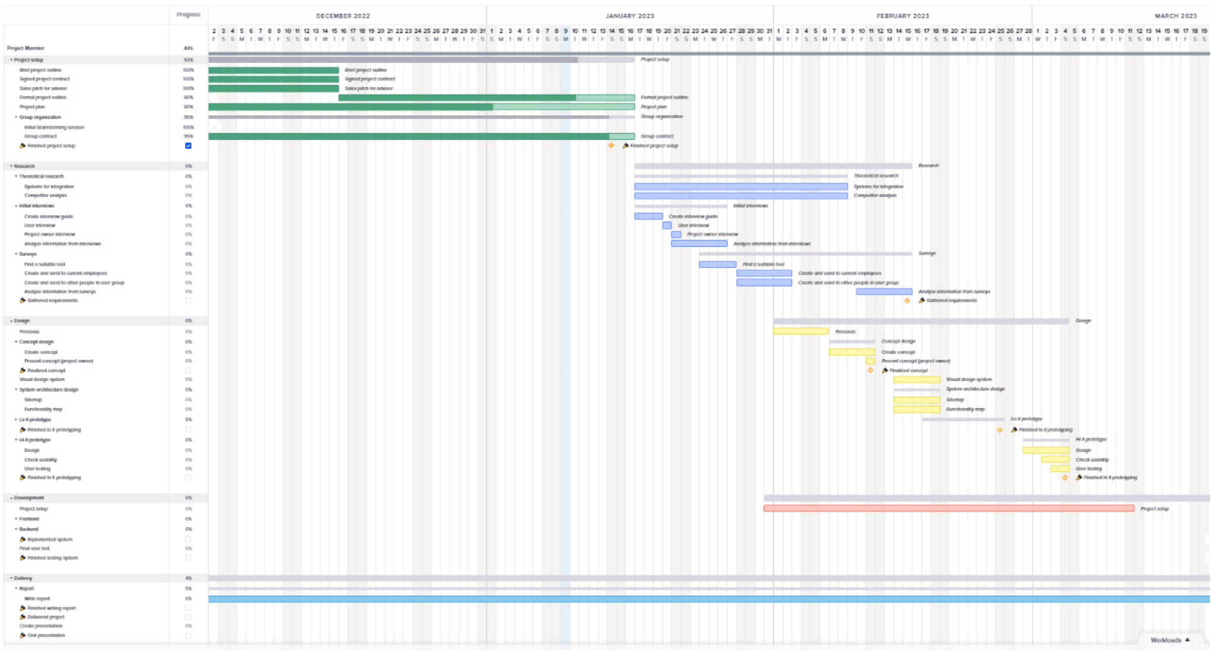


Figure 1 – Shared Google Drive folder

To keep track of the team's progress, I created a Gantt chart using the web service TeamGantt (see Figure 2). A Gantt chart is a type of chart used in project management that illustrates the tasks required in a project. The chart shows the start and end dates of each task using bars placed on a timeline and can include dependencies between tasks. This was used to plan and track progress throughout the project.



	Progress
Project Monitor	43%
Project setup	92%
Brief project outline	100%
Signed project contract	100%
Sales pitch for advisor	100%
Formal project outline	80%
Project plan	80%
Group organization	95%
Initial brainstorming session	100%
Group contract	95%
Finished project setup	✓
Research	0%
Theoretical research	0%
Systems for integration	0%
Competitor analysis	0%
Initial interviews	0%
Create interview guide	0%
User interview	0%
Project owner interview	0%
Analyze information from interviews	0%
Surveys	0%
Find a suitable tool	0%
Create and send to current employees	0%
Create and send to other people in user group	0%
Analyse information from surveys	0%
Gathered requirements	☐
Design	0%
Personas	0%
Concept design	0%
Create concept	0%
Present concept (project owner)	0%
Finalized concept	☐
Visual design system	0%
System architecture design	0%
Sitemap	0%
Functionality map	0%
Lo-fi prototype	0%
Finished lo-fi prototyping	☐
Hi-fi prototype	0%
Design	0%
Check usability	0%
User testing	0%
Finished hi-fi prototyping	☐
Development	0%
Project setup	0%
Frontend	0%
Backend	0%
Implemented system	☐
Final user test	0%
Finished testing system	☐
Delivery	4%
Report	5%
Write report	5%
Finished writing report	☐
Delivered project	☐
Create presentation	0%
Oral presentation	☐

Figure 2 – Gantt chart

1.4 Summary and thesis structure

In the Introduction chapter, I have explained the background for the project and the thesis by giving some information about Mennt AS, the challenges they are facing with organizing information and keeping everyone up-to-date, and a brief explanation of the solution the project group came up with. The research questions that will be answered in this thesis were also presented, and I provided some basic information about how the project was structured and how the project team was organized.

Chapter 2, "Theory", will explain some of the theories that are the foundation for the project and research in this thesis. It will be split into two parts, the first focusing on the user-centered design processes and principles that were used in the research and implementation, and the second part on sustainability related to the web.

I will go into detail about all the methods we have used in the project in Chapter 3, "Methods". This is where I explain how the research was conducted, how the results were analyzed, and how the web solution was designed. In this chapter you will also find the preliminary results that we got from the use of these methods.

In Chapter 4, “Development”, I will talk about how the development of the web solution took place. The chapter will talk about the process of deciding on which features to include, technologies to use, and the implementation process.

The results from the research relating to our research questions, as well as a presentation of the final design and the implemented web solution, can be found in Chapter 5, “Results”.

The 6th chapter, “Discussion”, will go into detail about the research questions and the related sub-questions that are presented in Chapters 2.1.6 and 2.2.3. This is also where you can find the conclusion to both the research questions and the main problem statement.

Finally, chapters 7, 8, and 9 will contain a list of sources, an overview of the figures and tables, and the appendix.

2 THEORY

In this chapter, I will explain the main theories that I have based the research, and the design of the project, on. The theory is relevant to how I chose the methods I used in the project and the decisions that were made in the design process, and it will also be used in the Discussion chapter later in the thesis.

2.1 User-centered design

This chapter provides an overview of various design concepts and processes used in user-centered design, that I have used in this project. The Interaction Design Foundation defines user-centered design (UCD) as "an iterative design process in which designers focus on the users and their needs in each phase of the design process" (Interaction Design Foundation, no date-a). The design process puts the user in focus, not the product, and involves thinking about the users' needs and testing on users every step of the way when designing and developing your product.

2.1.1 DESIGN THINKING

Design thinking is a user-centered design framework wherein designers see the problem at hand as a suggestion, and try to analyze it to find the underlying issues (Norman, 2013, pp. 218-219). They do not jump to conclusions and immediately start searching for a solution, but instead search for the real problem. Doing it this way ensures that the final solution will more accurately fix the issue that the users are experiencing.

Design thinking processes are iterative, and usually consist of several phases. Norman Nielsen Group have stated that there are 6 of these phases – *Empathize*, *Define*, *Ideate*, *Prototype*, *Test*, and *Implement* (Gibbons, 2016). See Figure 3 below for an illustration of these stages and how one can move between them. Implementation of the final solution is often not included as a step in the design thinking process, since design thinking focuses on coming up with well-designed solutions, not the development of the final product.

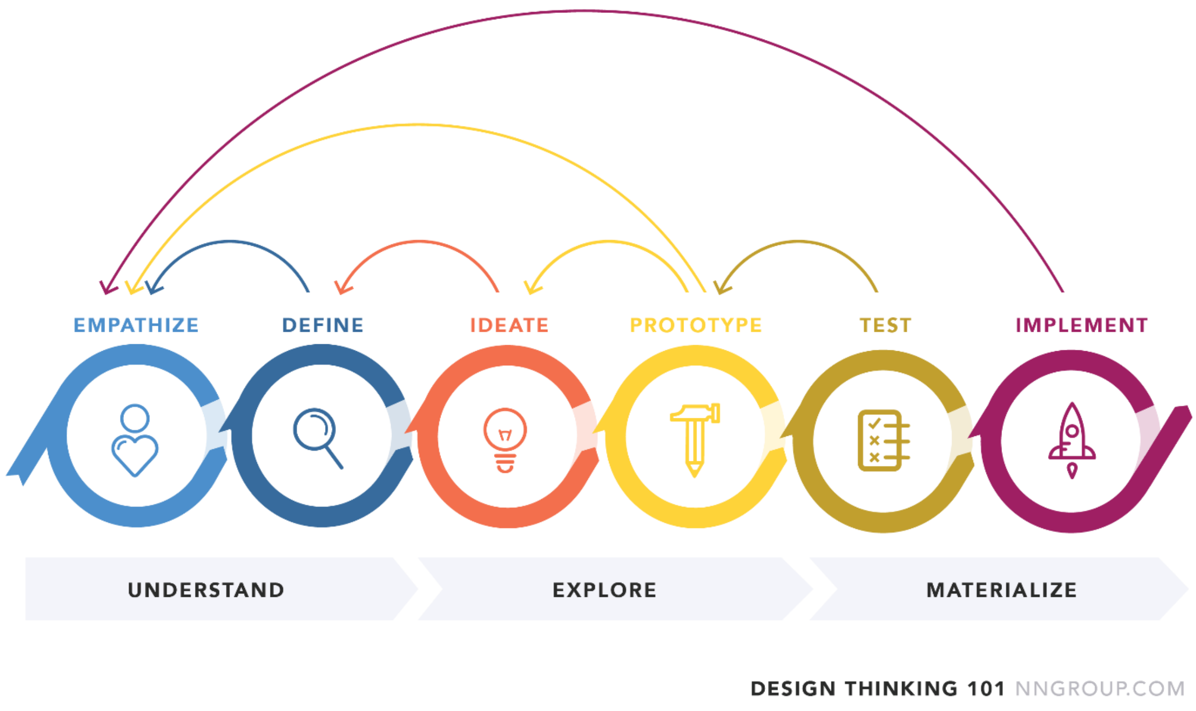


Figure 3 – Design thinking framework stages (Gibbons, 2016)

I have used all of these six stages in this project. The framework was chosen mainly because it focuses on finding the correct solution that fits the user's needs, not what the company assumes them to be. Another reason I chose this design process is because the framework is very flexible. There was room to jump between the distinct phases, which was something that was needed in this project.

In the first two phases, *Empathize* and *Define*, you perform research to empathize and understand your users, and then analyze this information to figure out what the problem areas are. This way of doing it ensures that you find the correct problem to solve, instead of just coming up with a bunch of solutions right off the bat.

In the *Ideate* phase, you then come up with ideas for how to solve the problem that you have found in the previous phases. Then you put these ideas into action in the *Prototype* phase, by creating a representation of your idea to see if it will work in practice.

In the *Test* phase of the framework, you test your ideas on real users by letting them try out your prototype. This way, you get feedback from the people that will use your final product about how they feel about the solution you have come up with. Usually, after testing, you will then go back to the *Ideate* and *Prototype* phases and create new ideas for how to solve the problems you found in the *Test* phase.

Depending on what problems you found in your tests, it could also be useful to step back to the *Empathize* and *Define* phases at this point if it seems that your current idea does not solve the problems at hand, and you require a deeper understanding. Once all iterations are done, you are left with the design of your solution. The last phase, the *Implement* phase, is where you put your vision into effect and develop the product that you have prototyped and tested.

2.1.2 TEN USABILITY HEURISTICS

The "10 Usability Heuristics for User Interface Design" are a set of suggested rules created by the famous web usability expert Jakob Nielsen in 1994 (Nielsen, 1994). These ten rules have been widely adopted in the field of user experience design and were developed to guide designers in creating more intuitive, efficient, and usable web interfaces.

The ten rules include keeping users informed about what is going on, creating an experience that matches the real word and is understandable to the users, giving users a clearly marked emergency exit, adhering to consistency and web standards, and preventing errors. They also try to minimize the user's memory load, adding shortcuts for more experienced users, keeping the design minimalistic, explaining error messages in clear language, and providing documentation or help when the users might need it.

2.1.3 ACCESSIBILITY AND UNIVERSAL DESIGN

Web accessibility refers to the practice of designing and developing websites, web applications, and digital content in a way that makes them usable and accessible to people with disabilities. In the "Convention on the Rights of Persons with Disabilities (CRPD)" universal design is defined by the United Nations as "the design of products, environments, programmes and services to be usable by all people, to the greatest extent possible, without the need for adaptation or specialized design" (United Nations, no date-c).

Accessible design and universal design are related concepts that are often used interchangeably, but they have distinct differences. While accessible design is a process in which we design products that can be used by people with disabilities, universal design considers the needs of all its potential users, including, but not limited to, people with disabilities (Kalbag, 2017, pp. 5-7).

2.1.4 INFORMATION ARCHITECTURE

In the late 1990s, User-centered design was split into two distinct “professions” that were dedicated to enhancing the usability of websites – Usability and Information Architecture (Krug, 2014, p. 183). Information architecture refers to the design and organization of information and content on the website. A well-designed information architecture will structure the data in a meaningful way, which will make it easier for the users to locate the information they need and understand how to use the website effectively.

There are two main ways to visualize information architectures (Rosenfeld, 2015, pp. 80-88). Top-down information architecture refers to a design approach that starts by looking at the broad perspective and gradually breaks it down into smaller, more manageable pieces of information. On the other hand, bottom-up information architecture refers to an approach where the focus is on the content itself, and the designer starts by looking at the content in detail to determine how it can be grouped and organized. This approach allows for a more user-centered design.

In general, top-down information architecture is more rigid and focused on maintaining consistency within an established structure, while bottom-up information architecture is more flexible and responsive to the actual content and users' needs.

In essence, the design of the information architecture is incredibly important to the usability of a website. Effective information architecture helps create navigational structures that facilitate efficient browsing and searching, a clear and intuitive hierarchy of content, and consistency in labeling and terminology throughout the web solution.

2.1.5 COMPONENT-DRIVEN DESIGN

Component-driven design and development is a modern approach to software design that focuses on building applications as a collection of reusable, modular components (Chromatic, no date). This approach allows the creators to build complex user interfaces and applications more efficiently, by breaking them down into smaller, more manageable parts.

In component-driven development, each component is a self-contained unit that encapsulates all of its logic, styles, and markup. This allows developers to focus on building individual components without worrying about how they will fit together in the larger application.

One of the main advantages of component-driven design is that it promotes reusability. Because components are modular and self-contained, they can be reused across multiple pages or even across projects. This can greatly reduce the amount of time and effort required to build complex applications. Component-driven design and development also ensures consistency across the user interface and makes it easier to change and expand the project in the future.

In our *Prototype* and *Implement* phases, we followed component-driven design practices when creating a high-fidelity prototype and when implementing the final design.

2.1.6 SUMMARY AND REFINEMENT OF THE FIRST RESEARCH QUESTION

The first research question in this thesis asks how to design the web solution in a way that makes it user-friendly and ensures efficiency for the user. Through my research on user-centered design, universal design, and information architecture I refined this question into more specific questions that I wanted to answer in my project.

- What are the specific user needs and preferences that the web solution should address?
- How can I ensure that the web solution is accessible and inclusive for users with different abilities and backgrounds?
- How can I streamline the user experience and minimize cognitive load through intuitive navigation, clear labeling, and effective use of visual design?
- How can I test and evaluate the user-friendliness and efficiency of the web solution, and make iterative improvements based on user feedback?

By using the Design Thinking framework and relying on the theory that I have explained in this subchapter, I will try to answer these questions to the best of my abilities.

2.2 Sustainability

Sustainability was defined by the United Nations Brundtland Commission as "meeting the needs of the present without compromising the ability of future generations to meet their own needs" (United Nations, no date-b).

2.2.1 UN'S SUSTAINABILITY GOALS

In 2015, the United Nations presented an agenda that was created to help solve the most important issues we have on the planet – "The 2030 Agenda for Sustainable Development", commonly known as the Sustainable Development Goals or the SDGs (United Nations, no date-a). There are 17 goals with a total of 169 targets that cover a broad range of issues and are all meant to address the universal need for development that works for all people and the planet. The SDGs are intended to be achievable by 2030 and provide a framework for governments, businesses, organizations, and individuals to work together to achieve a more sustainable future for all.

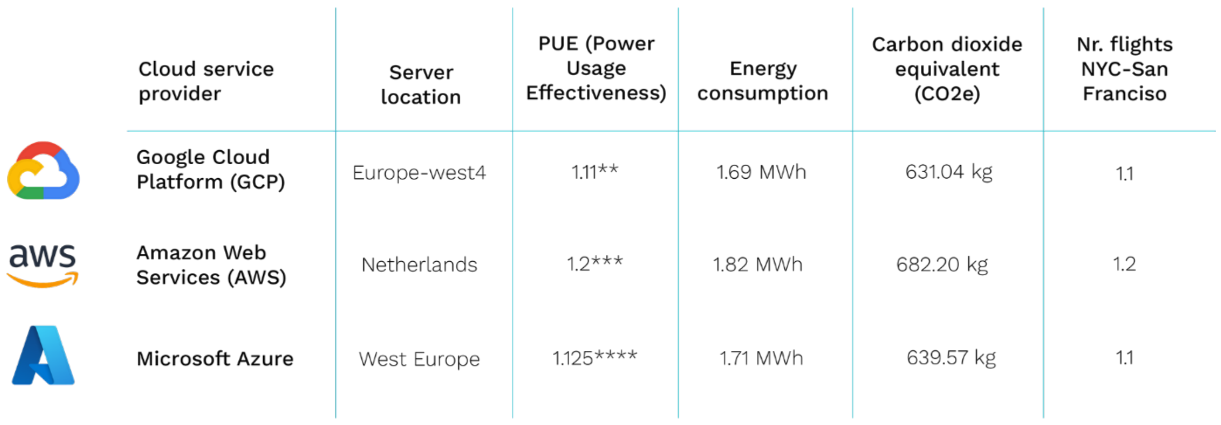
2.2.2 ICT RESOURCE USAGE

Web development is highly relevant concerning several of the goals presented by the UN. Information technology is a huge part of the modern world, and we need to make sure that as many people as possible have access to these services. However, the consumption of these services also uses an enormous amount of resources. It has been estimated that Information and communications technology could be responsible for as much as 2.1-3.9% of global greenhouse gas emissions, which is way more than what was initially assumed (Freitag *et al.*, 2021). When we design and develop new websites and web services, the data has to be stored somewhere so that our users can access it over the internet. In a project like this, that is usually done through cloud hosting. Massive data centers across the world hold large amounts of servers that store the data that we send to "the cloud".

Building these data centers and manufacturing all the equipment requires us to use a lot of natural resources, but the data centers also consume lots of energy. The servers generate a lot of heat, which in turn requires even more energy to cool down. The more data we store, the more servers are required, which in turn requires a higher use of resources. According to Statista, the total amount of data predicted to be created and consumed globally in 2023 is 120 zettabytes (Statista, 2021). And these data centers not only store the data we create but also spend energy on sending the data out to clients every time they log on to use our websites.

2.2.2.1 Cloud hosting

The biggest cloud hosting providers, taking up two-thirds of the market share, are Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure (Richter, 2023). In 2019, the online magazine WIRED did a comparison of the three providers and ranked them based on who has the greenest cloud (Oberhaus, 2019). GCP got the highest rating with a B+ for overall greenness, Azure followed right behind with a B, while AWS got a C- from the magazine. Recently, the AI consulting company Xomnia posted a blog post where they calculated the three cloud providers' carbon footprint using the "Green Algorithms" method (Kuijt, 2023).



Cloud service provider	Server location	PUE (Power Usage Effectiveness)	Energy consumption	Carbon dioxide equivalent (CO2e)	Nr. flights NYC-San Franciso
Google Cloud Platform (GCP)	Europe-west4	1.11**	1.69 MWh	631.04 kg	1.1
Amazon Web Services (AWS)	Netherlands	1.2***	1.82 MWh	682.20 kg	1.2
Microsoft Azure	West Europe	1.125****	1.71 MWh	639.57 kg	1.1

Figure 4 – Measurements created using the Green Algorithms calculator (Kuijt, 2023)

In this test, we can see that Google Cloud Platform had the smallest carbon footprint out of the three, with Microsoft Azure as a close second, and AWS last with a much larger footprint. This correlates well with how they were ranked by WIRED. However, the method that was used only calculates the carbon footprint of computations in a very specific use case, and the server locations were decided based on what was closest to where the testers were located.

The carbon footprint of data centers owned by any of the cloud providers varies greatly. Below, in Figure 5, you can see the carbon intensity of data centers owned by Google. This figure shows data I fetched from Climatiq visualized by using a digital heatmap. There is a huge gap in intensity between all of Google's data centers, with "Switzerland (europe-west6)" at the bottom of the list with 0.1008 grams of CO2e/h, and "Indonesia (asiasoutheast2)" at the top with 2.68347 grams of CO2e/h. This shows that even if you choose to use the greenest of the three top providers, you can still end up using a data center that is less than green, depending on the location of the servers being used.

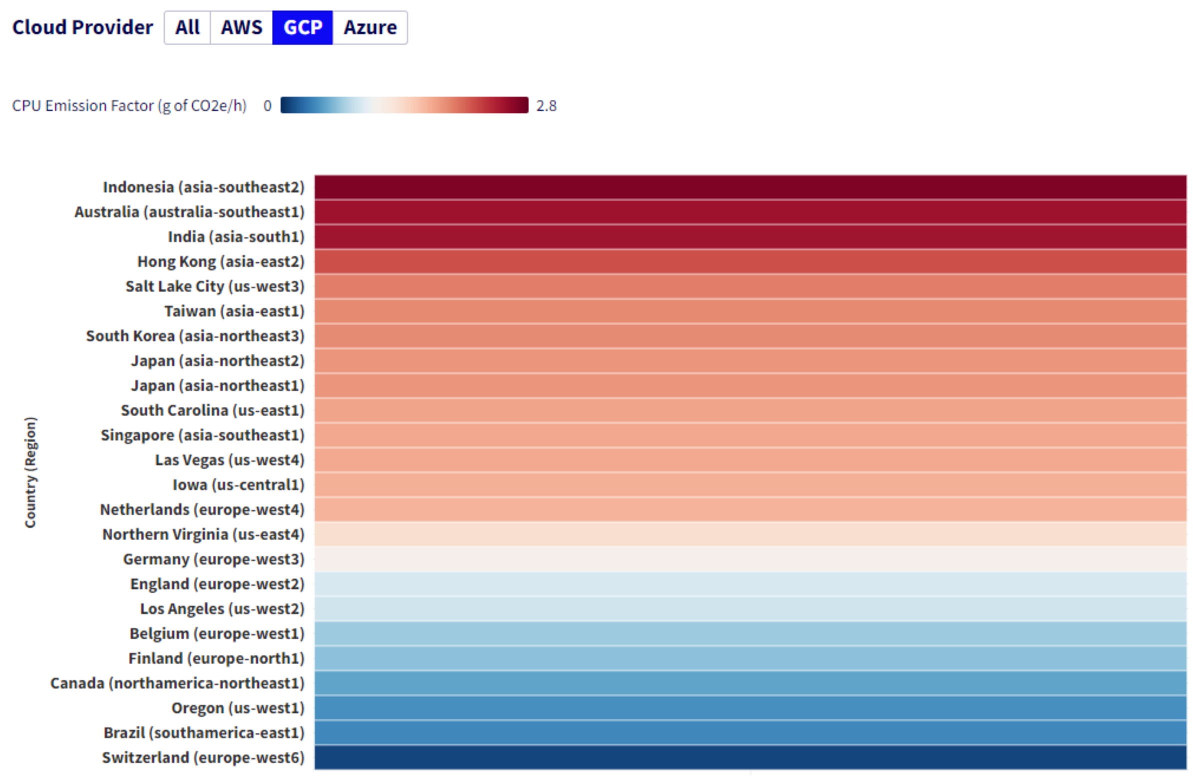


Figure 5 – Carbon Intensity of GCP data centers (Lavi, 2022)

Using the same tool, I could also see that Microsoft's least carbon-intensive data center is their "London (UK South)" with 0.57841 g of CO2e/h, and AWS has the top result, with their data center "Sweden (eu-north-1)" with only 0.01893 g of CO2e/h. The data center in Sweden also happens to be the one that is the closest to Norway.

2.2.3 SUMMARY AND REFINEMENT OF THE SECOND RESEARCH QUESTION

The second research question that was posed asked what measures we can take to make the web solution sustainable. After doing more research into sustainability and web development, I was left with some more specific questions that I wanted to answer in relation to my project.

- What are the key environmental impacts and challenges associated with web development?
- How can I ensure that the web solution uses energy-efficient technologies and infrastructure, and avoids unnecessary resource consumption?
- How can I measure and report on the sustainability of the web solution?

2.3 Summary of chapter

In this chapter, I have talked about the theory that lays the foundation for the work that was done in the project, and this thesis.

The most important user-centered design processes that were followed in this project were explained, including the Design Thinking framework, Jakob Nielsen’s ten usability heuristics, universal design, information architecture design, and component-driven design.

Then, I talked about why sustainability has been important to me in this project, what the United Nation’s sustainability goals are, and how the internet uses a shocking amount of natural resources, which is important to consider when developing new or existing web solutions.

3 METHODS

The use of effective methods is crucial to the success of any project. In this chapter, I will explain which methods were used, why I chose to use these specific methods, and the process me and the team went through when using them. To provide clarity, I have categorized the methods that were used into practical research methods, information architecture, design methods, and testing methods. The methods and development of the web solution described in this thesis will be laid out in a separate chapter – Chapter "4 Development".

Preliminary results that were used to progress the design of the web solution will be explained in each of the subchapters. The insights and data gathered through these methods that are related to the research questions, as well as a more extensive presentation of the final design and implementation of the web solution will be further detailed in the "Results" chapter of the thesis.

3.1 Practical research methods

Practical research methods are normally used to gather information about the project and its users (Andersen, 2020). In this project, I used two types of interviews, as well as questionnaires to gather the data. To analyze the information that was obtained from the interviews, the project team used affinity diagramming. Finally, I did an analysis of a few competitors for the solution that I had in mind.

3.1.1 INTERVIEWS

Interviews are a flexible research tool used to gather information from other people (Tomitsch *et al.*, 2018, p. 78). The method is very useful when you already have a set problem statement and you have read up on the subject, but you want to shine more light on the problem at hand (Andersen, 2020, p. 133). I chose to use interviews because I wanted more in-depth information about the project I was starting. I also needed to get more qualitative data that I could use as a base for the questions in the upcoming questionnaires.

3.1.1.1 Unstructured interview

At the very beginning of the project, I held an unstructured interview with the manager of the company. An unstructured interview is a more free-flowing interview that asks open-ended questions, and new questions might emerge depending on the information you get from your informant (Tomitsch et al., 2018, p. 78). I needed this interview to get information about what the company wanted to get out of the project and to make sure that I understood the assignment correctly. The interview was conducted more like a casual meeting, but the manager was asked to participate and was informed about the purpose of the interview.

3.1.1.2 Semi-structured interviews

To gather information about the project, I was planning to use questionnaires. This method was chosen because I wanted to measure what all the employees of Mennt felt about the current timekeeping solution, which was the focus at the time, and the initial ideas that the project group had for the project. Considering that the company only has a handful of employees, I also wanted to send out questionnaires to a control group consisting of non-employees.

However, I felt like some essential information was missing that was needed to make the questionnaires useful. Some things were still unclear, like how the employees at Mennt were using their current systems and their feelings about it. I needed more background information, and I felt like a questionnaire with vague questions was not going to be particularly useful.

Therefore, I chose to conduct a couple more semi-structured interviews as one of the research methods, because I needed more in-depth information about the project. I wanted to understand more about the current system before I could create and send out questionnaires, and I wanted to uncover more of the issues Mennt was having with their current system, as well as what they would want from the new system.

Planning the interviews

While planning how the interviews were going to be conducted, I started by looking at what kind of information I wanted to get from the interviews. I set three goals for the interviews:

1. Find out more about how the employees are using the current system.
2. Find out more about the needs of the new system/solution.
3. Gather information for the upcoming questionnaires.

Choosing informants

When choosing the informants, I decided to limit the number of informants that would be interviewed, mostly due to concerns about available time – both in planning and conducting the interviews. I kept in mind that we would be asking some open-ended questions to get more qualitative information, which would result in more data, as well as more time spent analyzing the data. In addition to this, the data would be used to create questionnaires, which would let me get additional information from the other employees later.

After asking if they were willing to participate, the project group ended up interviewing two informants from the company. One of the informants was the manager of the company. I chose to do a second interview with him because he was the only one with knowledge of all aspects of the software that the company is using. He also collaborates closely with the developers during development. The other informant was a project manager at the time, who additionally worked closely with the design team. I tried to choose informants who would represent the end users well, so with the time constraints these two seemed like the best choice.

Interview preparation

To prepare for the interviews, the project group had a brainstorming session where we constructed questions to ask the CEO, an employee, or either of them. Since these interviews were going to be semi-structured, we also prepared interview guides (see Appendix B) to ensure that all the important questions were included and that the interviews were conducted in a consistent manner.

The interview guides began with an introduction that explained the background of the project and why the informant was being interviewed. After the introduction, the guides included some easy questions to help warm up the informants and make them comfortable, followed by questions about the current solution and finally about their thoughts on what the new system should look like. The questions in these sections also included some follow-up questions that we thought might be useful, depending on the answers given.

In some cases, it can be useful to observe the informants while they are in a situation related to your project (Andersen, 2020, p. 146). This is why, at the end of the interview, I included an observational section, where the interviewers asked the informants to show how they perform specific tasks in their current systems. I chose to do this so that I could see what it was like for them to navigate specific tasks in the current solution, but also to ask questions and hear more about how they were feeling while they were interacting with the software.

Conducting the interviews

The interviews were conducted online, via Microsoft Teams. To ensure the quality and accuracy of the data collected during the interviews, two members of the project team were present for each interview. One team member read from the interview guide and talked to the informant while the other wrote down detailed notes. After we conducted the interviews, the team members had a debrief to talk about the interviews, while the interviews were still fresh in mind. This helped ensure that we had written everything down, that the notes were accurate, and to clarify anything that seemed ambiguous to us. To analyze the results that we got from the interviews, we used a method called affinity diagramming.

3.1.1.3 Affinity diagramming

To analyze the results from the interviews, the project team used Affinity diagramming. Affinity diagramming is a method that helps organize large collections of qualitative data to find recurring patterns (Tomitsch *et al.*, 2018, pp. 22-23). We followed the four-step process described in "Design. Think. Make. Break. Repeat. The Handbook of Methods" (Tomitsch *et al.*, 2018, pp. 22-23). See Appendix C for large-scale images of the process.

First, we collectively went through all the data we had, to identify specific problems or observations. Any feelings or issues that the interviewees had about their current system were noted down on separate yellow sticky notes, called "affinity notes".

To be able to do this digitally, the project team used Miro. Miro is a free, online whiteboard that lets team members work together in real-time. You can place text, sticky notes, and other elements on the board, and they have a range of features that help users work efficiently, like easy-to-use mind-maps and wireframe creators, or pre-made templates to help project groups get started.

In the second step of the affinity diagramming, the sticky notes on the wall were grouped according to common themes and labeled by using a blue affinity note for each cluster, with a need expressed in the voice of a user. We repeated this process until the final step, which was to "walk the wall" and generate ideas. Here, our team came up with concrete ideas for potential solutions, which were recorded on a different colored sticky note and attached to the related affinity note.

This left us with a visual representation of the data we had collected, organized by common themes and needs expressed in the voice of users. The affinity diagram helped us arrange the qualitative data into meaningful insights, which we could then use to inform our design process. By using this method, we were able to generate ideas for potential solutions that addressed the users' specific needs and concerns.

3.1.1.4 Results from interviews

We started with a total of 215 affinity notes on the board. After the third stage of the affinity diagramming, we had grouped the users' feelings into 12 areas. Each of these areas had between two and six noted in them which correlated to a user's feelings, wants, or needs (see Figure 6 below).

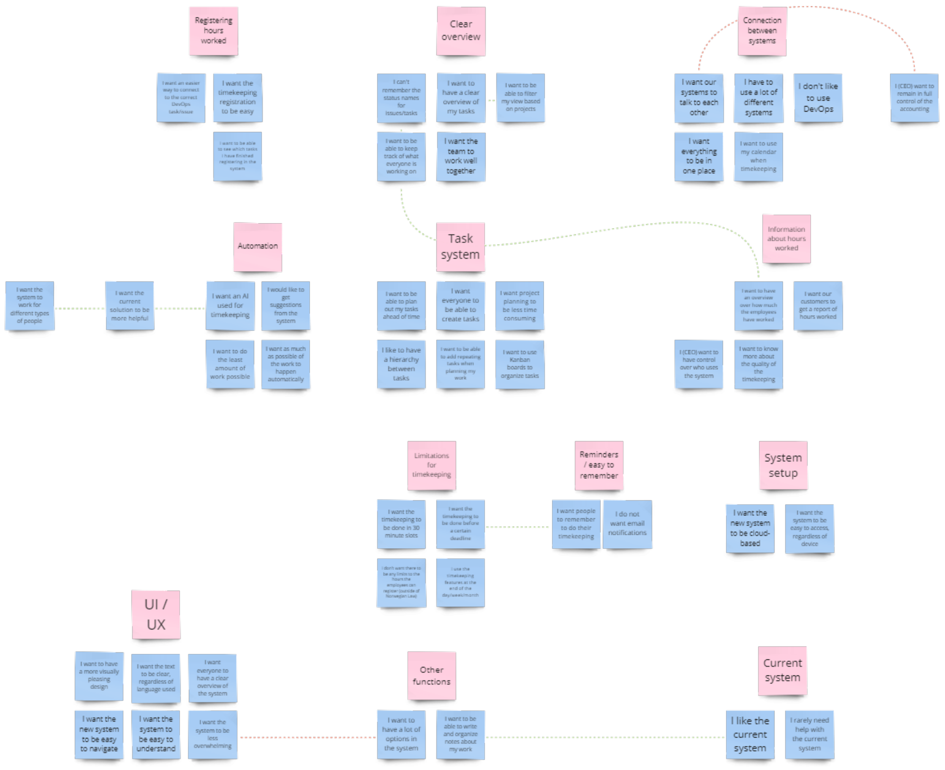

Registering hours worked
I want an easier way to connect to the correct DevOps task/issue
I want the timekeeping registration to be easy
I want to be able to see which tasks I have finished registering in the system
Clear overview
I can't remember the status names for issues/tasks
I want to have a clear overview of my tasks
I want to be able to filter my view based on projects
I want to be able to keep track of what everyone is working on
I want the team to work well together
Connection between systems
I want our systems to talk to each other
I have to use a lot of different systems
I don't like to use DevOps
I (CEO) want to remain in full control of the accounting
I want everything to be in one place
I want to use my calendar when timekeeping
Automation
I want the system to work for different types of people
I want the current solution to be more helpful
I want an AI used for timekeeping
I would like to get suggestions from the system
I want to do the least amount of work possible
I want as much as possible of the work to happen automatically
Task system
I want to be able to plan out my tasks ahead of time
I want everyone to be able to create tasks
I want project planning to be less time consuming
I like to have a hierarchy between tasks
I want to be able to add repeating tasks when planning my work
I want to use Kanban boards to organize tasks
Information about hours worked
I want to have an overview over how much the employees have worked
I want our customers to get a report of hours worked
I (CEO) want to have control over who uses the system
I want to know more about the quality of the timekeeping
Limitations for timekeeping
I want the timekeeping to be done in 30 minute slots
I want the timekeeping to be done before a certain deadline
I don't want there to be any limits to the hours the employees can register (outside of Norwegian Law)
I use the timekeeping features at the end of the day/week/month
Reminders / easy to remember
I want people to remember to do their timekeeping
I do not want email notifications
System setup
I want the new system to be cloud-based
I want the system to be easy to access, regardless of device
UI / UX
I want to have a more visually pleasing design
I want the text to be clear, regardless of language used
I want everyone to have a clear overview of the system
I want the new system to be easy to navigate
I want the system to be easy to understand
I want the system to be less overwhelming
Other functions
I want to have a lot of options in the system
I want to be able to write and organize notes about my work
Current system
I like the current system
I rarely need help with the current system

Figure 6 – Results from the third stage of Affinity Diagramming

What I saw from the final result was that from these two interviews, we had found a huge number of issues with Mennt's timekeeping system. Yet, they also stated that they "liked the current system" and that they "rarely needed help with the current system". Looking at all the features and options they wanted from a new system, I could see that there were too many things to get done, and fixing everything would not be possible within the timeframe of the project. I did also notice that a lot of the feedback from the informants regarding their current system was not about the timekeeping but could be related to a lack of organization.

3.1.2 QUESTIONNAIRES

To gather a lot of data with a limited amount of resources, questionnaires are a good way to go (Tomitsch *et al.*, 2018, pp. 102-103). Questionnaires are a versatile and cost-effective way of collecting data from a large number of people. They provide standardized data that can be analyzed quickly and efficiently, making them a valuable tool for researchers in many different fields. I chose to use online questionnaires as a research method because the project group had only interviewed two informants from the user group, and I wanted more perspectives on the problem at hand and potential solutions.

3.1.2.1 Preparing the questionnaires

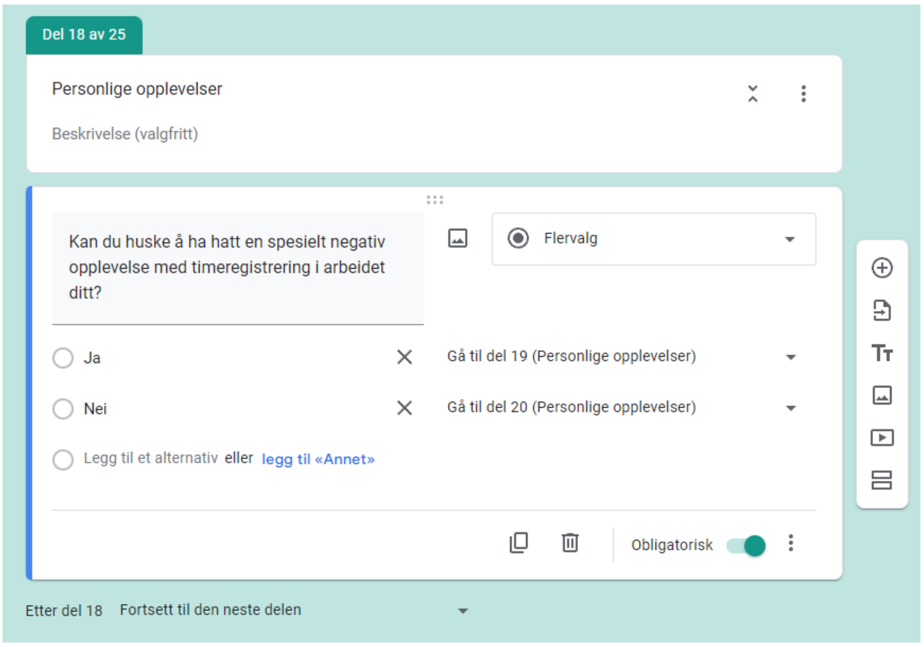
The project group got together to brainstorm questions for the questionnaire. The questions we produced were based on the data we got from the interviews, as well as a few focus areas that we had agreed on beforehand. Some of them were areas that were highlighted in the interviews as problem areas or areas of interest, while the others were areas the group thought were important to consider. The focus areas were:

- Timekeeping
- Reducing frustration
- Features to include or scrap
- Automation of timekeeping
- Providing the best usability
- Saving time for the users
- Making the system enjoyable

We put sticky notes with these focus areas on a Miro board and created related sticky notes framed as questions. Most of the questions we came up with were related to timekeeping. We asked about the respondents' experiences with timekeeping software, as well as their thoughts and feelings about what they would want to see in their timekeeping software. At the time, the project group was still focused on creating a new timekeeping solution for the company, which is why this was the focus of these questionnaires. I also included some questions that were meant to check whether the perspectives of the informants in the interviews matched the respondents in the questionnaires.

To make the questionnaire as user-friendly as possible, I split the questions that the project group and I had come up with into sections. Long questionnaires are less likely to be completed (Tomitsch *et al.*, 2018, p. 102), and to fix this problem, I chose to use the method "chunking" (Halarewich, 2016), by creating sections for all the questions. When choosing a digital tool, I considered that I wanted to be able to create a multi-step form to reduce the cognitive load for the respondents. For the same reason, I tried to make sure that the questions were not ambiguous, limited the number of answer options, and included "Other" and "Don't know" options wherever this was applicable.

Using questionnaires is a method that is usually limited when it comes to asking follow-up questions (Andersen, 2020, p. 139). With the use of a digital questionnaire service, I was able to ask follow-up questions depending on what the respondent answered in the previous question.


Del 18 av 25

Personlige opplevelser

Beskrivelse (valgfritt)

Kan du huske å ha hatt en spesielt negativ opplevelse med timeregistrering i arbeidet ditt?

Flervalg

Ja	Gå til del 19 (Personlige opplevelser)
Nei	Gå til del 20 (Personlige opplevelser)

Legg til et alternativ eller legg til «Annet»

Obligatorisk

Etter del 18 Fortsett til den neste delen

Figure 7 – Google Forms question setup with prompt to follow-up questions

I chose to write the questionnaires in Norwegian, because most of the respondents would be native Norwegian speakers, and I hoped this would make them more comfortable and willing to reply to the questions. I also made sure to set a deadline that was not too far ahead, to encourage respondents to complete the questionnaire as quickly as possible.

At the start of the questionnaire, I included an introduction for the respondents that explained the goal of the research, and how long it would take to complete. I decided to keep the respondents anonymous, so this was also included in the information given, as well as my contact information in case there were any questions or issues. The respondents were also told that there would be an open question at the end of the questionnaire, where they could write whatever they wanted, in case there was any additional information or comments they would like to provide.

I created two questionnaires. One questionnaire was directed at Mennt employees, which can be seen in Appendix D. I wanted to get an honest view of what everyone in the company thought about their current solution and the ideas for the new solution. The company only had six employees that were available to answer the questionnaire at the time, so I also sent out a second questionnaire to non-employees, as a control group. This was done to see how other respondents felt about the same questions and to ensure that the needs of potential future employees would be considered when deciding on a solution. The second questionnaire can be seen in Appendix E.

The reason behind separating the two questionnaires was that I wanted more information about the current system and more open-ended questions in the questionnaire directed at the Mennt employees. Respondents outside of the company would not be able to answer some of these questions, like "What features are you missing in the current system?", but I also wanted to limit the amount of total open-ended questions to cut down the time I would have to spend analyzing the data (Andersen, 2020, p. 140).

3.1.2.2 Choosing the right tool

I felt that the easiest way to reach out to more respondents would be to use digital questionnaires, but I needed to choose the right tool for the job. I did some research into the biggest competitors of online questionnaires, like Microsoft Forms, Google Forms, SurveyMonkey, and TypeForm. I ended up choosing Google Forms because it had all the alternatives for question types that I needed, it was the technology I was the most familiar with, and from my research, I could see that no features were missing that I needed in the questionnaires.

Once I had finished setting up the questionnaires, I did a pilot test on each of the questionnaires to ensure that everything was in order and then sent them out to all the employees at Mennt. The control group questionnaire was sent to a lot of people, and they were told that they could share them with whomever they wanted since it was not relevant whether they were in the user group. Any relevant information about the respondent would be collected through the questionnaire.

3.1.2.3 Results from questionnaires

A lot of the research I did in the questionnaires was focused on the timekeeping system, which did not end up being that useful to the project due to it moving in a different direction, so I will only include the results that were relevant to the project and the thesis. Due to some of the open-ended questions including personal information, the full extent of the results will not be included in the appendix.

The questionnaire aimed at the employees at Mennt was the most extensive one. There was a total of 6 respondents from Mennt that filled out the questionnaire. I wanted to know what their feelings were about the current system, so the questions “How satisfied are you with the current solution as a whole?” and “How much do you like the following systems?” were asked in the questionnaires.

Half of the respondents answered that they were “Neither satisfied nor dissatisfied” with the current solution, while the other half said they were “Satisfied” (see Figure 8). This showed that they were not dissatisfied with the current timekeeping solution.

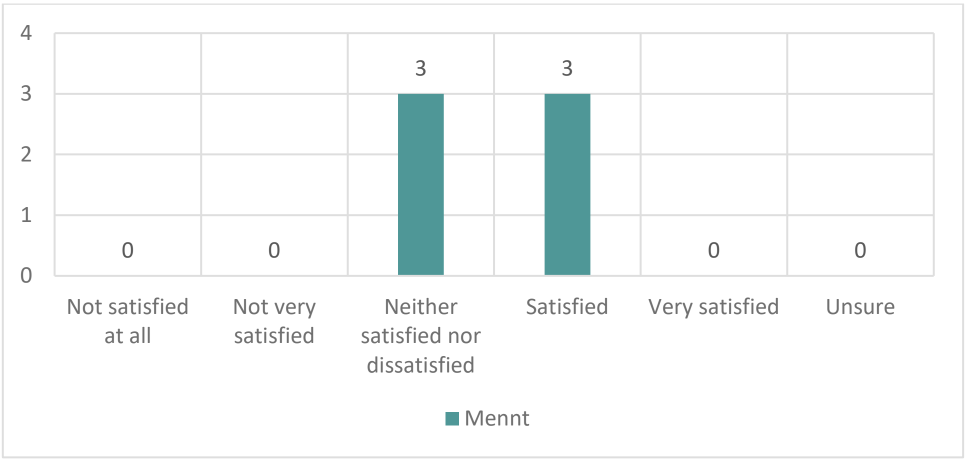


Figure 8 – "How satisfied are you with the current solution as a whole?"

When asked about specific systems they were using, most of the respondents leaned towards neutral opinions or liking the systems, with one system being the exception (see Figure 9). The organizational system “ClickUp”, which Mennt was trying out for timekeeping at the time, got 3 replies for “Not especially”, 2 for “Neutral”, and 1 for “Likes” from the respondents. Apart from ClickUp, it seems that the employees are mostly neutral or positive towards the software systems that they are currently using.

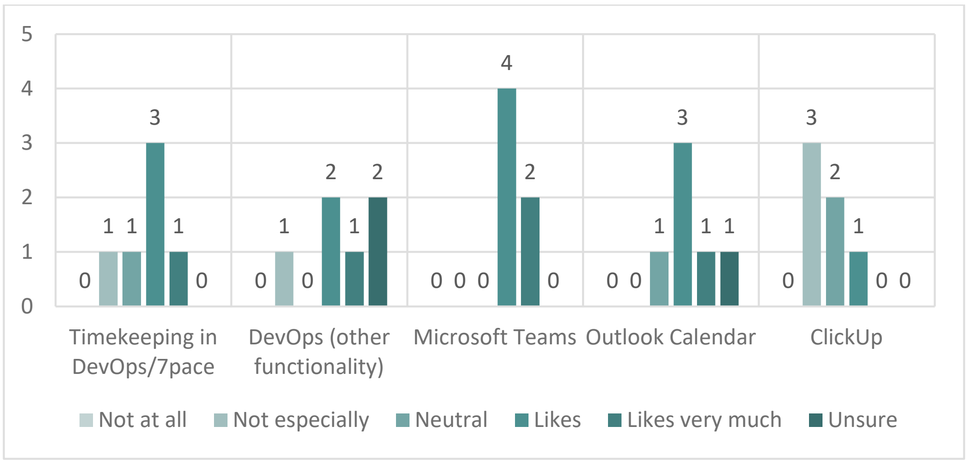


Figure 9 – "How much do you like the following systems?"

There was also a question in the questionnaire regarding the user-friendliness of their current system, where the respondents could rank the user-friendliness on a scale from 1 to 10. The result can be seen in Figure 10 below, and the average score was 5.6 out of 10. The midpoint on this scale would be 5.5, so the user-friendliness score was just above this point.

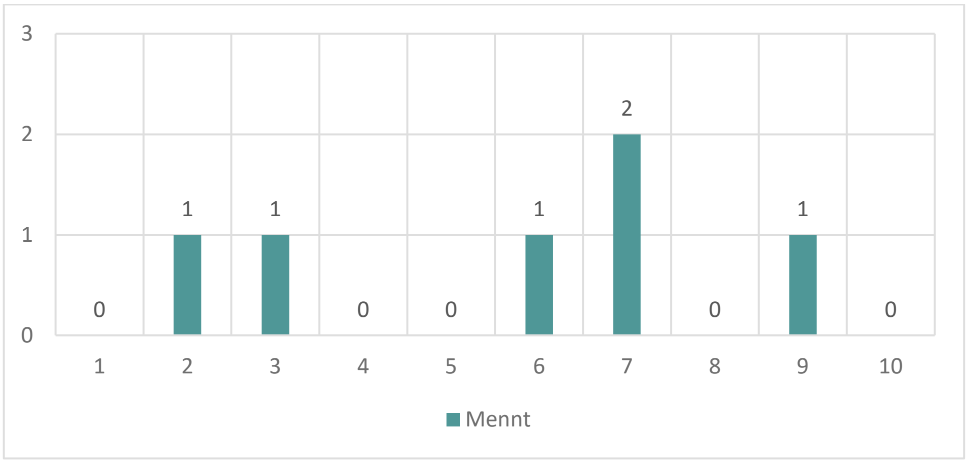


Figure 10 – "How user-friendly do you think the current system is on a scale from 1 to 10?"

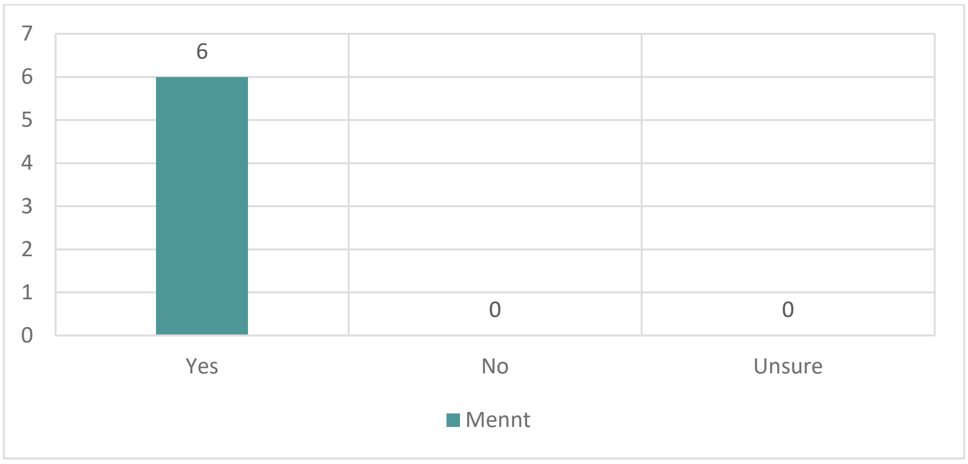


Figure 11 – "Would you have wanted to register your hours directly into your calendar?"

Some of the questions in the questionnaire were based on the project team's ideas for solutions to fix the issues we had found in the interviews. Two of the most positive responses were in relation to being able to register hours directly into their calendar (see Figure 11 above) and whether they thought the current solution would be more usable if there was an easily accessible guide they could use (see Figure 12 below). 100% of the employees would have liked to do their timekeeping through their calendar system, while 50% of the employees would like an easily accessible guide. The other 50% did not answer "No" but were unsure about whether this would make the current solution more usable.

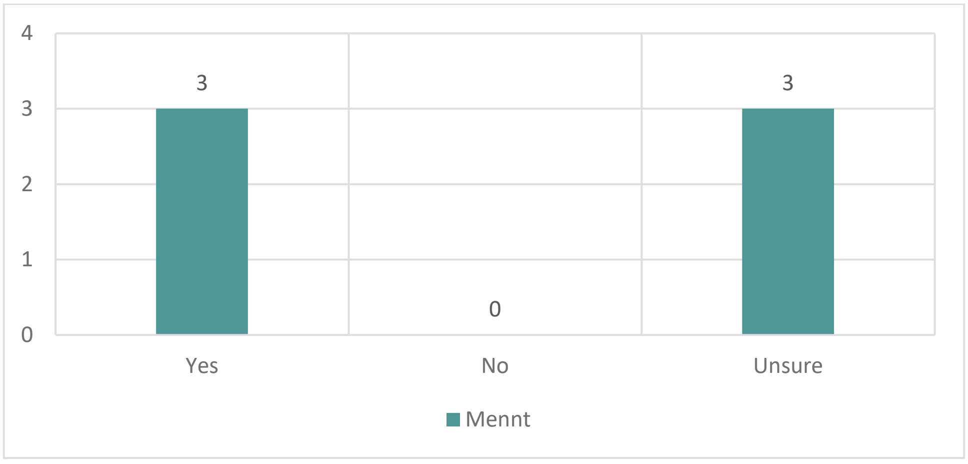


Figure 12 – "Do you think that the current solution would be more usable if there was an easily accessible guide?"

The project team strongly considered building a plugin, or something similar, that would work for timekeeping directly into an e-mail system. There were two main reasons we chose not to do this. One of the reasons was that none of us had any experience with developing plugins for existing software solutions. Even after the team had done some research, we couldn’t figure out if it was even possible to build what we imagined with our current skills. The second reason was that the company is connected through Microsoft Azure, which means that they are using Microsoft calendars. Microsoft calendars can be opened by so many different clients, like Microsoft Store apps, Outlook software, web sites, and more. The employees might be using different services to access their calendars, so even if we built a plugin, it would not work for every type of calendar software they are using.

I had to consider what was feasible within our project’s timeframe. The question about having an accessible guide showed that no one was opposed to having a guide, which seemed like the most feasible option at the time.

From the “other” questionnaire group, which consisted of 14 respondents, the most interesting results were related to what language they would prefer their timekeeping system to be in, and what type of devices they would prefer to use when timekeeping. I have to take into account that these questions were asked with a timekeeping system in mind, and the respondents could have other thoughts about a documentation system, but chances are these results would look similar. These questions were asked in both of the questionnaires, so I have looked at them in relation to what the employees at Mennt answered.

For the question about language preference, the respondents could choose between Norwegian and English, as well as a write-in option. They could choose one or more of these. All of the respondents chose Norwegian as an option, while only 45% chose English (see Figure 13). No one in either of the groups chose to add any languages.

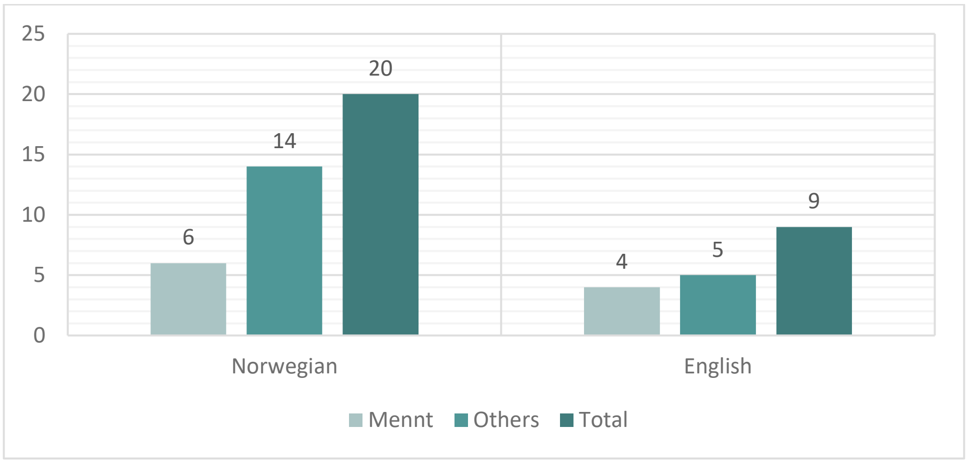


Figure 13 – "Which language would you prefer the timekeeping system to be in? (Choose one or more)"

When asked about preferred devices, the top result was mobile phones, with 80% of all the respondents, followed by desktop and laptop computers with 65% and 60% of respondents respectively (see Figure 14). 15% of the respondents answered tablets and only 5% chose to answer that they did not know.

When only looking at the respondents from Mennt, the “Mobile”, “Laptop”, and “Desktop” options were all answered by all the respondents, with no one answering “Tablet” or “Unsure”.

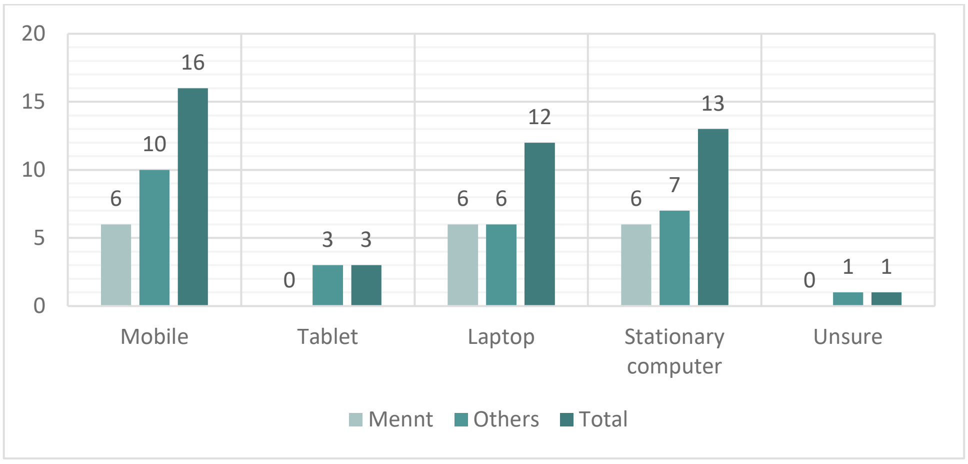


Figure 14 – "What type of devices would you prefer to use to record your working hours? (Choose one or more)"

Some of the open-ended questions provided some useful, general feedback that inspired the project. One respondent said that it is “hard to find the correct task”, and another that there were “too many choices and options in the system – it’s sometimes hard to know what things are”. Having a general guide that explains things like this could potentially help the users overcome these challenges.

When asked about what functionality they were missing, respondents also stated that they missed having a “Norwegian language option” and that they wanted “better search functionality”. These two comments were taken into account in the solution we developed.

From the open-ended question answers, I could also see that respondents wanted very different, often contrasting, features from the timekeeping system. Some employees wanted something simple and minimalistic, while others wanted to have lots of features and configuration options. This correlated well with what I saw in the data I got from the interviews. It was becoming very clear that finding one shared solution for a timekeeping system that would please all the employees at Mennt would be impossible within the project’s timeframe.

3.1.3 COMPETITOR ANALYSIS

Based on what the respondents answered that they wanted in the open-ended questions in the questionnaires, I looked at other web services that offered timekeeping features to see if anyone could already fulfill all the requirements.

First, I tried to identify relevant competitors. I used G2, the software marketplace, to search through all project management software that had timekeeping features, and found that there were a total of 106 listings available that matched our criteria (G2, 2023). This would be too many results to go through, so I limited my search to four of them, ClickUp, Clockify, Timesheet, and Timely. I decided to include ClickUp, the timekeeping service that the company was already using in addition to Azure DevOps. This was done because I wanted to see how it compared to the others.

I made a comparison of the four competitors and selected four key attributes to use when comparing them. The four attributes were selected because they seemed important to the company and the employees. I researched their subscription prices, visual design, and automation features. I also checked all the competitors for integration possibilities, against the five most important ones for Mennt, which were Microsoft Teams, Azure DevOps, Outlook, Visual Studio Code, and Figma.

3.1.3.1 Results from competitor analysis

The results from the competitor analysis can be seen below in Table 1. The four listings were compared by their subscription price, automation, visual design, and integration features.

Most of them had a free tier with basic capabilities, with “pay per user per month” options if more features were needed. Clockify was the cheapest one at 15 USD maximum price per user, but its visual design was somewhat lacking. The best option out of the four seemed to be Timely. It had the most interesting automation feature, with AI tracking, a modern design, and could integrate with all of the options I was checking for. The informants in the interviews and respondents in the questionnaires had told us that they wanted AI tracking, so Timely would be my recommendation for Mennt to check out if they wanted to try a different solution.

Name	Subscription price	Automation	Visual design	Integrations
ClickUp	Free - $29 USD per user per month	Some, with manual setup	Modern	3/5, not VS Code/DevOps
Clockify	Free - $15 USD per user per month	Some	Somewhat modern	4/5, not Figma
Smartsheet	Free - $32 USD per user per month	Some, with manual setup	Modern	2.5/5, no Figma, VS Code, only DevOps through MS Flow workaround
Timely	$9 USD - $28 USD per user per month	AI tracking	Modern	4.5/5, DevOps only through a browser

Table 1 – Comparison of competitors

3.2 Information Architecture

In chapter 2.1.4, I explained what information architecture is (IA) and why it is important for projects like this. In this chapter, I will focus on the specific design methods I used within IA to create an overview of the content for the final design. After deciding to pivot away from creating a timekeeping software, I shifted the focus to developing an internal documentation system for Mennt AS. Given the system's emphasis on users quickly finding the information they need, the information architecture is crucial to the usability. Therefore, I needed to design two types of architecture: one for how the site would be structured and one for the data that would be on the site. The following subchapters will detail how we approached designing these architectures.

3.2.1 DATA ARCHITECTURE

3.2.1.1 Archive content

To get an overview of the type of content that could be placed in the archive that would be the main focus of "Menntor," the project team got together and brainstormed articles that we thought could be useful for the employees to have access to. We wanted the guide's archive to be flexible, expandable, and created in a way that would make it easy for the users to locate the information they needed. The design of the archive content would work as a guideline, but ultimately it is Mennt AS that will have full control over what content to add to their archive.

During the team's brainstorming meeting, we put all of the ideas for potential articles down on sticky notes and placed them on a Miro board. Doing it this way is an example of bottom-up information architecture design, as explained in Chapter 2.1.4. There were a lot of ideas, which we also wanted the company to be able to expand on later, but we figured that if all of these articles were placed onto a web page, it would most likely create information overload for the user and it would be difficult and time-consuming to find the correct one. For this reason, we tried to find articles that had something in common and placed them into matching categories. See Figure 15 for the result.

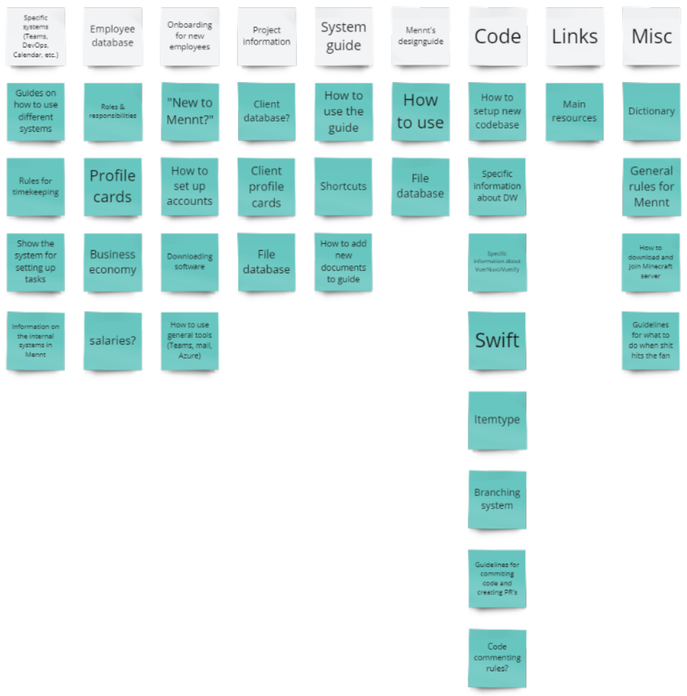

Specific systems (Teams, DevOps, Calendar, etc.)
Guides on how to use different systems
Rules for timekeeping
Show the system for setting up tasks
Information on the internal systems in Mennt
Employee database
Roles & responsibilities
Profile cards
Business economy
salaries?
Onboarding for new employees
"New to Mennt?"
How to set up accounts
Downloading software
How to use general tools (Teams, mail, Azure)
Project information
Client database?
Client profile cards
File database
System guide
How to use the guide
Shortcuts
How to add new documents to guide
Mennt's designguide
How to use
File database
Code
How to setup new codebase
Specific information about DW
Specific information about Vue/Nuxt/Vuetify
Swift
Itemtype
Branching system
Guidelines for commiting code and creating PR's
Code commenting rules?
Links
Main resources
Misc
Dictionary
General rules for Mennt
How to download and join Minecraft server
Guidelines for what to do when shit hits the fan

Figure 15 – Brainstorm of content in the guide

From this point, we created a bigger overview of the potential archive by using a mind map that showed the relations between every category. We also built on the previous ideas and created subcategories to ensure that no single category had too many articles inside and avoid information overload (see Figure 16).

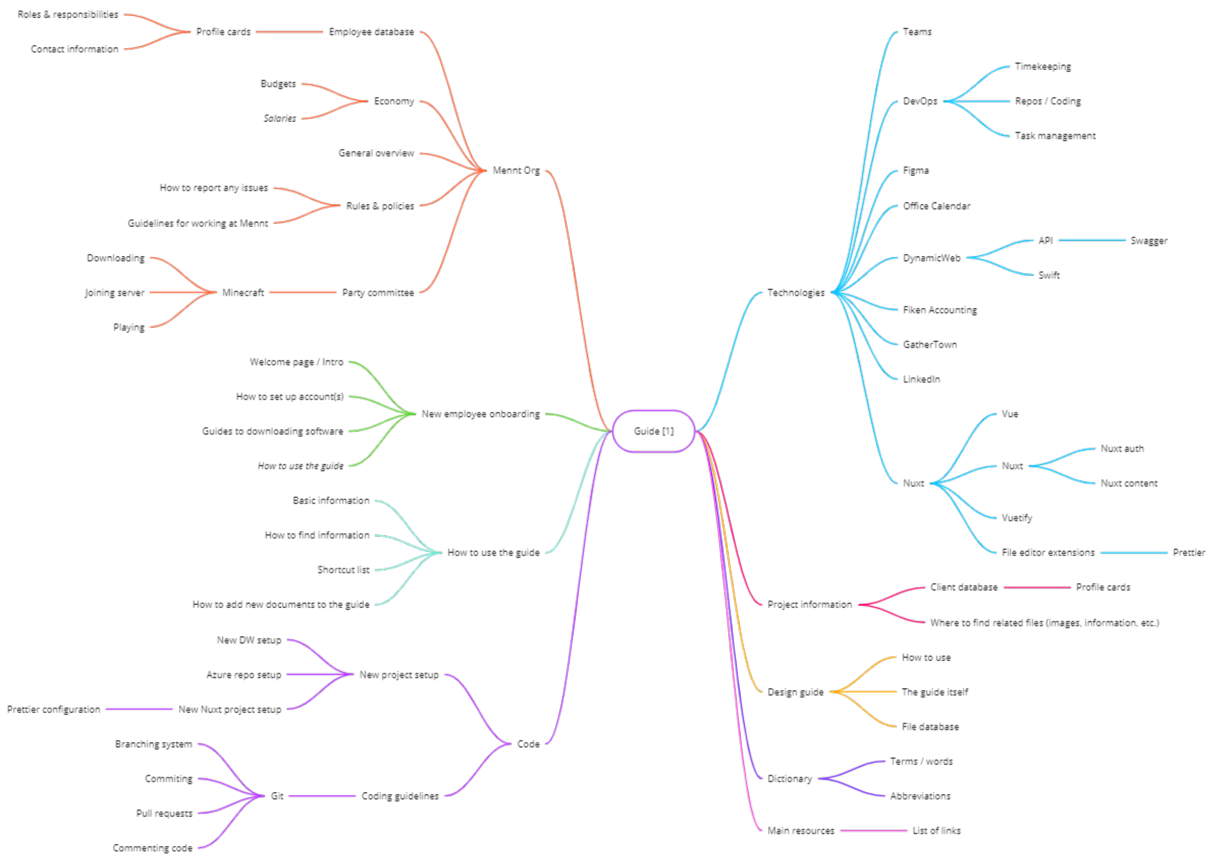


Figure 16 – Mind map that shows the first draft of the content archive

When looking at this overview of the potential database, I realized that we had a lot of top-level categories. Some of them, like *Code* and *Technologies*, could be confused for containing the same things, and others, like "How to use the guide," did not make sense to keep as a top-level category that would always be visible. People who already know how to use the guide would not need to see this category every time they logged in to the web page. That is why I created a new overview of the top-level categories and removed a few of them by placing the current categories that had similar themes into more suitable top-level categories. I also had to restructure the entire database when doing this (see Figure 17 below) I ended up with four main categories – *Code*, *Design*, *Project management*, and *General* (see Figure 18 below). With the new database design, there were way fewer categories, and the navigation from top to bottom makes more sense.

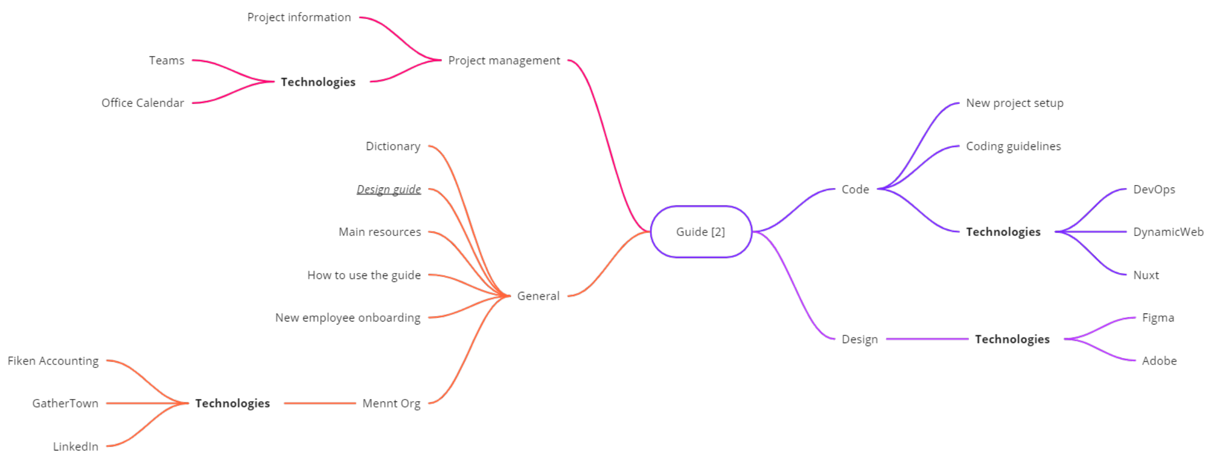


Figure 17 – Revised database suggestion

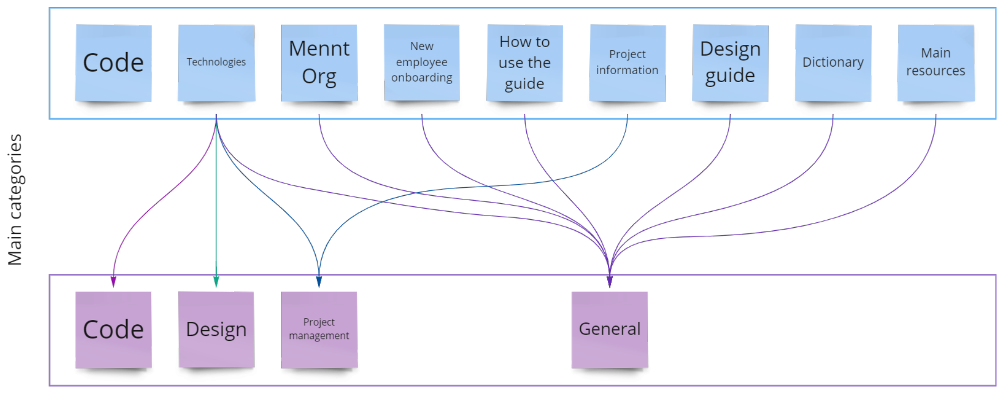


Figure 18 – Main categories, before and after restructuring

3.2.1.2 Database structure

I also needed to create a structure for the database that would hold the content for the web page. The design for the archive content detailed in the subchapter above worked as a guide for the design of the database structure. I knew that I needed information about articles and categories because they were included in this design. I also knew that I needed a way to log in users because the information in the archive should be exclusive to Mennt AS's employees and would therefore need to be secured behind some form of authentication. Through the research, the project group had also come up with an idea about users being able to "favorite" or "bookmark" articles, which would be saved to each user, so this needed to be included in the database structure to provide a way to create relationships between the users and the articles. The final draft of the structure was put into a mind map and can be seen in Figure 19.

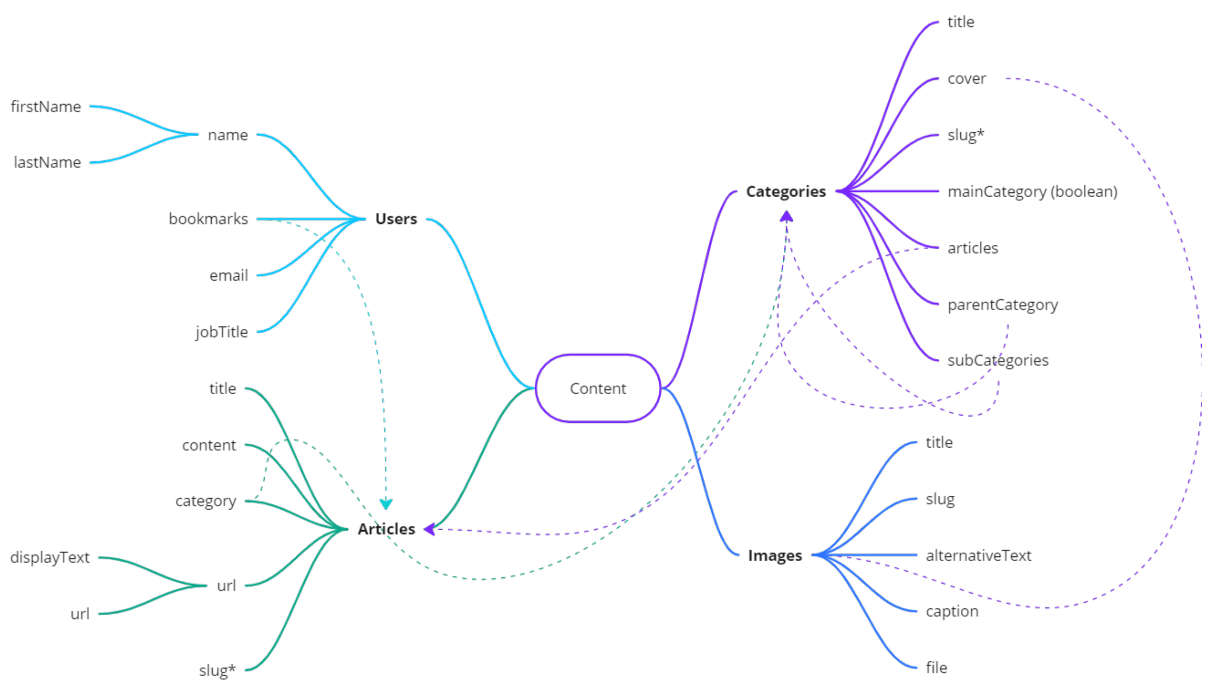


Figure 19 – Content database design

I initially designed the *Categories* collection with multiple levels, where the main categories were on top, then their subcategories, et cetera. This proved to be difficult in practice, and I decided to keep categories as a single collection, where they have a *mainCategory* attribute that is *true* or *false* depending on whether or not they are a top-level category, as well as *parentCategory* and *subCategories* attributes that have relationships with other entities in the same category. That way, I could always keep track of which category is their parent and whether they contain any subcategories. Having this data available allowed the option of creating a hierarchy in the navigational system on the web page.

The two content designs and the database structure provided a good overview of how the web solution should be structured and a good base for further development of the information architecture. The work of creating a good information architecture continued when I created the prototypes, especially in relation to how the users would navigate and look up information on the web page.

3.2.2 WEBSITE ARCHITECTURE

The site architecture of a web page refers to how its content is structured and organized to create a logical hierarchy of information. It determines the ease with which users can find what they are looking for and navigate through the site. To achieve this, I used the design method of site mapping, which helped me create a visual representation of the site's page hierarchy. I created a high-level sitemap, which is a broad overview of the website's page structure and does not include details about individual pages (Rosenfeld, 2015, pp. 395-397).

I used a top-down architecture process while creating this sitemap. I knew that no one outside of the company should be able to see the content of the page, so I needed a page where users could be prompted to log in. In the interviews, I found that the company uses Microsoft Azure and Teams. This means that everyone that works there has a work account that is connected to Microsoft. For this reason, I chose to use Microsoft services as a way to authenticate the users. I decided to have a login page where the users could click a button that would send them through the Microsoft authentication process, but the *Forgot password* and *Register user* pages would be kept outside of the system since they would be managed from Microsoft's end. In the sitemap, which can be seen in Figure 20, you can see that every page that was originally planned for, but that we later realized would be managed by Microsoft, is marked with a blue Microsoft logo.

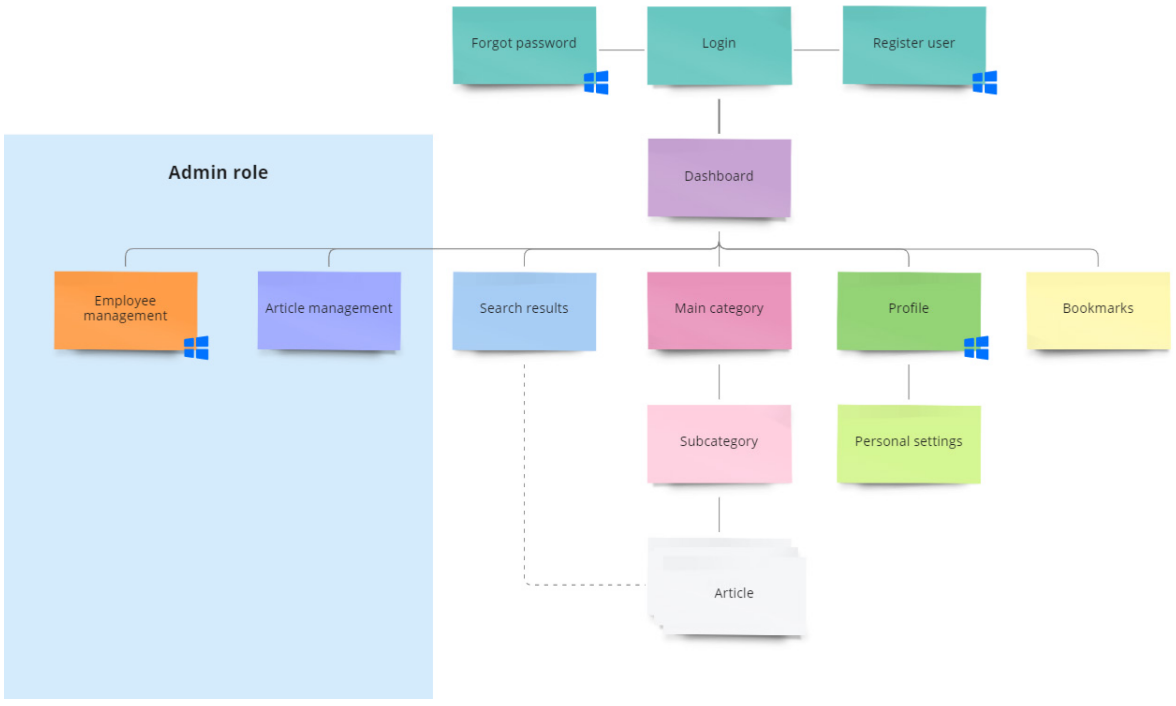


Figure 20 – Sitemap

From the *Login* page, the users will be brought to the *Dashboard*, which works as the main hub for the web page and is where the user will be able to access all other pages. A regular user would be able to navigate to a *Main category* page, their *Bookmarks* page, their *Personal settings* page, or use a search bar that would lead them to a *Search results* page. An administrative user would also be able to access pages where they could manage the categories and articles, but the user management would again be managed through Microsoft's services. In the sitemap, this was visualized by placing a blue background behind the pages that only users with the admin role would be able to access.

The users can either navigate to the categories and articles by going through the hierarchy of categories, e.g.: *Code* [main category] > *Technologies* [subcategory] > *Nuxt* [article], or they could be reached by using the search functionality. I decided that there was a need for search functionality in the system early on because I wanted the users to be able to find information in the shortest amount of time and the shortest number of clicks. If the user has something specific in mind, a search is often quicker than clicking through several levels of categories.

Having a sitemap helped me immensely in thinking about how the users would navigate through the page and look for information. It was very useful when I was creating prototypes and needed to know what pages to prototype. During the prototyping, the sitemap was also used to discuss what functionality and components would exist on the pages.

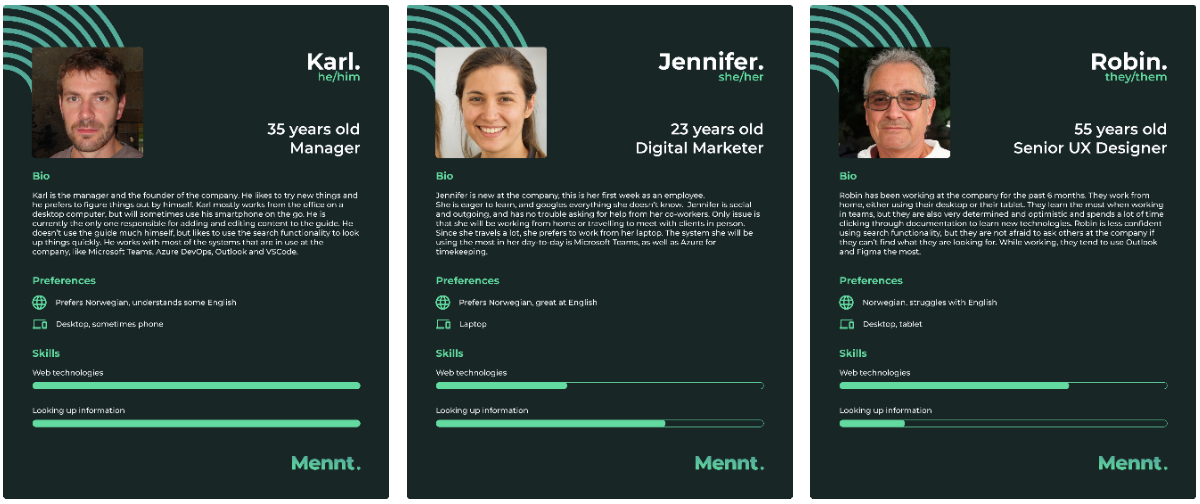
3.3 Design methods

3.3.1 PERSONAS

The results I got from the interviews and questionnaires conducted in the *Empathize* phase were useful in making a set of personas. Personas are fictional characters that represent typical users of whatever you are designing for.

When designing the personas, I focused on attributes that were pertinent to the project. To try and cover all use cases, the personas had differing ages, genders, and job titles, all within the age range of someone that could work at the company. They also had some background information relevant to how they would use the product, like if they usually work from home or at the company location, and their own language and device preferences. At this point in the project, I had decided to move away from the timekeeping solution and towards making a documentation system for the company, so I included their skills with "web technologies" and "looking up information," ranked on a scale from 1 to 5.

I ended up with three personas that helped guide the team when we were making decisions in the design phase. We used storytelling with the personas to imagine what the users would think or do. They were especially useful when creating the prototypes. When I was stuck or uncertain about which direction to go in, I would think about what the personas would want and need and use that to decide.



Karl.
he/him

35 years old
Manager

Bio

Karl is the manager and the founder of the company. He likes to try new things and he prefers to figure things out by himself. Karl mostly works from the office on a desktop computer, but will sometimes use his smartphone on the go. He is currently the only one responsible for adding and editing content to the guide. He doesn't use the guide much himself, but likes to use the search functionality to look up things quickly. He works with most of the systems that are in use at the company, like Microsoft Teams, Azure DevOps, Outlook and VSCode.

Preferences

Prefers Norwegian, understands some English

Desktop, sometimes phone

Skills

Web technologies

Looking up information

Mennt.

Jennifer.
she/her

23 years old
Digital Marketer

Bio

Jennifer is new at the company, this is her first week as an employee. She is eager to learn, and googles everything she doesn't know. Jennifer is social and outgoing, and has no trouble asking for help from her co-workers. Only issue is that she will be working from home or travelling to meet with clients in person. Since she travels a lot, she prefers to work from her laptop. The system she will be using the most in her day-to-day is Microsoft Teams, as well as Azure for timekeeping.

Preferences

Prefers Norwegian, great at English

Laptop

Skills

Web technologies

Looking up information

Mennt.

Robin.
they/them

55 years old
Senior UX Designer

Bio

Robin has been working at the company for the past 6 months. They work from home, either using their desktop or their tablet. They learn the most when working in teams, but they are also very determined and optimistic and spends a lot of time clicking through documentation to learn new technologies. Robin is less confident using search functionality, but they are not afraid to ask others at the company if they can't find what they are looking for. While working, they tend to use Outlook and Figma the most.

Preferences

Norwegian, struggles with English

Desktop, tablet

Skills

Web technologies

Looking up information

Mennt.

Figure 21 – Personas

3.3.2 PROTOTYPING

After I had found what I think is a helpful solution to Mennt's challenges and had mapped out what the main parts of the information architecture would look like, I could move on to the *Prototype* phase of the Design Thinking framework. Prototyping is the process of creating a preliminary or working model of a product, for the purpose of testing and evaluating one or more of its features. The prototype can be a physical model or a digital simulation and is used to identify and address potential design flaws or technical problems before the final product or system is developed. Prototyping is a crucial step in the product development process, as it allows designers and engineers to refine their ideas and improve the overall quality and usability of the final product.

3.3.2.1 Low-fidelity prototyping

To create the first prototype, I used the low-fidelity prototyping method, which involves creating a rough mockup of the design using simple materials like pen and paper or digital tools. Low-fidelity prototypes are often used in the early stages of design to explore different ideas and test basic functionality before investing time and resources into higher fidelity designs.

I started by sketching out some ideas on paper but needed a way to turn them into an interactive prototype that would allow us to perform usability tests in online sessions. I chose to use the online wireframing tool Balsamiq because it allowed me to prototype the layout, features, and navigation quickly and efficiently without worrying about advanced visual design (Steane, 2014, p. 50). This was important because I wanted to focus on functionality and user experience before investing time in a more polished visual design.

Balsamiq's wireframing tool comes with hundreds of pre-designed components that we could easily drag and drop into our prototype (Balsamiq Studios, no date). In the software, I could make these components clickable and trigger a jump to another page in the prototype, which allowed me to create an interactive prototype that I could assess with users in online sessions. To see an example of what the prototype looked like, see Figure 22.

Because of the limited time and resources that I had available for this project, I decided that there was not enough time to design desktop, tablet, and mobile layouts. With over 60% of internet traffic being mobile (Oberlo, 2023), it makes sense that it has become best practice to design for mobile devices before moving on to bigger screens (Wroblewski, 2011). However, in the questionnaires, I found that most of the users would use desktop and desktop applications in their workday, which is why all of our personas prefer at least one device that would use the desktop design. In a developer company, like Mennt AS, I could also assume that the employees will spend most of their time working on their desktops, which is why I decided to only create the prototypes with desktop layouts. I did keep the personas in mind when I was prototyping and thought about how some of them would probably want to check something in Menntor on their phone or tablet, so I tried to limit the amount of content I put on every page and made sure that every component could be put into mobile or tablet layouts later.

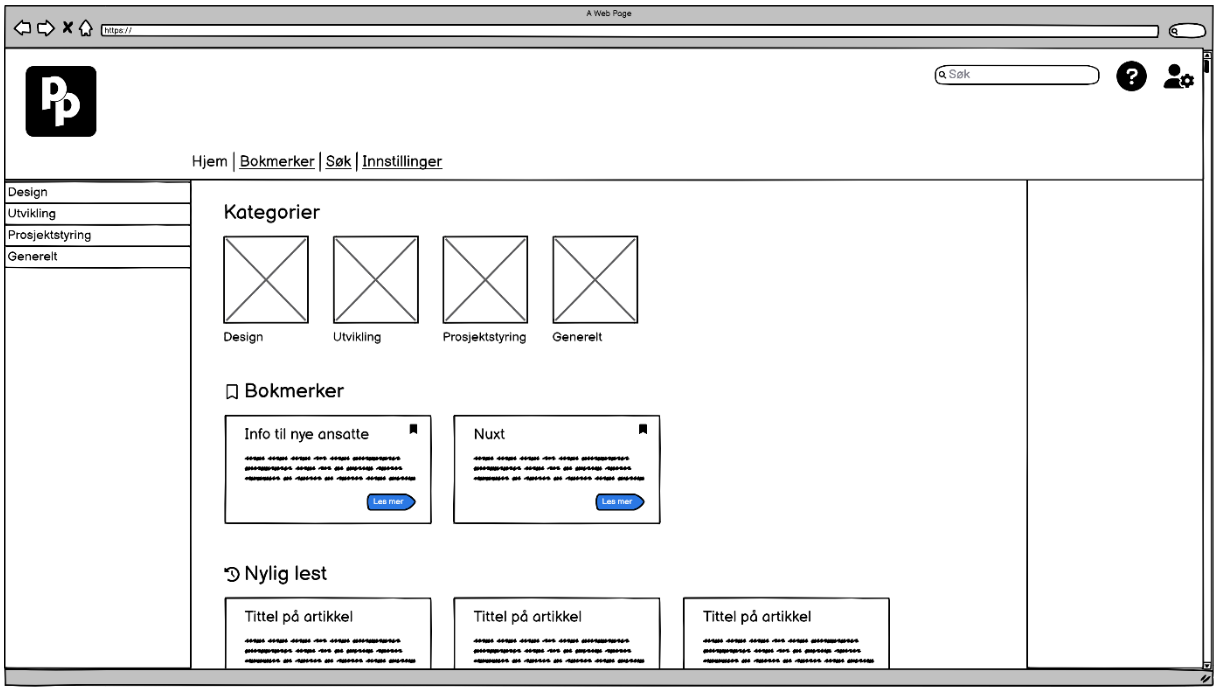


Figure 22 – Dashboard from the lo-fi prototype created in Balsamiq

The main focus when creating this prototype was to design the system in a way that helped the users find the information that they were looking for quickly, and with ease. At the same time as I was prototyping, I created a few tasks that I wanted to use to evaluate the usability in our upcoming usability tests. These revolved around the users being able to log in and navigate in different ways to find content on the page. More details about this can be read below, in the chapter “Usability testing.”

I put a lot of thought into how the users would navigate around the page. Jakob Nielsen found in his research that half of all users are what he calls “search-dominant”, a fifth are “link-dominant”, and the rest use combination of the two (Nielsen, 1997). This is why I wanted to include more than one way for the user to navigate to the content.

In a system where the users will come to the page looking for specific information, it makes sense to have a search system, and the users will most likely expect it to be there (Rosenfeld, 2015, pp. 212-216).

In addition to searching for content, we also wanted the clickable navigation to be easy to understand. To adhere to the fourth usability heuristic from Jakob Nielsen’s “Ten Usability Heuristics for User Interface Design”, which describes how standards should be used, we used both internal and external consistency in our design (Nielsen, 1994; Krause, 2021).

Internal consistency refers to the design being consistent within our web application, while external consistency is about using established conventions that are used across the web. As an example of this, we kept the top level of navigation at the top of the screen and placed the logo for the website in the left corner. The logo was then made clickable and would lead to the home page. Considering that some users might not be aware that this is a normal feature in web pages, I also included a "Home" link in the navigation.

There were initially two ways to navigate between categories. I placed a sidebar on the left side of the screen that showed the names of the categories, where if the user entered one of the categories, the sidebar would expand and show the clickable subcategories within it. There would always be an "overview" tab at the top of every list of subcategories, to allow the user to navigate back to the current parent category. See Figure 23 to see what the closed and opened sidebar navigation looked like.

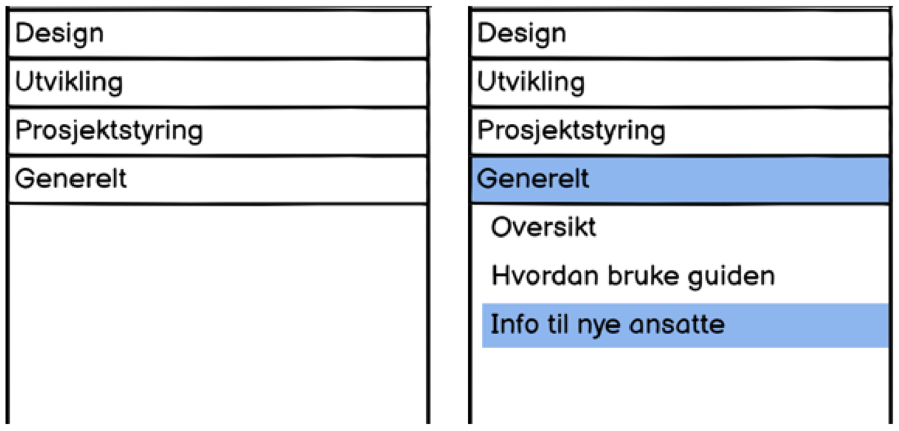


Figure 23 – Sidebar navigation closed (left) and open (right)

The second way that users could navigate through the categories was on the main section of the page. The main page functioned as a dashboard, and the first content on this page would be the main categories of the archive. I wanted them to be prominent on the page because this made sense with how I imagined the users would use the page. Instead of just having links with the names of the categories, I decided to include "cover images" for each of the categories (see Figure 22 above), which could help them stand out from each other and make them more recognizable to the users. The category links were also placed in a module beneath a subheading that says "Categories" to make it more clear to the user what these elements were referring to.

I wanted to hear what potential users thought of the idea of saving articles that they used often, so I designed a way for them to “bookmark” articles and view these bookmarks on a separate page in the system. I wanted these bookmarked articles to be easily available from the dashboard, so I included a “Bookmarks” module below the “Main categories” module that would show the three most recently bookmarked categories (See Figure 22 above). The rest of the bookmarked articles would be available from the “Bookmarks” page, which can be seen in Figure 24.

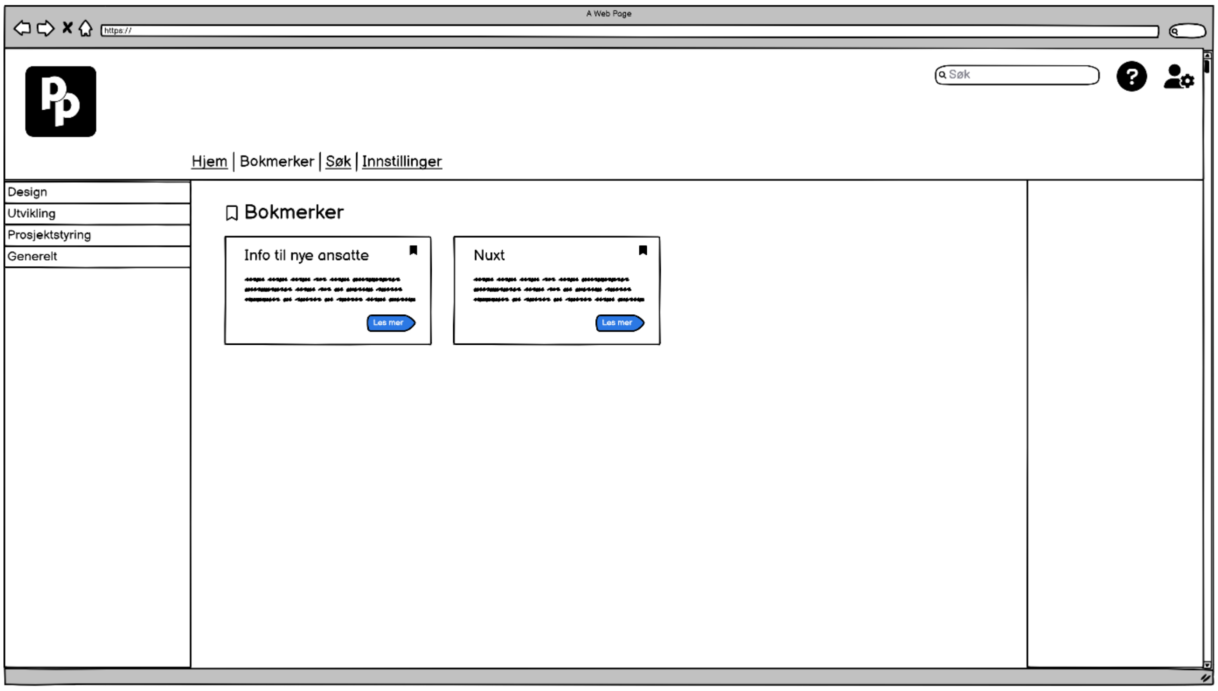


Figure 24 – Bookmark page from the low-fidelity prototype

In addition to the “Main categories” and the “Bookmarks” modules on the main page, I included a “Recently read” module in the main content section that would contain up to three most recently read articles for the authenticated user. I figured this would be useful if the user had closed a tab on accident, or if they were picking up work that they had left off at another time. To ensure that the user knew that the page was scrollable and to avoid “The Illusion of Completeness” (Salazar, 2016, para. 3), I put the articles in the “Recently read” section at the bottom of the page. The articles in the section were not completely visible, to indicate that there is more content to be seen on the page.

At the end of the prototyping, the low-fidelity prototype included the pages *Login*, *Dashboard*, *Bookmarks*, *Article*, *Category*, *Subcategory*, and *Search results*. Some pages that were planned out in the sitemap (see Figure 20) were not included. These were the page for the personal settings and the page where administrative users could add, edit, and delete categories and articles.

The *Personal settings* page was not included, because I was not sure what kind of settings I would need to include on the page, and it did not feel relevant to the usability test at this time. In hindsight, this should have been planned out before we added it to the sitemap, but the page could still become relevant later. The idea was that if I wanted the users to be able to customize their experience, I would need a place in the web solution where they could change these settings. I knew that I would not have time to include this level of customization in the MVP, so I chose not to include any more of this in our initial design.

The administrative pages were not included because at the time I was still unsure about what type of system I would use to store the categories and articles, and whether this would happen in a separate system or within the web solution.

Once all of these prototype pages were finished, the project team performed a usability test that will be explained in chapter 3.4.1 – "Usability testing".

3.3.2.2 High-fidelity prototyping

High-fidelity prototyping is a form of prototyping that involves creating a highly detailed and realistic representation of a product or system. High-fidelity prototypes are designed to closely mimic the final product in terms of appearance, functionality, and user experience. This type of prototyping is often used in the later stages of the design process and can be created using a variety of tools and techniques, including digital mockups, interactive wireframes, and physical models.

The high-fidelity prototyping was created in Figma and can be seen at this link, while the link for the clickable high-fidelity prototype that was used for testing can be found here. I based the design on the low-fidelity prototype, as well as the results I got from the usability test the team had conducted since then and Mennt's design system (see Figure 25 below). My goals with this prototype were to improve upon the first iteration of the design by fixing the issues found in the usability test and to create a coherent visual design that is modern enough that it would not be displeasing to the designers, or the other employees, at Mennt.

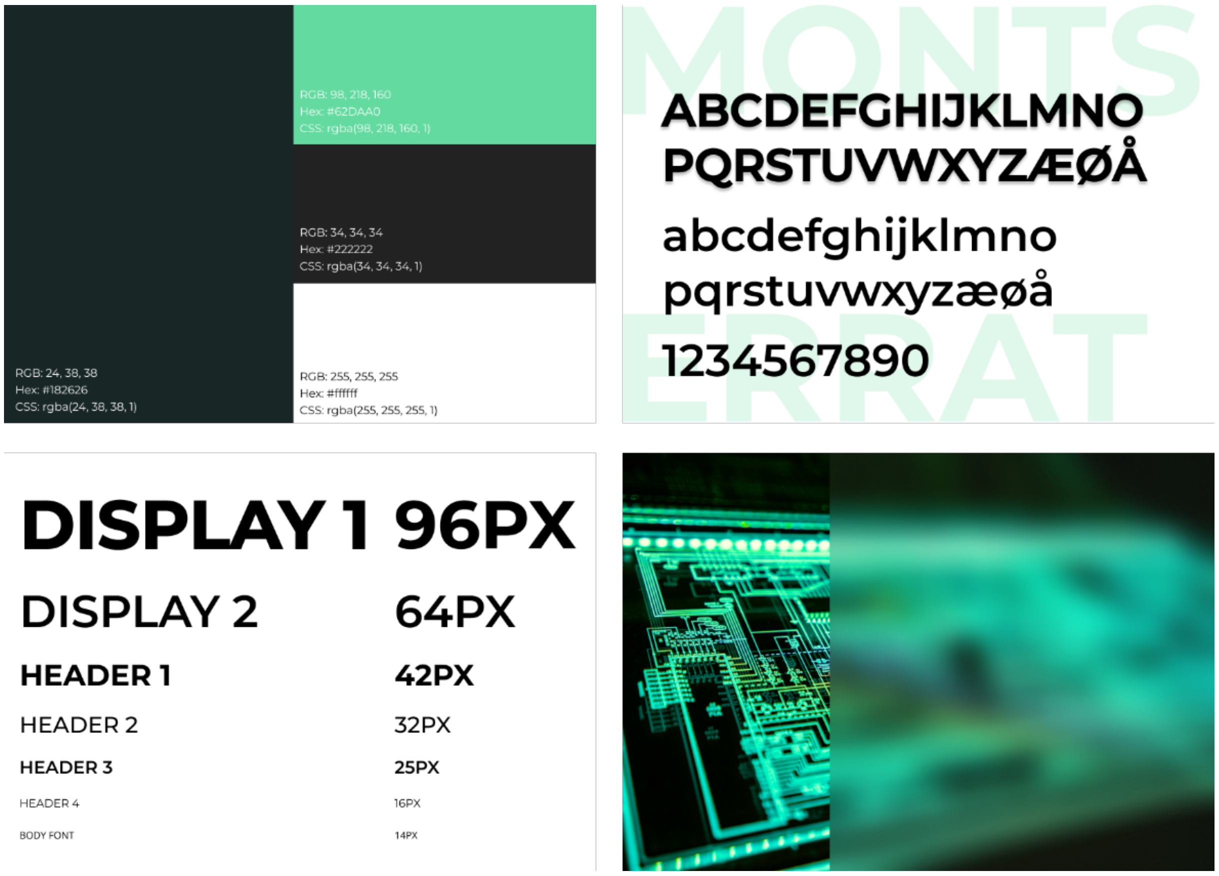


Figure 25 – Font, color, and display items from Mennt's design system

Since Mennt AS had developed a design system that they use for their internal systems, like their company website, it made the most sense to me that I would use that design system in a web solution that would be used within the company. I had a meeting with the designer that had developed the design system to make sure I used it correctly. In the hi-fi prototype, I used their colors, font types, font sizes, and some pre-designed design elements.

When building the prototype, I also used component-driven design, as defined in Chapter 2. Inspired by Atomic Design methodology (Frost, 2016), I started by building out our components from the smallest pieces, categorized as “atoms”. The atoms were then combined with each other, or with other elements, to create “molecules.” An example of two atoms being used in a molecule component, combined with text types and colors from Mennt’s style guide, can be seen in Figure 26.

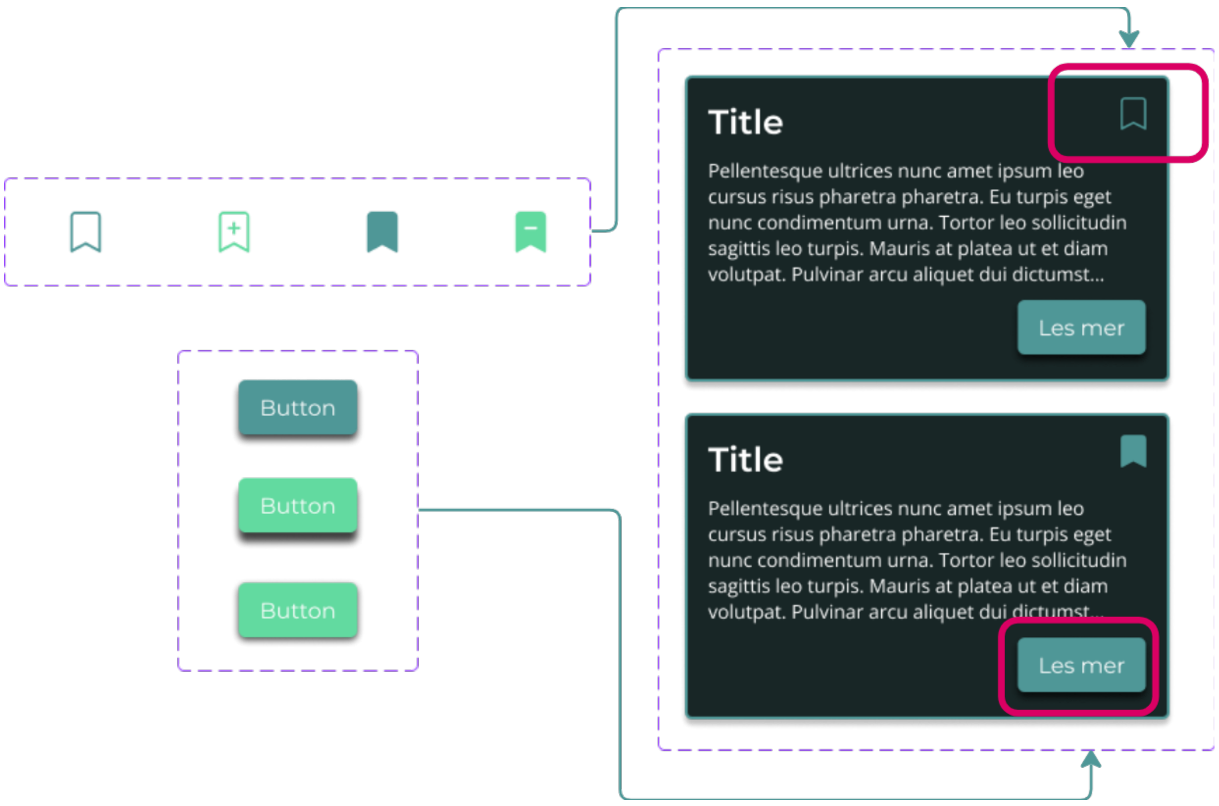
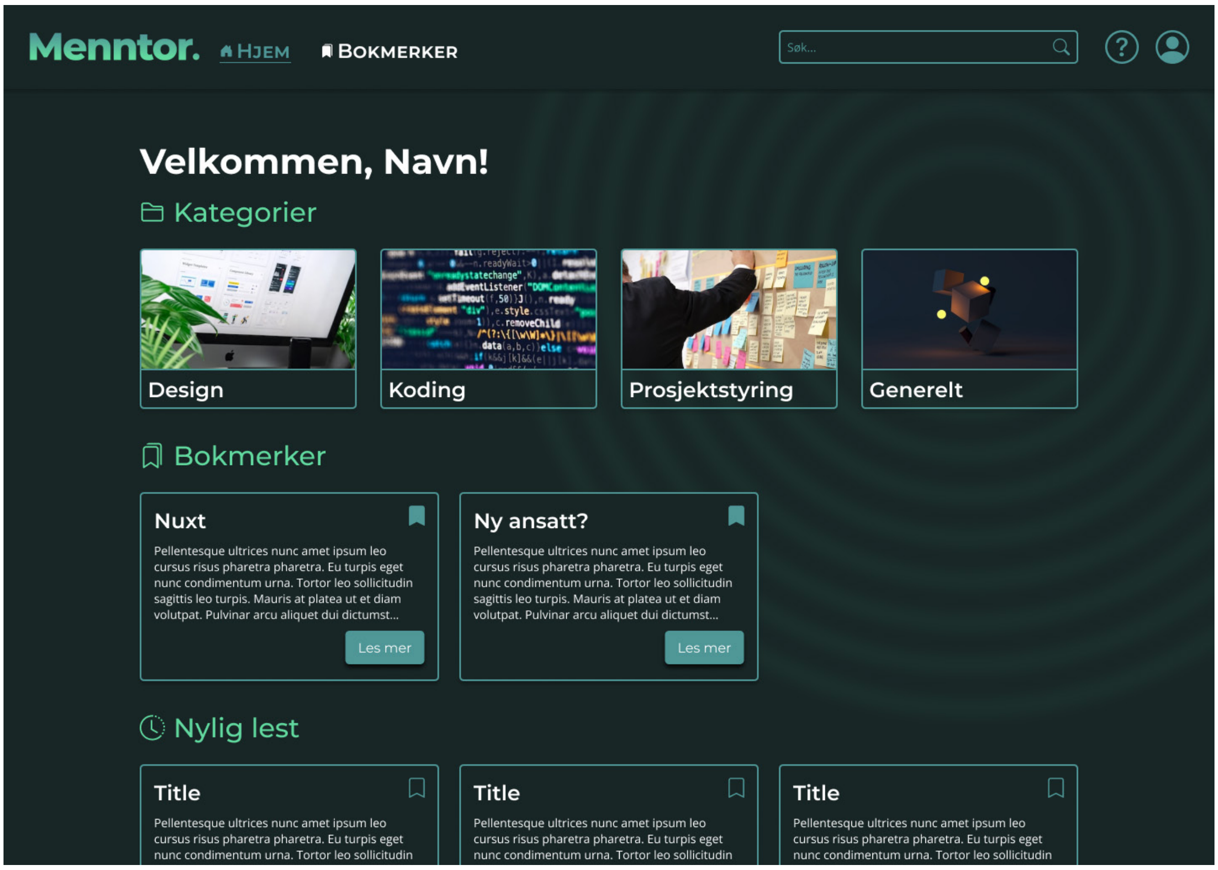

Title
Pellentesque ultrices nunc amet ipsum leo cursus risus pharetra pharetra. Eu turpis eget nunc condimentum urna. Tortor leo sollicitudin sagittis leo turpis. Mauris at platea ut et diam volutpat. Pulvinar arcu aliquet dui dictumst...
Les mer
Button
Button
Button
Title
Pellentesque ultrices nunc amet ipsum leo cursus risus pharetra pharetra. Eu turpis eget nunc condimentum urna. Tortor leo sollicitudin sagittis leo turpis. Mauris at platea ut et diam volutpat. Pulvinar arcu aliquet dui dictumst...
Les mer

Figure 26 – Two atom components being used in the creation of a molecule component

Creating components like this in Figma saved me a lot of time. Figma has a built-in feature for creating components, where you can create variants within the component. This way, I could have different states for each of the atoms that will be triggered by user interactions. As an example, the button shown in the figure above had three states: *default*, *hover*, and *active*. Once a user hovers over the button, the component will animate into its *hover* state; the background color of the button will change, the drop-shadow will move further out from below the button, and the entire button element itself will move up a few pixels. This way, it seems to the user as if the button is moving up and out from the page. Once the user clicks the button, the *active* state will be activated, the drop-shadow will become smaller at the bottom of the button, and the element will move several pixels down on the screen as if the button was being pressed further into the screen. These state changes would also work when the atoms were used in combination with other components, which meant that if I wanted to change the functionality or look of one of the components, I could do this at the smallest possible level, and it would change across the entire prototype, and inside all the other components. This ensured consistency in the design across all the pages.



Velkommen, Navn!

Kategorier

Bokmerker

Nuxt

Pellentesque ultrices nunc amet ipsum leo cursus risus pharetra pharetra. Eu turpis eget nunc condimentum urna. Tortor leo sollicitudin sagittis leo turpis. Mauris at platea ut et diam volutpat. Pulvinar arcu aliquet dui dictumst...

Les mer

Ny ansatt?

Pellentesque ultrices nunc amet ipsum leo cursus risus pharetra pharetra. Eu turpis eget nunc condimentum urna. Tortor leo sollicitudin sagittis leo turpis. Mauris at platea ut et diam volutpat. Pulvinar arcu aliquet dui dictumst...

Les mer

Nylig lest

Title

Pellentesque ultrices nunc amet ipsum leo cursus risus pharetra pharetra. Eu turpis eget nunc condimentum urna. Tortor leo sollicitudin

Title

Pellentesque ultrices nunc amet ipsum leo cursus risus pharetra pharetra. Eu turpis eget nunc condimentum urna. Tortor leo sollicitudin

Title

Pellentesque ultrices nunc amet ipsum leo cursus risus pharetra pharetra. Eu turpis eget nunc condimentum urna. Tortor leo sollicitudin

Figure 27 – Dashboard from high-fidelity prototype

The most noticeable change from the first to the second prototype was in the visual design. In Figure 27 you can see the "Dashboard" from the high-fidelity prototype, which is the same page that was shown in Figure 22 on page 41. In this screenshot, you can see that I have added Mennt's colors and fonts to all the elements. I also created a logo for the project, based on the Mennt logo. This can be seen in Figure 28 below.

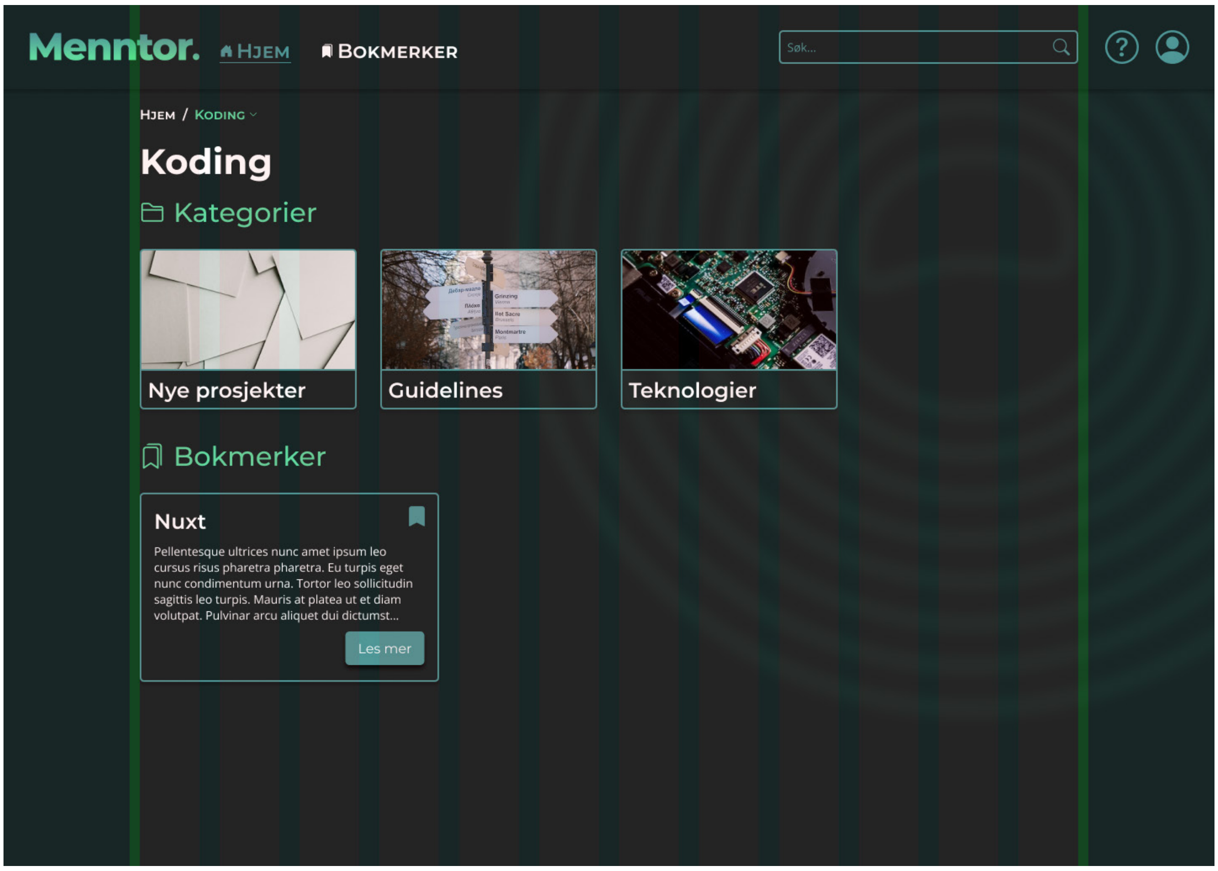
Mennt. → Menntor.

Figure 28 – Logo change from company logo to web system logo

I chose to go with a darker design with rounded edges because that is what Mennt uses on their home page. Keeping in mind that the company might want to have a "light mode" implemented on Menntor in the future since light mode can be better for the users' performance (Budio, 2020), I decided to take steps in the prototyping and development stages that would make this easy to design for and implement. In the prototype, I saved all the colors as local styles, which means that if you change the color of that style later on, all the components that use the color will also change. The same thing was done for font styles, gradients, box shadows, and grid layouts.

A grid layout was created for the content section of the page. The grid is 1320px wide and consists of 12 columns with gutters of 24px in between them. There is also a spacer on each of the sides that is 12px wide, to ensure that the content never goes all the way to the edge of the browser.

Below, in Figure 29, you can see an example of this. This is a *Category* page, that shows a module with subcategories, as well as a module with bookmarks within that category. Here, you can see that the category components span 3 columns in the grid, while the article component that has a little more content spans 4 columns. All the other elements, the breadcrumbs and titles are aligned to the left of the grid.



Pellentesque ultrices nunc amet ipsum leo cursus risus pharetra pharetra. Eu turpis eget nunc condimentum urna. Tortor leo sollicitudin sagittis leo turpis. Mauris at platea ut et diam volutpat. Pulvinar arcu aliquet dui dictumst...

Figure 29 – Category page – "Coding" - with grid layout showing, from the hi-fi prototype

In the figure above, you can also see that I created a hierarchy of the titles on the page by giving the page name the biggest size and a color that stands out more, and the subtitles have a smaller size. This was done according to the first heuristic in Jakob Nielsen's 10

Usability Heuristics (see Chapter 2.1.2), which states that the user should always be informed about the system status, and knowing which page they are on is crucial to this.

The subtitles also have an icon next to their names that communicates the content of the module. These icons are used across the prototype in the content modules, as well as in the navigation header, as you can see in the previous figure. The reason for this change was to get the users to recognize these links and headings faster, which could speed up the time they use to navigate (Harley, 2014).

3.3.2.3 Changes from lo-fi to hi-fi based on user feedback

I also made a lot of changes based on the feedback that was received from the users in the usability test of the low-fidelity prototype. Some of the feedback I got was due to the first prototype being low-fidelity and thus lacking some functionality. I added visual feedback when the user hovers over the bookmarking icon, where it would change color, and a little "plus" or "minus" icon would show up in the middle depending on the bookmarked state. The states can be seen in Figure 26 on page 46, where the bookmark icon component was used as an example.

A few users in the test mentioned that they were confused by the search functionality in the first prototype. In that prototype, when the user clicked the search field in the navigation bar, they were instantly led to the search results page. No functionality was added to the search field in the prototype because the goal was to check whether users who were asked to search for something would use the search bar in the navigation or the "Search" link that would lead them to a different page where they could search. I asked follow-up questions and all the users who were confused by the first prototype expected there to be a dropdown list below the search field in the navigation that would show a few matching results. In the high-fidelity prototype, I created an interactive component that was placed in the navigation bar at the top right of the screen. The component in the layout can be seen above in Figure 27 and Figure 29, and the component with all its variants can be seen below in Figure 30.

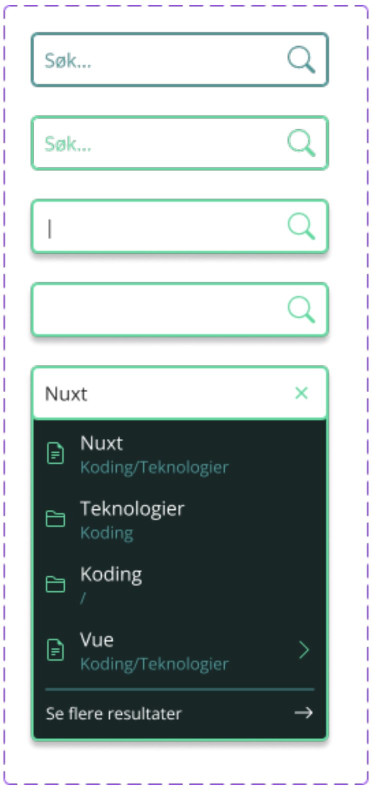


Figure 30 – Search field component with states Default, Hover, Active, Active – blink, and Searching

There were also some things that users brought up that I had completely missed, like them expecting the "Bookmarks" heading title on the main page to be clickable and lead to the *Bookmarks* page. This was fixed in the hi-fi prototype and considered for all the headings being used for modules from then on.

One user also stated that they expected the most recently bookmarked article to be on the far left of the overview, not the last one on the right. The users were asked in the usability test to bookmark an article named "Nuxt", which then updated the main page to include the bookmarked article. The bookmarked article was placed to the right of the other bookmarked article on both the *Dashboard* and the *Bookmarks* pages. This was an oversight on my part, but using common web conventions, having the last bookmarked article as the first item in the row makes the most sense and was fixed in the new prototype. The change from the first to the second prototype can be seen below in Figure 31.

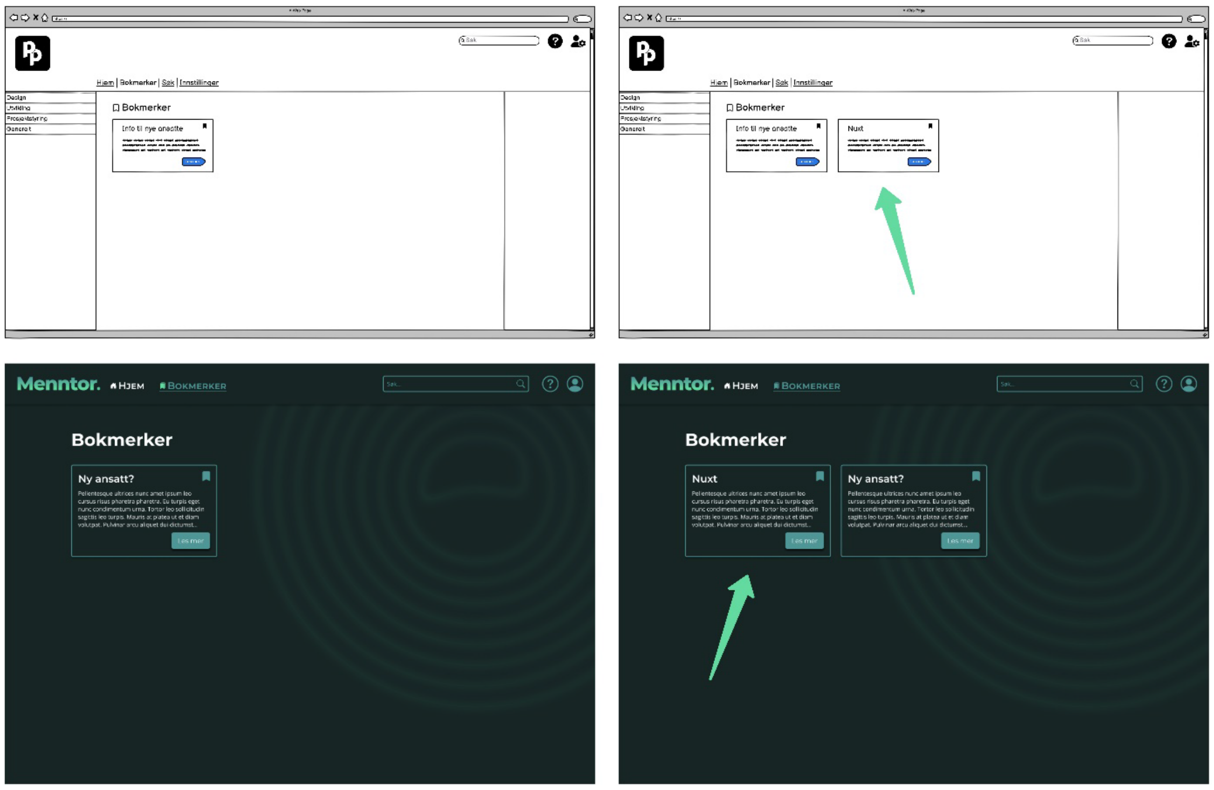


Figure 31 – Comparison of the Bookmarks page in hi-fi and lo-fi prototypes (green arrow pointing out the placement of the most recently bookmarked article)

Even though I got feedback about the users wanting to have the option to filter and/or sort articles and bookmarks, as well as search results, I did not implement this in this iteration of the prototype because the functionality was going to be dependent on the setup of the database. It would also be very time-consuming to set up and test the functionality of this in the prototype, but I figured that as long as the database collections are designed and set up properly, it should not be an issue to add something like this to the pages later.

For the *Login* page, I changed the text and location of the Microsoft login button after receiving feedback about it from a user. Logging in with an MS account was going to be the main way for the employees to log in since every employee had access to one. When adding visual styling to the button, I had to follow Microsoft's branding guidelines, which meant that I couldn't choose a color that would stand out more from the page (Microsoft, 2023). As you can see in the figure below, the MS login button was placed as the first login option on the page, and the form where users would be able to fill out their username and password was placed at the bottom. I wanted this page to stand out and be different from the others in the system, so that there would be no confusion for the user about whether they were logged in. This is why this page is the only page in the prototype that does not follow the grid layout I mentioned before since it is a page that only unauthenticated users see.

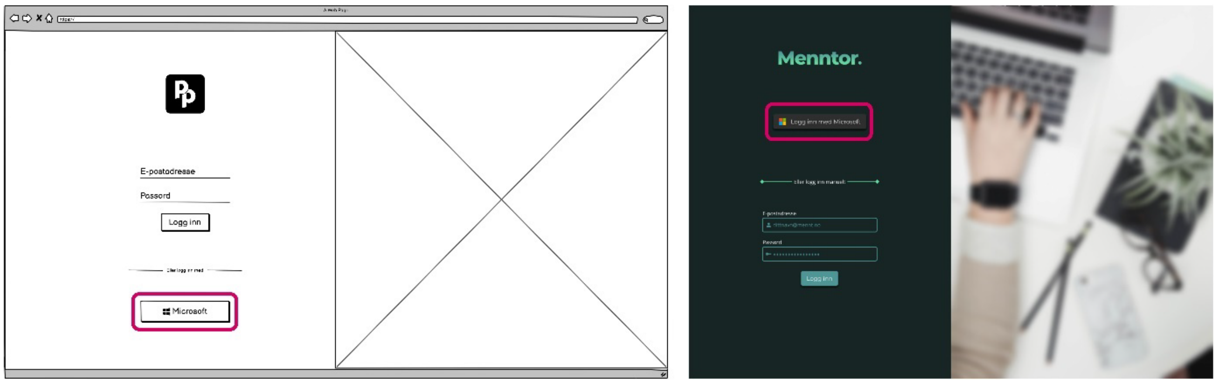
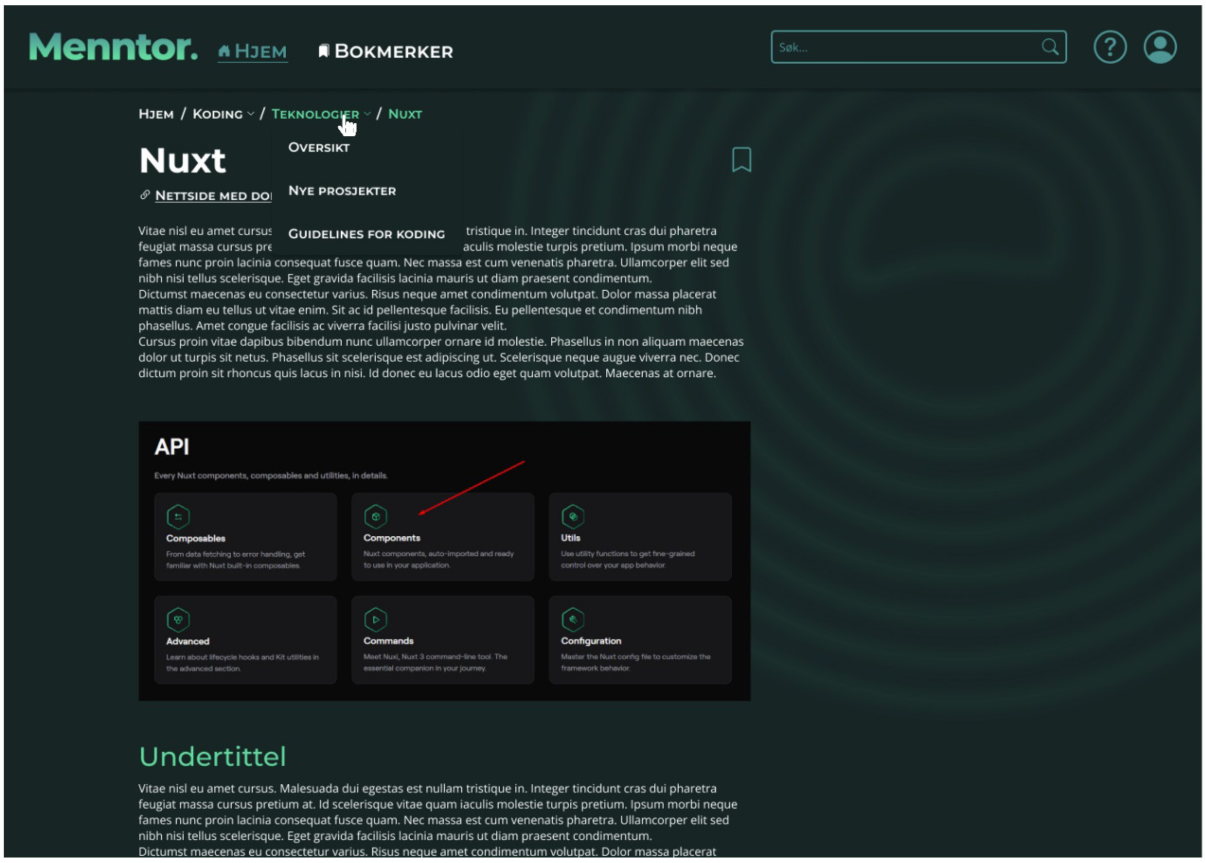


Figure 32 – Login page from lo-fi (left) and hi-fi prototypes (right) that shows the changed placement of the login button

Some changes were made to the navigation, as well. Most of the users that I tested the previous prototype on managed to find their way around the pages easily, but there was some confusion about how to go back to one category when moving into subcategories. As mentioned in the previous chapter, I made this possible by having an "overview" tab in the side menu below every open category (see Figure 23). Since this was not clear to the users, I chose to remove the side menu entirely and created breadcrumbs in their place on all pages related to the categories. This way, the users could navigate down the category tree by clicking the category components and navigate back up the tree by using the breadcrumbs. I chose not to use the breadcrumbs on the *Bookmarks* and *Search* pages since they did not have any deeper levels, and breadcrumbs would not make sense there. See Figure 33 for an example of how the breadcrumbs look on an *Article* page when you hover over the name of the article's parent category.



Nuxt

Nettside med do

Vitae nisl eu amet cursus … tristique in. Integer tincidunt cras dui pharetra feugiat massa cursus pre… aculis molestie turpis pretium. Ipsum morbi neque fames nunc proin lacinia consequat fusce quam. Nec massa est cum venenatis pharetra. Ullamcorper elit sed nibh nisi tellus scelerisque. Eget gravida facilisis lacinia mauris ut diam praesent condimentum.
Dictumst maecenas eu consectetur varius. Risus neque amet condimentum volutpat. Dolor massa placerat mattis diam eu tellus ut vitae enim. Sit ac id pellentesque facilisis. Eu pellentesque et condimentum nibh phasellus. Amet congue facilisis ac viverra facilisi justo pulvinar velit.
Cursus proin vitae dapibus bibendum nunc ullamcorper ornare id molestie. Phasellus in non aliquam maecenas dolor ut turpis sit netus. Phasellus sit scelerisque est adipiscing ut. Scelerisque neque augue viverra nec. Donec dictum proin sit rhoncus quis lacus in nisi. Id donec eu lacus odio eget quam volutpat. Maecenas at ornare.

API

Every Nuxt components, composables and utilities, in details.

Composables
From data fetching to error handling, get familiar with Nuxt built-in composables.

Components
Nuxt components, auto-imported and ready to use in your application.

Utils
Use utility functions to get fine-grained control over your app behavior.

Advanced
Learn about lifecycle hooks and Kit utilities in the advanced section.

Commands
Meet Nuxi, Nuxt 3 command-line tool. The essential companion in your journey.

Configuration
Master the Nuxt config file to customize the framework behavior.

Undertittel

Vitae nisl eu amet cursus. Malesuada dui egestas est nullam tristique in. Integer tincidunt cras dui pharetra feugiat massa cursus pretium at. Id scelerisque vitae quam iaculis molestie turpis pretium. Ipsum morbi neque fames nunc proin lacinia consequat fusce quam. Nec massa est cum venenatis pharetra. Ullamcorper elit sed nibh nisi tellus scelerisque. Eget gravida facilisis lacinia mauris ut diam praesent condimentum.
Dictumst maecenas eu consectetur varius. Risus neque amet condimentum volutpat. Dolor massa placerat

Figure 33 – Nuxt article page that shows breadcrumbs hover interaction

Another thing you can see on the *Article* page is that I left a lot of room on the right side of the grid layout. I did this intentionally because there was talk from some of the employees of the company that took part in the first usability test, that they might want to see things like related articles or links there in the future.

I also made changes to the *Search* page. Previously, all the search results were grouped together, but I decided to split them into "Top result", "Articles", and "Categories" to lessen the cognitive load for the user. See Figure 34 below for a comparison of the two prototypes. The final result was a prototype that was used in the second usability test, which will be discussed in Chapter "3.4.1 Usability testing".

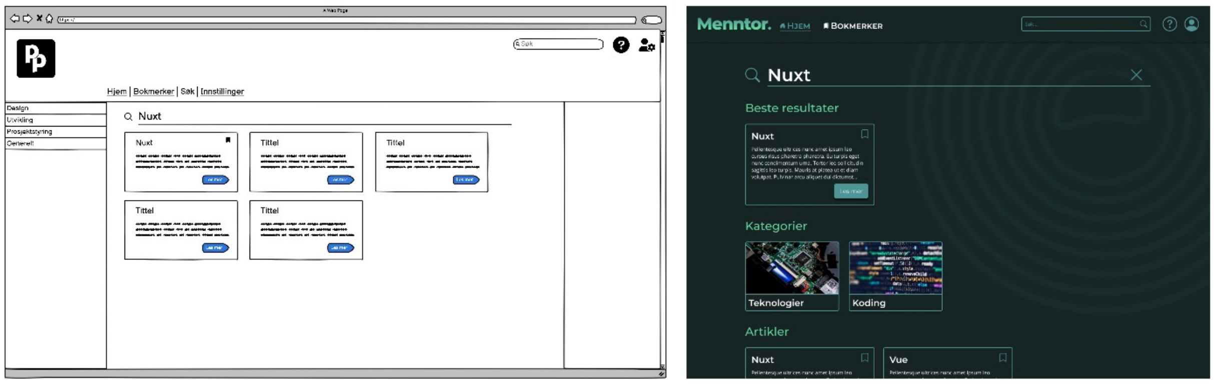


Figure 34 – Comparison between lo-fi (left) and hi-fi prototype (right) that shows the difference in search results

3.4 Testing methods

As mentioned in the Design Thinking chapter (see 2.1.1), the Prototype phase is followed by a Test phase, where you take the solution that you came up with and test it on real users to get feedback that you can use to improve the solution. In this project, I used usability testing and heuristic evaluation as my methods to test the prototypes that were created.

3.4.1 USABILITY TESTING

Usability testing is a critical step in a user-centered design process that helps ensure that the products we create are not only aesthetically pleasing but also user-friendly. By testing the product with real users, designers can gather valuable feedback on the usability and user experience of the design. This feedback can be used to identify any pain points, confusion, or frustration that users may encounter while using the product, and to make necessary improvements. Below, I will describe how I tested the usability of our low-fidelity and high-fidelity prototypes, and how the results were analyzed.

3.4.1.1 Choosing the participants

To test the usability of the prototypes, I conducted usability tests with five real users. The reasoning behind choosing only five participants for the test is based on research done by Jakob Nielsen and Thomas K. Landauer that showed that after five people, the reward you get from testing on more users is minimal, and is likely not worth it (Nielsen, 1993). For both of the tests, I tested three participants with no relation to the company, as well as two employees. I wanted to test on Mennt's employees to ensure that I got feedback that would be the closest to real-world use. I could not use more than two employees per test because there were not that many employees at the company, and I did not want to test the same person twice. After all, I wanted them to have no prior knowledge of how to navigate the system, because this could affect how they performed in the test.

For every person I tested, I made sure to ask their age, occupation, and experience level with similar systems. The age data from the participants in the test, as well as the ages of the employees from Mennt at the time they filled out the questionnaire, can be seen below in Table 2. From the table, you can see that the mean age of everyone that participated in the usability tests is 6.7 years higher than the mean age of the employees at Mennt. However, there was an outlier in the usability test group, with an age of 70. I think that it is important to include users outside of the user group, not only when it comes to age, but also varying occupations and experience levels. This was especially important to me in this project because one of the goals was to future-proof the system.

My second research question asked about ways to make the system sustainable. For something to be sustainable, it would have to last for a long time. If the system was only designed for the employees that currently work at Mennt, this would maybe not be the case, but I also want the usability to be in place for all future employees of the company. This is important, as the company is quickly growing and will most likely change out and get new employees in the future. A few of our participants had no prior experience with this type of system. I thought about whether gender should play a role in our choice of participants but ultimately decided that it was irrelevant to note down any data about it. Users of different genders were still included in the usability tests.

Age of users	Data set	Mean	Median	Min	Max
Test 1	24, 70, 27, 31, 28	36	28	24	70
Test 2	47, 38, 32, 31, 39	37.4	38	31	47
Both tests	-	**36.7**	**31.5**	24	70
Mennt	27, 27, 28, 30, 30, 38	**30**	**29**	27	38

Table 2 – Age data for users in the usability test

3.4.1.2 Preparing the tests

After I had selected the types of participants I would test on, I needed to prepare some material for the test. Once the low-fidelity prototype had started taking form, I had written down some tasks that I wanted the users to complete. All the tasks can be seen in Appendix F, which contains the script for the usability test. I wrote a script to ensure that all the tests were performed in a similar fashion and that all the participants were given the same information and tasks. I figured this would help me later when I had to compare the results.

The script I created contains an introduction that gives information to the user about the project, what was going to happen in the test, and some encouragement about how there were no wrong answers from their side. I made sure to include this to make the participants feel more at ease before the test.

I then wrote down some easy questions for the user, about their age, occupation, and experience level. This way, I started off easy to get the participants talking and to lessen any stress before starting the usability tasks.

Before the user was given the task, I had written out an explanation covering how the tasks were going to be conducted. This included telling the user that I would stop between each step to hear their thoughts about the task, but that I also wanted them to speak out loud as much as they could while they were doing each task.

There were three sections of tasks. The first task was about finding a bookmarked article, but the user had to log into the system first. In the second task, they were told to navigate to a specific article and bookmark it. The third, and final task was to find an article by searching for it. Each of the tasks had several steps that we would give the user. That way, I could stop and hear their thoughts about all the little details they might otherwise forget if they navigated through several pages and steps at once.

Finally, in the usability test, I had written out three follow-up questions, a "reflection section". The three questions were:

1. Was there anything you particularly liked about the system?
2. Was there anything you particularly disliked about the system?
3. If you could change anything about the system, what would you do differently?

These questions were meant to push the participants to think more deeply about how they felt while using the system and try to drag both positive and negative comments out of them.

3.4.2 CONDUCTING THE TESTS

Before the first user test, I conducted a pilot test on one user, where the goal was to correct anything unclear in the script. This proved to be useful, as there were some steps in the tasks that seemed clear to me when I wrote them but turned out to be difficult for someone with a more limited understanding of the system.

The usability tests for the low-fidelity prototype were conducted online, using Discord and TeamViewer, two communication platforms with screen-sharing features. Ideally, there would be two team members available, so that one person could focus on instructing and talking to the user, while the other could write down detailed notes about what happened in the test. At the time, this was not possible, so I conducted the first round of usability tests alone. To make sure that this impacted the test results as little as possible, I planned to use longer time for each test, to ensure that I would have enough time in between each task to take notes about everything. The participants were informed that there would be a little bit of a wait time between each step in the tasks because of this.

For all of the tests, I tried to keep them as similar as possible by sticking to the script, with one exception. In one of the tasks, the participants were asked to navigate to an article related to a technology that is used in the company. I wanted to see if the employees at Mennt could figure out where to find this article without any guidance, but participants who were unfamiliar with the technology would be at a disadvantage here. This is why I chose to give the non-employee participants information about what categories the article would be placed under.

In the second usability test, for the high-fidelity prototype, the same script was used. I wanted to see if the changes from the first prototype had improved the usability test, so I figured that it would be best to conduct the same test over again. However, there were some differences in how the tests were conducted, mainly that the second usability tests were done with the participants present in-person, and with more than one member of the team present.

Due to concerns about available time, the second usability test was performed by the two employees from Mennt that were part of the project team, at the same time as the development of the coded web solution was happening. Ideally, I would have waited to see what the results were and changed the web solution accordingly, but I did not have enough time to do this. The analysis of the data from the first usability test happened right after the test, while the analysis from the second usability test was done after the development was done, during a meeting where the whole team was present.

3.4.2.1 Analyzing the test data

I analyzed the data from the usability tests in two steps. First, I went through all the transcripts from the usability tests and placed all the insights into a grid where the columns were sorted by the participant and the rows by what functionality in the system the insight was related to. There were six categories – *Login with Microsoft account*, *Bookmarks*, *Navigation to main page*, *Navigation to article*, *Search for article*, as well as a category named *Other* – that contained any other insights collected from the participants.

For step two of the analysis, I sorted the insights by placing similar insights on top of each other to see how many times they occurred. In Figure 35 below you can see the final result of the second usability test, where if sticky notes were stacked, every other sticky note would have a differing color to make the stacks more visible. The stacking was done to get a feel for what insights were the most important, although I looked through all of them and some insights were noted down as highly important regardless of how many times they came up. The grids with the analysis from both of the usability tests can be seen in Appendix G.

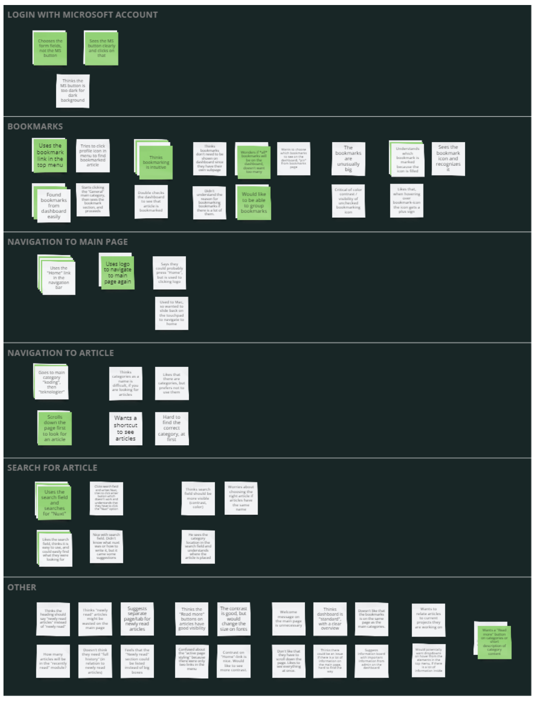


Figure 35 – Results from the second step of the high-fidelity prototype test

The data that was collected in the first usability test was used to improve the design and guided the decision-making when making improvements in the next prototype. Detailed information about the changes can be found in Chapter 3.3.2.3.

The data from the second usability test was not available until after the development of the final design had finished, so this data will be handed over to Mennt to guide any further development. However, I did make a few changes during development that correlated to the user feedback that was given in the second usability test.

3.4.2.2 Results from the second usability test

In this section, I will go through the most important results from the second usability test that was tested using the high-fidelity prototype. The results will be split into the categories “login”, “navigation and labeling”, “bookmarks”, “search functionality”, and “other”.

Login

The users were asked to log in with their Microsoft account in the tests. As mentioned in chapter 3.3.2.3, the “Login with Microsoft”-button had been moved to the top of the content section to make it more visible (see Figure 32 on page 52). Some test users were still confused by this and tried to select the form fields to log in. In the Menntor MVP, these form fields were removed.

Navigation and labeling

By focusing on the navigation aspect in the usability tests, I ensured that the users would have an easier time navigating to the content in the documentation system, which makes the system more efficient for the users. Most users used the “Home” link in the header navigation, but there were also some that used the logo to navigate back to the *Dashboard* page.

Having several ways of navigating to pages in the web solution seemed to be useful because the users in the usability tests always chose different ways to navigate around the web solution. Some people showed a preference for navigating through the header navigation, while others would look for ways to navigate through the main section of the page.

Bookmarks

There were still a lot of questions surrounding the bookmarks, especially about the number of bookmarks that would be displayed, if they could choose which ones would be on the

Dashboard page, and about grouping or sorting the bookmarks. Doing some more research into bookmarking functionality could be useful for further development.

There was some criticism about the visibility of the unchecked bookmarking icons. During the development, we did check the contrast of the colors and found that they are compliant with current WCAG standards (see Chapter 4.3.4, and Figure 42 for results). Most users could easily recognize the bookmarking icon and had a good understanding of the different states of the bookmarking icon, as seen in Figure 26 on page 46.

Search functionality

The search field was overall well-liked among the test users, and everyone seemed to inherently understand how it would work. The only issue was that one user tried to click the enter-button on their keyboard to search for an article, which was not functional in the prototype. This was however the intended functionality and was implemented in the MVP of the web solution.

Other

There were a lot of comments regarding the "Newly read" articles module that was on the dashboard. Test users wanted to know how many articles would be displayed, and some had opinions about it being placed on the dashboard. There was talk of it being moved to a separate page or removing it completely. Since there has been an overweight of positive feedback on the functionality, I would not choose to remove it, but doing some more research into how it should be used and potentially making it an optional feature on the Dashboard page could be useful for future development.

3.4.3 HEURISTIC EVALUATION

Since there were not a sufficient number of prototype iterations that I could test the usability on, I wanted to make sure that I had not missed any major problems in the design.

To solve this problem, I asked one of the UX designers at the company if she would do a heuristic evaluation of our high-fidelity prototype. A heuristic evaluation is a technique in which usability experts assess user interfaces and report any issues they find (Interaction Design Foundation, no date-b). This can reveal insights into issues that go against common usability standards and give teams a chance to fix them before further design or development.

The UX designer followed the method described in "Design. Think. Make. Break. Repeat. The Handbook of Methods" (Tomitsch *et al.*, 2018, p. 74). She used the ten usability heuristics that was presented in Chapter 2.1.2 of this thesis as a checklist to evaluate the prototype. She then ranked each of the ten categories in the checklist by a severity rating that can be seen in Table 3 below.

Rating	*Description*
0	No usability problem
1	Cosmetic issue
2	Minor issue
3	Major issue
4	Usability catastrophe

Table 3 – Severity ratings used in heuristic evaluation

From this evaluation I learned that I had some minor issues with lack of keyboard functionality, the potential for having too many categories displayed on a page, and a high potential for confusion about the naming of the categories. This could cause confusion when users were trying to navigate to an article. The keyboard functionality was not available due to this being a prototype, and the category naming would be controlled by the company, so there was not much we could do about this feedback at this point.

However, we also got a score of "4 – Usability catastrophe" on the "Visibility of system status" heuristic. The indicators for which page is currently active in the header navigation were not clear, due to vague colors and there only being two links, which could make it difficult for the user to see which one was highlighted as being active. In the implemented web solution, this was fixed by correcting the color choices, and there were also more links present, which could help the users distinguish between the active and non-active links. See Figure 40 on page 69 for a comparison between the headers that contain the navigation.

3.5 Summary of chapter

In this chapter, I have explained the methods that were used to research, design and develop the project that is the focal point of this thesis.

I provided detailed information about the practical research methods, information architecture, design methods, and testing methods that were used, and presented preliminary results from my findings. The results provided a solid foundation for the design and development of the web solution.

The development of the web solution will be laid out in the next chapter, Chapter “4 Development.”

4 DEVELOPMENT

One of the requirements for this project was to develop a minimum viable product (MVP) of the final design. This chapter will explain how I decided which features to include in this MVP, the general organization of the development, how I chose which technologies to use, and how they were implemented into the solution.

Once the *Prototype* phase was finished and there was no time for creating more iterations, I decided that I needed to start the *Implementation* phase to make sure that there was enough time to develop our MVP within the available time. While the two team members from Mennt were conducting usability tests on the second iteration of the prototype, I developed the solution.

4.1 Feature prioritization

With the small amount of time that we had left on the project, I needed to prioritize what would be included in the development of the solution. I used the MoSCoW method, which is a method used to sort through all requirements in a project and figure out which ones should be prioritized (Agile Business Consortium, 2014). This method helped me categorize the features into four categories, which are explained in Table 4 below.

Category name	Description
Must Have	Every feature that was needed to make the system usable
Should Have	Features that would add a lot of value to the system, but they are not absolutely necessary for the system to function
Could Have	Features with a much smaller impact that could be great for further development
Won't Have (this time)	Features that will not be prioritized in the current development

Table 4 – MoSCoW prioritization categories

I placed all the features that had been included in the prototype, as well as some ideas that I had gotten from the first usability test and the ideas for future development onto the board. I decided that the "Must Have" category was our Minimum Viable Product (MVP), while the "Should Have" features were features that I would implement if there was time available when I had finished implementing the other features. Regardless, all the features would be thought of during development as something that should be possible to add to the project later on. The “Could Have” category included features that I wanted to incorporate, but that I felt needed more planning, design, or thought before implementing into the actual system. The “Won’t Have (this time)” category had the smallest number of features in it, which only included ideas that were not fully formed or would need a lot more planning before they could be implemented. The final prioritization matrix can be seen in Figure 36 below.

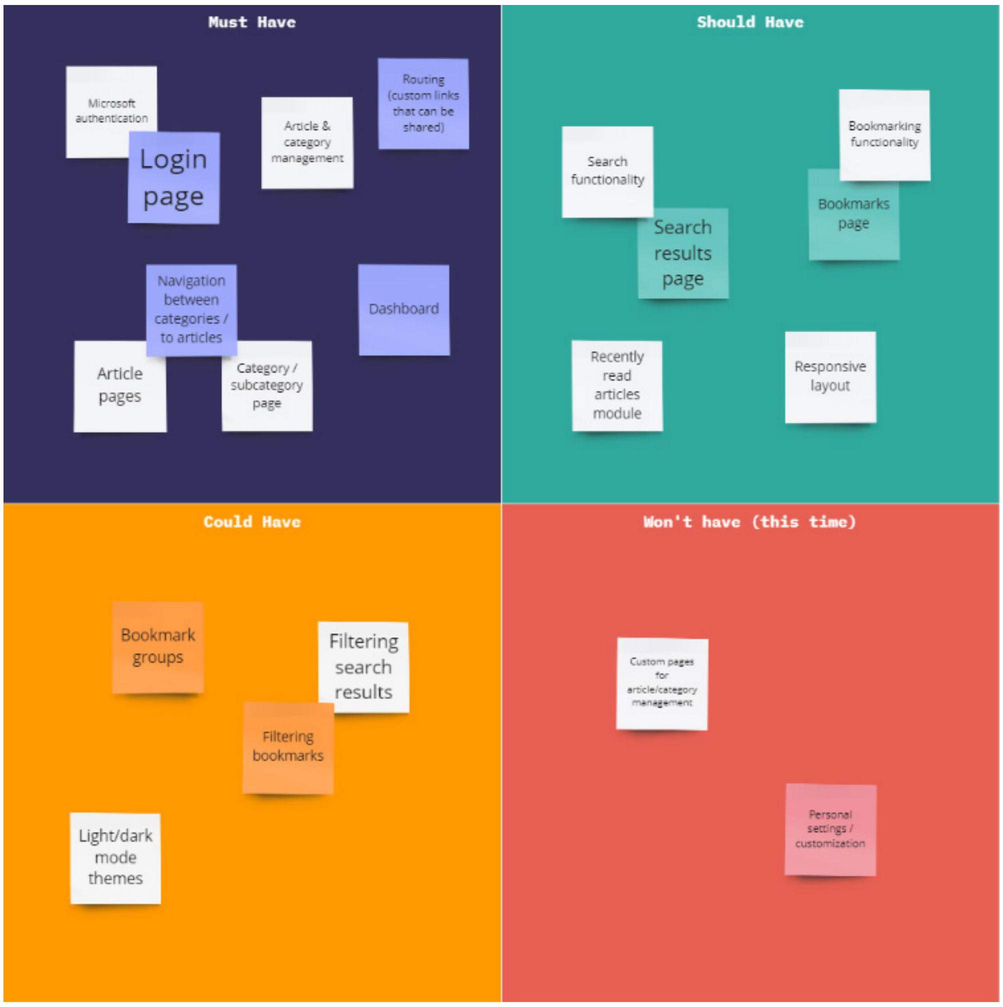

Must Have
Microsoft authentication
Login page
Article & category management
Routing (custom links that can be shared)
Navigation between categories / to articles
Dashboard
Article pages
Category / subcategory page
Should Have
Search functionality
Search results page
Bookmarking functionality
Bookmarks page
Recently read articles module
Responsive layout
Could Have
Bookmark groups
Filtering search results
Filtering bookmarks
Light/dark mode themes
Won't have (this time)
Custom pages for article/category management
Personal settings / customization

Figure 36 – MoSCoW prioritization matrix

4.2 General organization

After I knew which features to prioritize, I started the coding process. I used Git for version control and set up a GitHub repo with a GitHub Project that used a Kanban board to keep track of all tasks we needed to get done. I also used a branching system, where all new major features were created as a branch, and then merged with the main branch when finished. This way, I could work on different tasks at the same time without them interacting with each other and creating issues.

4.3 Frontend components & functionality

4.3.1 CHOOSING A FRAMEWORK

Before I started coding, I needed to choose a framework to build with. Most modern web frameworks are based on creating components and using them as building blocks in your development, so this worked well with me following component-driven development. I did not have time to wait for the results from the final usability test before I started development, but by using components throughout the project, it should be much easier to change anything in the future. If a change is needed, it can be done by customizing only a few lines of code, instead of changing the same code on every page of the solution. The thorough work I did in the design of the high-fidelity prototype, where I created components and categorized them by atoms, molecules, and so on, really made it quick and easy to develop most of the components, pages, and layouts in the final design.

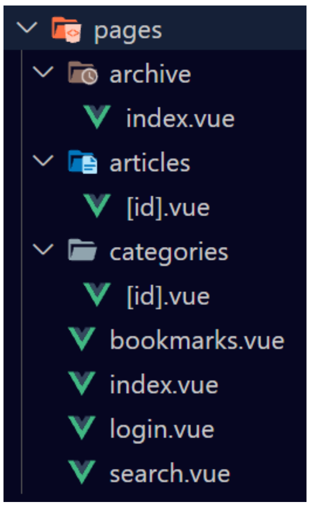
I looked at a few frameworks and ended up considering two frameworks that offer server-side rendering. One of these was Next.js, which is based on the React framework. This was the most obvious choice because I have the most experience working with React. The other framework I considered was the Vue framework equivalent to Next.js, called Nuxt. Ultimately, my decision was made easy by the fact that the developers at Mennt use Nuxt in their work, so building the solution with Nuxt would make it easier for them to further develop and expand on the project.

I built the frontend with the newest version of the framework, Nuxt 3. Nuxt has some features that were very useful to me during development, like automatic page routing, easy layout customization, automatic component imports, universal data fetching support, and easy state management.

4.3.2 AUTOMATIC ROUTING, LAYOUTS, AND PAGE SETUP

The automatic routing was especially useful, as it provided me with an easy way to create the pages of the system. The pages would then get automatically generated URLs, which can be shared between employees if they want to link to a specific article or category in the system. This was one of the features that I chose to be required in the MVP, and Nuxt made it really easy to set up.

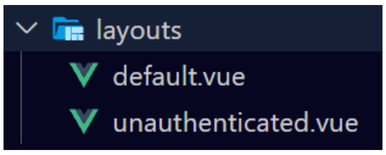
When creating a *.vue* file in the folder */pages* inside of the Nuxt project, Nuxt will automatically generate a route for that page. For the pages I created that were not meant to have any subpages, I created a single file with the name of that route. For the *Article* and *Category* pages, that would be related to a specific article or category from the database, I created a folder with the name of the route, and an *[id].vue* file inside of it. Inside these files, we could then access the id of the item from the parameter given in the route. E.g.: if the user navigates to the route */categories/coding*, Nuxt would know to go to the */categories* pages folder, and "coding" would be accessible as the "id" of the route, which in turn could be used by the page component to fetch information about the category that would be displayed on the page. The structure of the pages folder can be seen below in Figure 37.



```
pages
  archive
    index.vue
  articles
    [id].vue
  categories
    [id].vue
  bookmarks.vue
  index.vue
  login.vue
  search.vue
```

Figure 37 – Structure and content of the /pages folder

Nuxt also recognizes page layouts automatically if they are placed in the */layouts* folder. As explained in the chapter about the high-fidelity prototype, I had two different layouts in the design – one for authenticated users, and one for unauthenticated users that was used on the *Login* page. To implement this, I created two layout files in the repository (see Figure 38). The *default.vue* file will always be recognized by the framework as the default layout for all pages unless something else is specified, so I used this for our authenticated layout. The other layout was then specified as the layout for the *Login* page file.



```
layouts
  default.vue
  unauthenticated.vue
```

Figure 38 – Content of the /layouts folder

4.3.3 COMPONENTS

The components were all created as *.vue* files inside the */components* folder in the project. This way, Nuxt knew that they were components, and I didn't have to import them into every page or parent component when I wanted to use them.

Below in Table 5, you can see an overview of all the components that were developed. Most of these were identified when creating the prototype, but a few did not exist as separate components. These new components were *ArticleLink*, *Logout*, *MainCategories*, *NavSearchBarResults*, and *SkeletonPlaceholder*.

Name of component	Description	Sub-components
ArticleCard	Container that displays information about, and functions as a link to, an article	Bookmark Button
ArticleHeading	The heading used on article pages	Bookmark
ArticleLink	Link related to articles displayed on article pages	
Bookmark	Bookmarking icon that changes depending on bookmarked status	
Breadcrumbs	Breadcrumbs for the category and article pages	
Button	Button (animated)	
CategoryCard	Container that displays information about, and functions as a link to, a category	
DesktopNavBar	Header and navigation for desktop layouts	NavLink NavSearchBar Logout
Logout	Button that lets the user log out	
MainCategories	Module that displays the main categories from the database	CategoryCard SkeletonPlaceholder
MicrosoftLoginButton	Login button that adheres to Microsoft's design system	

ModuleHeading	Heading with icon used for modules	
NavLink	Internal links used for the main navigation	
NavSearchBar	The search bar that is displayed in the desktop header	NavSearchBarResults
NavSearchBarResults	Displays list of results from search in NavSearchBar	NavSearchBarResultsOption
NavSearchBarResultsOption	One of the results to be displayed in NavSearchBarResults	
PageHeading	Heading for single pages	
SearchField	Search field for the search page	
SkeletonPlaceholder	Skeleton loading placeholder for ArticleCard and CategoryCard components	

Table 5 – All developed components

In the prototype, *ArticleLink* and *NavLink* were variations of the same component. This worked fine for the prototype, but in the actual system they were very different, so I split them up to make sure that if I wanted to change something in one of them, it wouldn't have to affect the look or functionality of both. *NavSearchBarResults* was a part of the *NavSearchBar*, while the *NavSearchBarResultsOption* was its own component. I decided to split the results section into its own component since it included both a generated list of another component, as well as some other functionality. You can see a visual representation below in Figure 39 that explains how the component was broken up into smaller pieces.

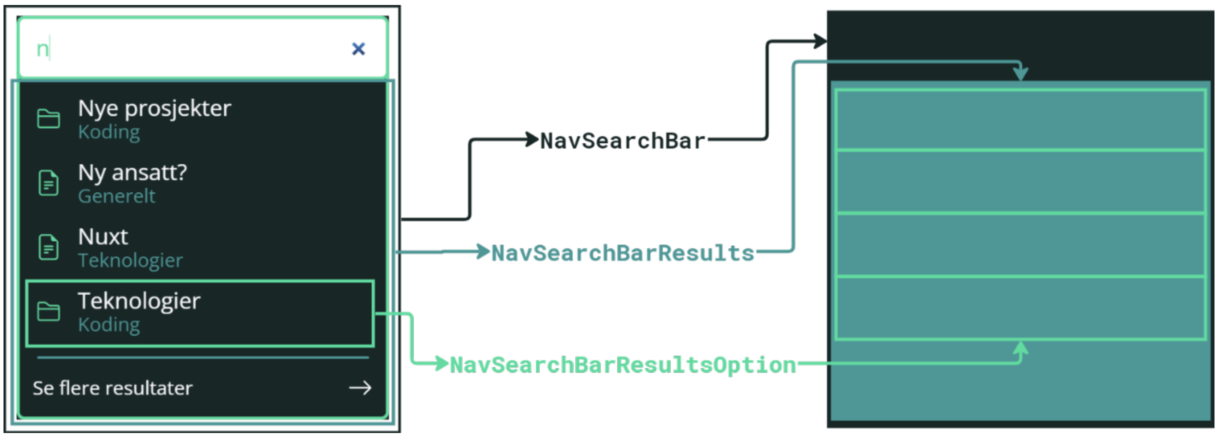


Figure 39 – Visual representation of how the NavSearchBar is composed

Logout, *MainCategories*, and *SkeletonPlaceholder* were all completely new components. The logout component replaced the two icons that were placed to the right of the search bar in the high-fidelity prototype navigation. The “help button” that was there before was meant to give basic information about the system, but I decided that it was taking up necessary space, and an article could be created in its place if the users needed this information. The profile icon that was there was not created due to there being no use for it at this stage of the system. It was meant to contain a dropdown menu that would lead the user to their personal settings, as well as a way to log out of the system. The *Logout* component was simply a button that would log the user out of the system, which was needed after authentication was set up. A comparison between the headers from the lo-fi prototype, the hi-fi prototype, and the finished solution can be seen in Figure 40.

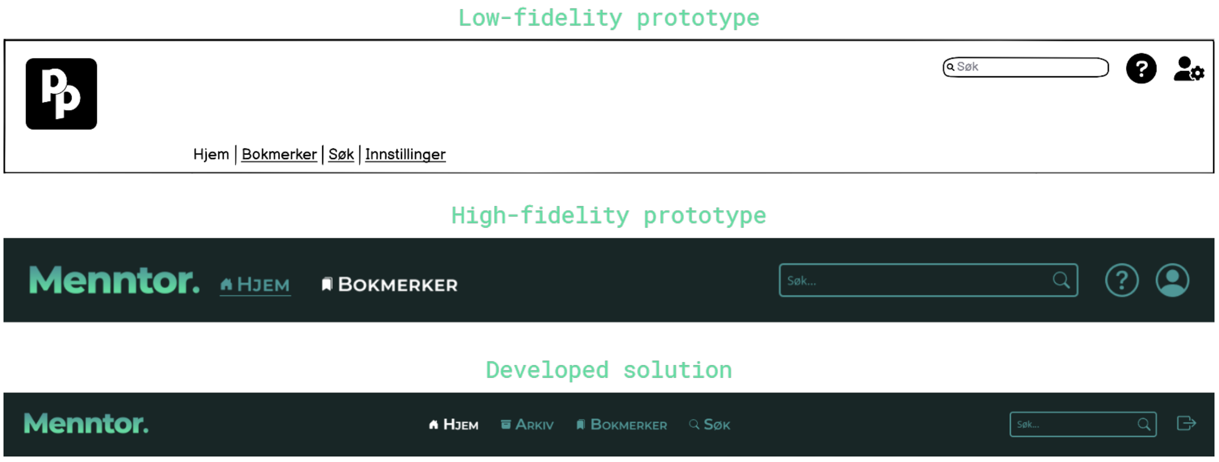


Figure 40 – Desktop header from lo-fi, hi-fi, and finished solution

MainCategories was created as a separate component that would fetch the main categories of the system, due to the way the category collection was set up in the database. I needed to filter out the main categories, and this was happening on more than one of the pages, so I saw the need for a component that would do this.

SkeletonPlaceholder was a component that I decided to add because the system would sometimes take a second or two to load all the categories or articles onto the page. We decided that for the user not to be confused about the missing components, having a skeleton loading component in place while the data was being fetched was a quick and easy solution to provide some feedback to the user. I added a package called *placeholder-loading* that contained pre-made skeleton loading style classes that were used to create a loading skeleton for the *ArticleCard* and the *ComponentCard*.

Apart from the aforementioned components, every other component was identified and created based on the high-fidelity prototype. However, as you could see in Figure 40 above, some of the components were changed slightly. I decided to add back the “Search” link to the navigation, which was there in the first prototype and then removed in the second due to it feeling like it was taking up unnecessary space. Since the search bar in the header was made less prominent in the developed solution, I decided to add it back as its own link again. This also made the most sense, as the search page also functioned as its own, separate page, unrelated to the search field in the header.

You can also see in the figure above that I added an additional link to the navigation that had not been there before – *Archive*. The archive page was added for two reasons. One of these was that I wanted to make it clear that the *Dashboard* page was its own page that showed content from other sections of the system, not the main page for the *Archive*. This had been a point of confusion for some users in the usability tests.

The other reason was that if the system was to be developed further, adding more functionality, there should be a page that contains the “archive” of the guide, i.e., the categories and the articles. I considered just calling it “articles”, but there had been some uncertainty around the word “article” from some of the users in our first usability test, so I decided to find a word that encompasses both the articles and the categories.

Another component that changed was the *Breadcrumbs* component. In the high-fidelity prototype, I designed the component with a dropdown-on-hover system, that would display the sibling categories of the category the user hovered over. I did not implement this in the web solution, as this would require the page to fetch and process more data than necessary every time the user changed pages, due to the way the system was set up.

There were some components that we left out of the development as well. The *Login* page had the biggest change due to some components being removed, as you can see in Figure 41 below. I decided to focus on only one of the ways to log in due to limited time, and the most important one was by far the Microsoft Azure login. This meant that the other login option on the page was rendered useless for the time being, so I did not spend any unnecessary time developing these components. More about this in the subchapter about Authentication later in this chapter.

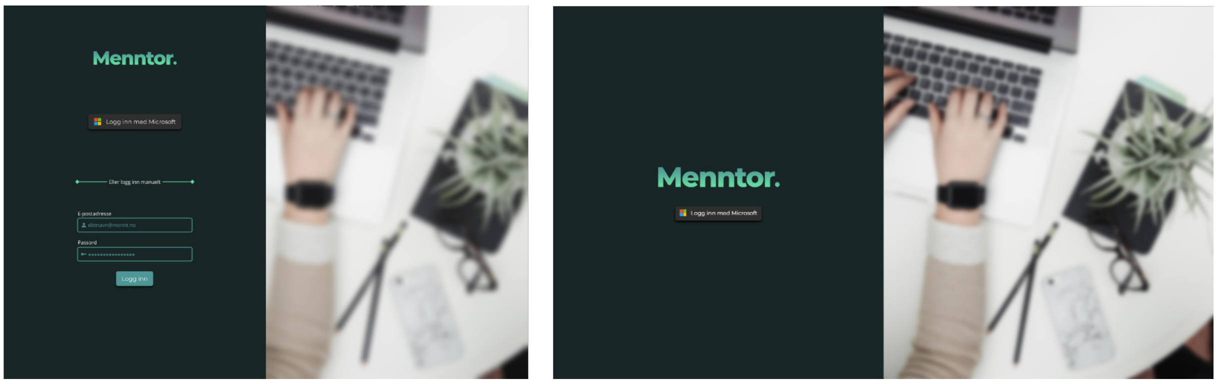
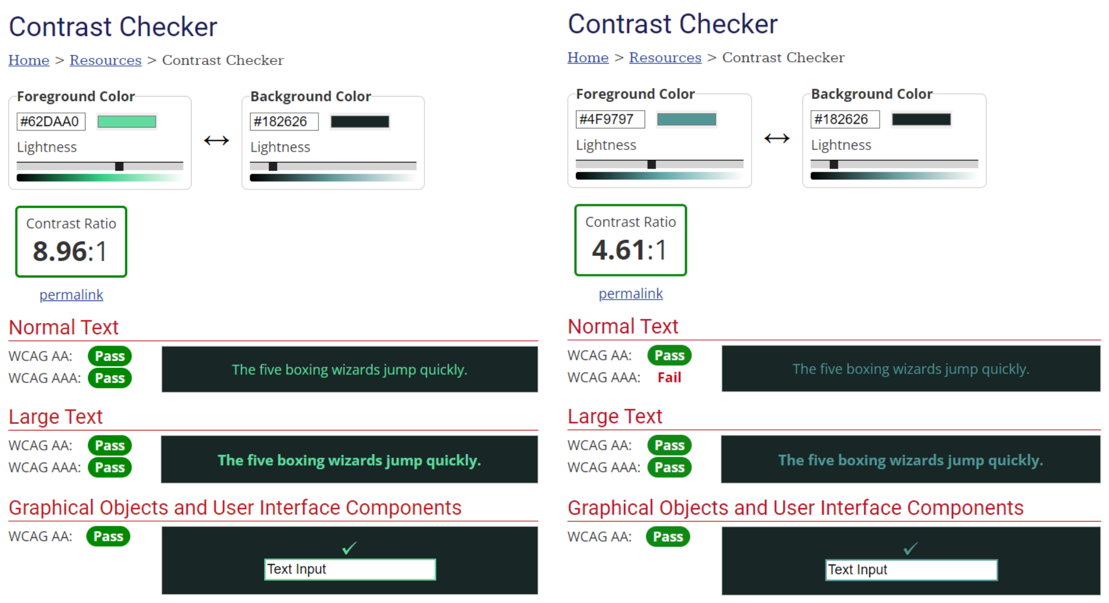


Figure 41 – Comparison of login pages between hi-fi prototype and finished solution

4.3.4 ACCESSIBILITY FEATURES

In the implementation of the web solution, I considered common ways to ensure that the pages are accessible to all users. I used semantic HTML wherever possible and made sure the headings on the pages were in the correct order. The page also naturally has very large font sizes due to my use of the Mennt design system, which can be good for users with low vision.

The color contrast checker developed by the "Institute for Disability Research, Policy, and Practice" was used when checking the contrast of the text and graphic elements on the web page (see Figure 42). The white and light green colors passed all tests against the background color, while the darker green color that was used for inactive links, buttons, and some graphic elements passed everything except the “Normal text – WCAG AAA”-test. With the way the colors were used, the color contrast on the page should be excellent.


Contrast Checker

Home > Resources > Contrast Checker

Foreground Color #62DAA0 Lightness ↔ Background Color #182626 Lightness

Contrast Ratio **8.96**:1

permalink

Normal Text

WCAG AA: Pass
WCAG AAA: Pass

The five boxing wizards jump quickly.

Large Text

WCAG AA: Pass
WCAG AAA: Pass

The five boxing wizards jump quickly.

Graphical Objects and User Interface Components

WCAG AA: Pass

Text Input

Contrast Checker

Home > Resources > Contrast Checker

Foreground Color #4F9797 Lightness ↔ Background Color #182626 Lightness

Contrast Ratio **4.61**:1

permalink

Normal Text

WCAG AA: Pass
WCAG AAA: Fail

The five boxing wizards jump quickly.

Large Text

WCAG AA: Pass
WCAG AAA: Pass

The five boxing wizards jump quickly.

Graphical Objects and User Interface Components

WCAG AA: Pass

Text Input

Figure 42 – Contrast checker results from WebAIM

Keyboard functionality was manually tested on all pages during development. An issue was discovered with the keyboard navigation in the navigation search field, where the user can’t navigate through the links in the search results without the search field closing the results and clearing the search field. This was due to the way I set up the functionality that displays the search results, and I did not have time to fix the bug before finishing up development. Due to there being no responsive layout implemented, there is also an issue where the users currently cannot zoom the page to 200% without the content being moved off the page or hard to read.

Except for these two cases that we are aware of, the website should be compliant with the universal design laws that are currently in place in Norway (IKT-løsninger, 2013; The Authority for Universal Design of ICT, no date-a).

4.3.5 STYLING

For the styling of the pages and the component, I wrote mostly custom SCSS. SCSS is a CSS extension language that gives us a few extra features. I mainly used it because of its ability to nest the styling code, as well as the import and variable functionality. Nuxt also supports SCSS with little to no configuration needed.

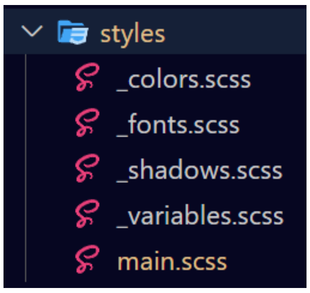


Figure 43 – Styles folder

There were several files relating to styling in the project. Three files in the */styles* folder (see Figure 43) contained variables, categorized by colors, fonts, and shadows. These files were then all imported into the *_variables.scss* file. Nuxt 3 uses Vite under the hood, and with a few lines of code in the Nuxt configuration file, I could tell Vite's CSS preprocessor to use this file as variables that would be accessible in every component file throughout the project.

As mentioned in the "High-fidelity prototyping" chapter, I chose to go with a darker design but wanted to ensure that every color and font could be easily changed later if necessary. By separating variables into different files and using the consistently in every component, it should be easy to make any changes in the project in the future.

For the main project styling, I had a separate *main.scss* file that contained general styling that was not related to specific components. For the component styling, I used scoped styles within each component to make sure that their styles did not affect other elements in the code.

To ensure cross-browser consistency, I also downloaded the CSS file *normalize.css* as a package and configured Nuxt to load this file, the *main.scss* file, and the *placeholder-loading* library CSS file globally into the system.

The icons used in the project were from the free, open-source icon library "Bootstrap Icons". These were available in Figma during the prototyping and were easy to copy over into the code when needed. By using SVG code, I could easily customize the icons in any way I needed. The icon library was selected because I knew that Mennt was planning to use Bootstrap as their styling library in future projects.

4.4 Data management

Setting up the technology that manages the data that is being used on a website can be time-consuming and difficult, which is why I chose to use a headless CMS to create, edit and delete the categories and articles in Menntor. A headless CMS is a content management system that exists separate from the application that it manages the content for.

I looked at four different providers for our CMS – Strapi, Decap CMS, Tina, and Sanity.io. Sanity.io did not offer a REST API, which meant that I would have to learn another new technology. With the time I had available, Sanity was not an option. Tina seemed nice for writing articles by using markdown but did not offer all the features that I needed. Decap CMS moved from Netlify to another agency partner while I was doing the research. I thought about the possibility of the new agency making big changes to the system in the near future and decided not to risk this.

I decided to use Strapi, which has a great user guide for both developers and content creators, a lot of customization options, and was very extendable. The Strapi graphic user interface is also very visually pleasing and intuitive and should be easy enough to use for whoever is going to manage the library. In addition to this, they offer image optimization, which saves on loading times and data sending. This is not only beneficial to user experience, in that the users won't have to wait as long for the images to load, but also good for sustainability, in that it limits the amount of data that is being fetched each time.

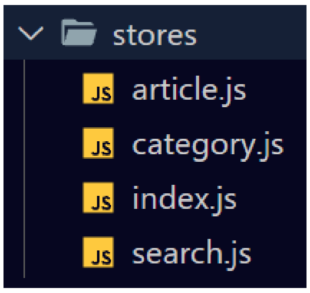
When thinking about further development, I looked at the options for creating custom pages inside of Menntor later that would talk to the Strapi CMS, where the employees could add, edit, and delete categories and articles. The research I did into Strapi showed that this should be entirely possible. Strapi also has a scalable database structure, with several options for SQL databases that can be exported into other databases if the company decides to move to another system later.

Strapi also has built-in support for adding alternative text and captions to images, which is great for accessibility and usability on the page, as well as the option to add i18n (internationalization) later on. If the company expands to other countries or gets employees that speak different languages than Norwegian, this could be beneficial in the future.

Strapi was set up in a folder inside the project, and being a headless CMS, it runs as a separate server from the Nuxt instance. I created content models inside the project, based on the database modeling I had done earlier in the project (see chapter 3.2.1 and Figure 19). These models can easily be changed by an administrator, and everyone given access to the system can create, edit, and delete content within the constraints of these content models.

4.4.1 PINIA

I needed a way to fetch the data from Strapi and use it throughout the web solution. To do this, I used Pinia. Pinia is a state management tool for Vue and Nuxt, referred to as a "store" because it stores the data. It allowed me to fetch and manage data, and to share the data across the components and pages in the application. I created files (see Figure 44) that functioned as "stores" for articles and categories that would talk to the Strapi server.

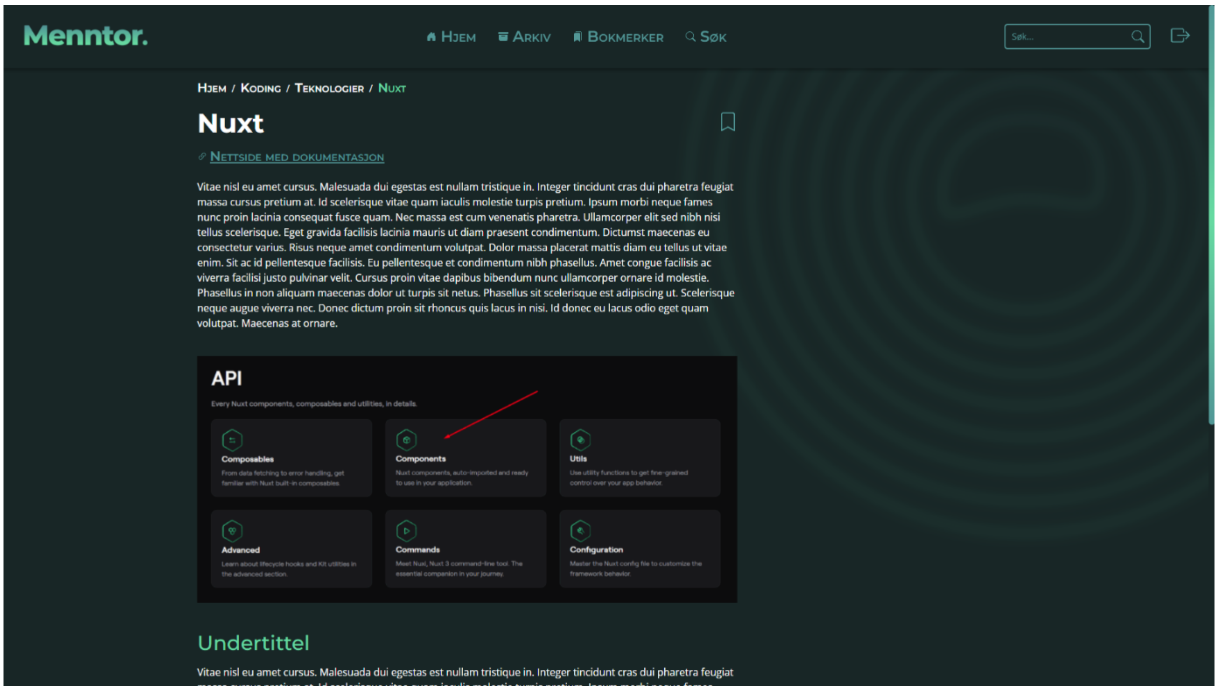


```
stores
    article.js
    category.js
    index.js
    search.js
```

Figure 44 – Stores folder

4.4.2 MARKDOWN PARSING

The articles that are written in Strapi will be sent to the front-end as markdown text. If this text was displayed directly on the page, it would include a lot of strange symbols, no formatting, and could confuse the reader. To parse the text, I included the package *markdown-it* as a dependency in the project. I then wrote it in as a plugin in the Nuxt project that could be used to parse the text inside of the *Article* pages. You can see how this looks on a page in Figure 45 below.



Hjem / Koding / Teknologier / Nuxt

Nuxt

Nettside med dokumentasjon

Vitae nisl eu amet cursus. Malesuada dui egestas est nullam tristique in. Integer tincidunt cras dui pharetra feugiat massa cursus pretium at. Id scelerisque vitae quam iaculis molestie turpis pretium. Ipsum morbi neque fames nunc proin lacinia consequat fusce quam. Nec massa est cum venenatis pharetra. Ullamcorper elit sed nibh nisi tellus scelerisque. Eget gravida facilisis lacinia mauris ut diam praesent condimentum. Dictumst maecenas eu consectetur varius. Risus neque amet condimentum volutpat. Dolor massa placerat mattis diam eu tellus ut vitae enim. Sit ac id pellentesque facilisis. Eu pellentesque et condimentum nibh phasellus. Amet congue facilisis ac viverra facilisi justo pulvinar velit. Cursus proin vitae dapibus bibendum nunc ullamcorper ornare id molestie. Phasellus in non aliquam maecenas dolor ut turpis sit netus. Phasellus sit scelerisque est adipiscing ut. Scelerisque neque augue viverra nec. Donec dictum proin sit rhoncus quis lacus in nisi. Id donec eu lacus odio eget quam volutpat. Maecenas at ornare.

API

Every Nuxt components, composables and utilities, in details.

Composables
From data fetching to error handling, get familiar with Nuxt built-in composables.

Components
Nuxt components, auto-imported and ready to use in your application.

Utils
Use utility functions to get fine-grained control over your app behavior.

Advanced
Learn about lifecycle hooks and Kit utilities in the advanced section.

Commands
Meet Nuxi, Nuxt 3 command-line tool. The essential companion in your journey.

Configuration
Master the Nuxt config file to customize the framework behavior.

Undertittel

Vitae nisl eu amet cursus. Malesuada dui egestas est nullam tristique in. Integer tincidunt cras dui pharetra feugiat

Figure 45 – Article page - "Nuxt", from the developed solution

4.5 Authentication

To authenticate the users, I had decided early on that they should be logged in via their work accounts, which were connected to Microsoft Azure. I asked the manager at Mennt to give me access to set up an “App” inside of their Azure Active Directory. In this app, I could limit the supported account types to “My organization only”, which means that only employees that have accounts that belong to the Mennt organization will be able to access this app and its contents.

To set this up inside the project, I used a package called *nuxt-auth* which is an open-source authentication module from Sidebase. This module is supported by Nuxt as one of its official modules and quickly allowed me to connect to Azure through the Open Authorization (OAuth) protocol by adding a simple configuration file to the project. The module then redirected any unauthenticated user to the login page, as well as gave me an easy way to log in the user and access the user data that was provided from Azure.

The data provided by the application was more limited than I had expected, only including the user's full name, e-mail address, and a profile image if it existed. This caused some issues for one of the key functionalities in the system – “bookmarking”. For a user to be able to bookmark an article, I would need some way to store the information in relation to that user. I was planning to use a unique ID of each user and store this information in the Strapi CMS, along with a relation to their bookmarked articles. I could have implemented this by using their e-mail as the unique ID, but the e-mail can be changed for the accounts in Azure, which would mean that if Mennt at some point wants to change their naming system for their e-mail addresses, the users would lose all their bookmarked articles. This would not be ideal and goes against the goal I set for future-proofing the system.

Since I did not have administrative access to the Azure Active Directory, I could not change the data that was sent to the system, but it is possible for Mennt to do this in the future. For this reason, I chose not to implement the bookmarking functionality, even though everything was made ready for the functionality to be implemented.

4.6 Search functionality

For the search functionality, I chose to not create a custom solution, but instead added a search engine that would run on another server. The reason I did this was that I wanted each change in the search menu to trigger an update in the results that were showing on the page, so I needed the search engine to be quick. Below in Figure 46, you can see what the search functionality looked like in action from the *Dashboard* and the *Search* pages.

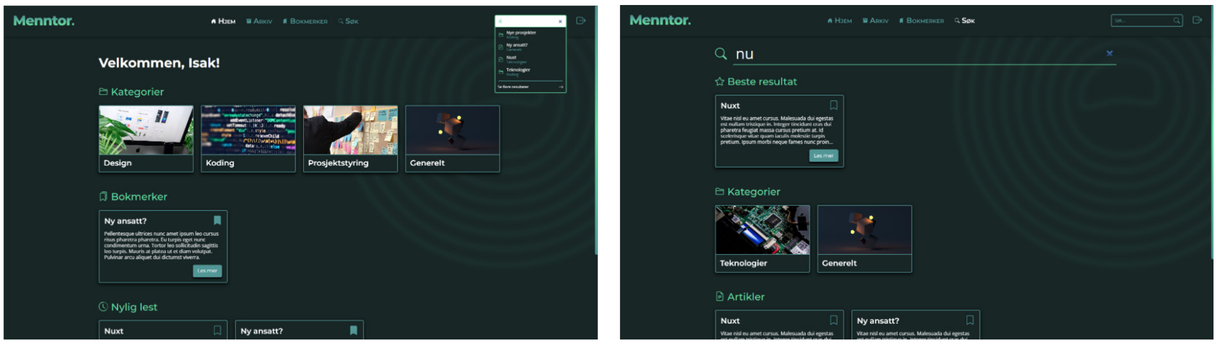


Figure 46 – Dashboard with search functionality (left) and search results page (right)

I ended up using Meilisearch as a search API because it fit the criteria well, and it already had a plugin on the Strapi Marketplace, which made it really easy to integrate. It allowed me to create a custom search collection based on Strapi collections that I wanted the users to be able to search through in an instant (Vermandel, 2023).

4.7 Summary of chapter

In this chapter, I have given an explanation of how the implementation of the web solution was carried out. The chapter started by describing how I decided which features would be prioritized in the development.

I have provided a thorough walkthrough of how the functionality of the web solution was set up, including the frontend components and functionality, the management of data for the content, how the users of the solution will be authenticated, and how the search functionality was implemented.

A presentation of the finalized web solution – the Menntor MVP – can be found in the next chapter, which contains a description of the results in this thesis.

5 RESULTS

In the previous chapters, I laid out the theory on which the thesis was based and explained which methods were used to empathize with the users and figure out a solution to the issue at hand. It was also explained how the project group and I designed and developed Menntor through the use of information architecture design, prototyping, usability testing, and, finally, the implementation of Menntor. This chapter will present and explain the most important results that were obtained from using all of these methods pertaining to my research questions in this thesis.

5.1 The design of "Menntor"

Together with the project team, I designed a solution that we think will solve a lot of the problems that Mennt is facing. This solution was "Menntor", an internal documentation system where the company can add, edit, and delete articles related to their work. The employees can then look up these articles in the system by either navigating through the system's categories or by using the search functionality.

5.1.1 SOLVING THE CORRECT PROBLEM

To arrive at this solution, I looked at all the data that was collected through the research that was done by the use of practical research methods. This includes the interviews, the questionnaires, and the competitor analysis. From the affinity diagramming and questionnaire analysis I could see that Mennt had a lot of issues relating to their timekeeping system, which was the main focus at the time, but there also seemed to be some challenges with organizing the information regarding the other software systems they were using. However, they stated that they were not dissatisfied with their current system.

I decided to change the course of the project after the *Empathize* and *Define* phases had been completed because I learned that the employees of Mennt wanted a range of different features and options from a timekeeping system that would be impossible for us to achieve in a short amount of time. I could either try to design and implement a timekeeping system that would be extremely limited due to the scope of the project, or I could pivot and try to create something more useful. The alternatives that were considered have been described in Chapter 3.1.2.3.

In the competitor analysis I conducted, the most interesting result was that I found a total of 106 listings that matched our criteria with only a quick search. This showed me that there is a lot of competition that has made similar software. One of these could likely be a better fit for Mennt AS and their employees than what a project team consisting of three people could design and develop in four to five months. This is ultimately what made me decide against the timekeeping solution plans. I could not see a way to justify spending a lot of time developing a timekeeping solution that would be less usable than something that already exists on the market today.

The second research question in this thesis asked what measures can be taken to make the web solution sustainable. Creating a system that is likely not going to be used would go against basic sustainability principles since hosting another timekeeping web solution would mean that more data than necessary would be stored.

After I decided to take the project in a different direction, there was not enough time to do extensive research into the potential solutions the team had come up with, but I had gotten some useful information from the research that helped guide our choice of solution.

5.1.2 FOCUS ON THE USERS & THE SOLUTION

I always made sure to include users outside of the user group when making design decisions, because I wanted the design to be universal. Regarding our target group, I found that the current employees had a mean age of 30, with a mix of gender identities. Mennt AS employs people with various backgrounds and roles. They currently employ web developers, designers, and marketing professionals, but could employ people from other professions in the future as they expand.

The research showed that the employees at Mennt wanted a new system that would:

- have a modern, visually pleasing user interface,
- be minimalistic, helpful, and easy to use,
- help them remember details about their work,
- help them collaborate as a team,
- have the Norwegian language as an option,
- and keep everything in one place.

"Better documentation" had also been a topic that showed up in both the data from the interviews and from the questionnaires. With all this in mind, I ended up with the idea that Mennt could benefit from a system where they could store all the information that the employees would need to guide them in their workday, sort of like an employee handbook.

Employee handbooks often contain information like HR information, employee rights, and company policies. Since the system would be created in a way where Mennt would have total freedom over what information they would store in it, I decided to not name the system an "employee handbook", but instead an "internal documentation system". My goal was to assist them in resolving their challenge of managing a large number of internal software systems, so this label was better aligned with the system's intended purpose.

5.2 Implementation of web solution

The main result of this thesis is "the Menntor MVP", a web solution that implements an internal documentation system for Mennt AS.

This web solution addresses the problems that Mennt had with their growing number of internal software systems by giving them a place to store documentation related to these systems that the employees can use when they need guidance. The research and design described in the Methods chapter provided the necessary guidance for the implementation fo the web solution.

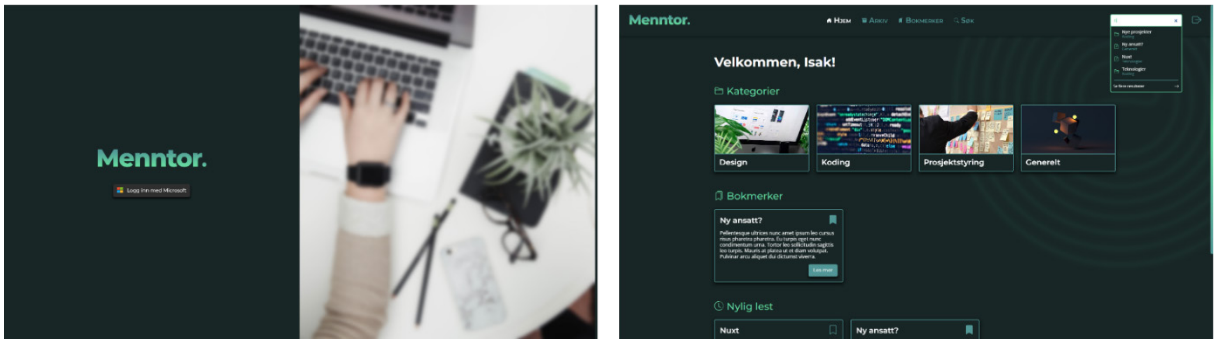


Figure 47 – Screenshots from the web solution (Login and Dashboard pages)

At the end of the project, the main results of the implementation of the web solution were:

- A modern design with a focus on usability
- Universal design with current and future employees in mind
- More features were implemented than necessary for the MVP
- Sustainable design features

5.2.1 USABILITY AND UNIVERSAL DESIGN

In the final implementation, Strapi CMS was added, which gives the company full control over the naming of these categories and articles so they can change this to fit their own needs. This was a result of the research from the usability tests which showed that the labeling of the categories, which was guided by the content hierarchy I created, could still be confusing to some users.

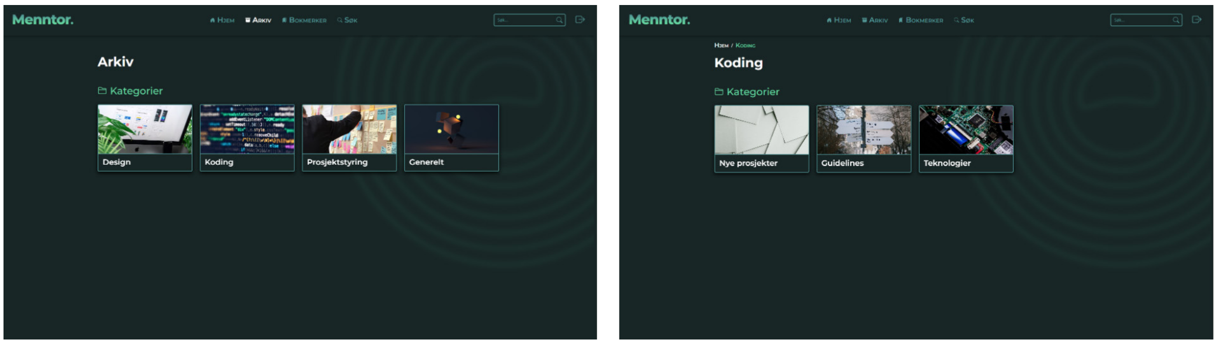


Figure 48 – Screenshots from the web solution (Archive and Category pages)

By reducing the amount of content that was placed on each page and focusing on testing the navigation design in the usability tests, I did my best to minimize the user's cognitive load and make the system as user-friendly and efficient as possible. The final visual design and page architecture was based on the prototypes I created, guided by the information architecture design, and improved by usability testing, as detailed in Chapter 3.3.2. Using a component-driven design system also ensured that web pages look coherent, which makes them more recognizable to the users.

With universal design in mind, the team tested the usability on users both in the user group and on other potential users to ensure that everyone is included in the design. I also made sure to include common accessibility features, as explained in Chapter 4.3.4. As mentioned, there were only a couple of features missing for the system to be WCAG-compliant.

5.2.2 FEATURES IMPLEMENTED

In Chapter 4.1, I explained how I chose which features to prioritize in the *Development* phase. All the features from the “Must have”-category were implemented in the web solution. I also had time to implement three of the features from the “Should have” category – “Search functionality”, “Search page”, and “Bookmarks page” (see Figure 36).

The system could work just fine without the search functionality since there are other ways to navigate to the information, but I chose to prioritize implementing this functionality because it is an important way for the users to be able to find the information they need in an efficient way.

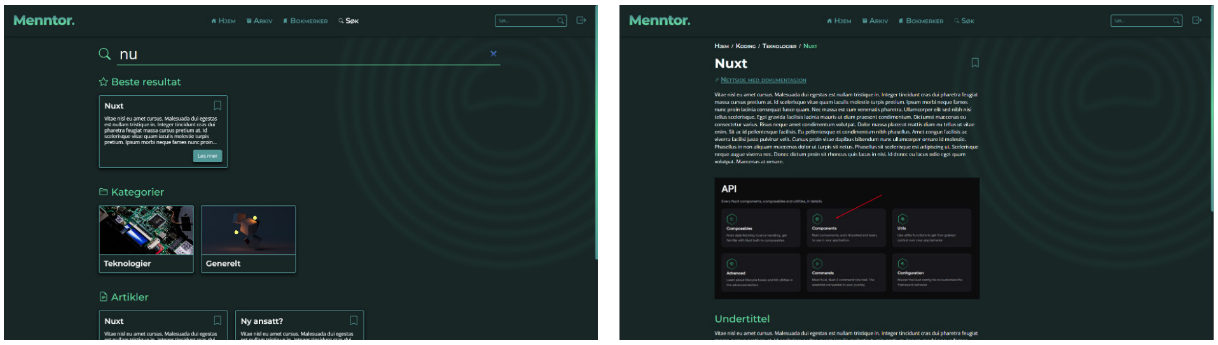


Figure 49 – Screenshots from the web solution (Search and Article pages)

Even though the bookmarking functionality was not implemented, as explained in chapter 4.5, the bookmarking page, buttons, and modules were created so that everything is ready for the functionality to be implemented. The bookmarking feature was a key feature in the solution, even though it’s fully functional without it. It should be quick and easy for Mennt to add the connection between the users and the bookmarked articles later on.

5.2.3 SUSTAINABILITY

To ensure sustainability in the web solution, there were two areas that I focused on. One of these was future-proofing the system, and the other was to reduce the amount of data that would be stored on the server.

Using universal design principles was not only done for the usability of the systems, but to ensure that the system would be usable for not only the current but also future, employees. To make sure that the system would last for as long as possible, I also chose technologies that were stable and considered “mature”, so that they would not cause any problems in the near future. When choosing the database for the CMS, I made sure that it could be migrated into another system later if needed. These considerations should improve the longevity of the system.

I did my best to reduce the amount of data that was on the page, and that was fetched between the frontend and the content management system. By saving the data fetched from the CMS inside a “store” on the frontend, and then using that store to mutate the data and share the data with components, I limited the number of times the client would need to fetch the data from the CMS.

Using component-driven development also reduces the amount of total code needed to be saved on the server, and by connecting to Mennt’s Azure registry for authentication, there was no need to store any duplicate data in the Menntor CMS. The majority of the styling code was created inside a component and scoped so that it would only load when the component was loaded to the page. This way, the server would have to send less data to the client for each page load.

The web solution was not deployed, because the company will take care of hosting the system. The research that was done into cloud hosting, where I found which major servers are the least carbon-intensive (see Chapter 2.2.2.1), will be delivered to the company as a recommendation for how to host the system in a way that uses the least amount of natural resources. High-quality screenshots from every page in the implemented solution can be seen in Appendix I.

5.3 Summary of chapter

In this chapter, I have presented the results of this thesis. The two main results from the project that the thesis is focused on are the solution, or concept, that the project team and I came up with, as well as the minimal viable product that was developed from this solution.

Related to the solution, I have talked about how we reached the right solution for this project by following the Design Thinking framework that focuses on finding the correct issue at hand before coming up with solutions. This framework also puts the user at the center of all design decisions, so the user needs that we found in the project were also presented in this chapter.

In the section about the Menntor MVP, I detailed which features were implemented, how universal design principles were included in the development process, and how the web solution was made as sustainable as possible.

In the next chapter, “Discussion”, I will go through all the research questions and their refinements and discuss my findings in this thesis. Finally, I will provide my conclusions on these questions, as well as a final conclusion related to the main problem statement in the thesis.

6 DISCUSSION

The main problem statement in this thesis was "*How can the issues that Mennt is having due to a growing number of internal software systems be solved through the development of a web solution?*" I have looked at principles for user-centered design to guide me in finding the correct solution to their issues. Two accompanying research questions were posed to account for the usability, efficiency, and sustainability of the web solution.

6.1 First research question - Usability and efficiency

Our first research question was "*How can the web solution be designed in a way that makes it user-friendly and ensures efficiency for the user?*" In chapter 2.1.6 I presented a list of questions that were asked to help me answer the research question. I will now go through these one by one and discuss my findings.

6.1.1 WHAT ARE THE SPECIFIC USER NEEDS AND PREFERENCES THAT THE WEB SOLUTION SHOULD ADDRESS?

By using the Design Thinking framework, explained in Chapter 2.1.1, I followed principles of design that focus on putting the user's needs first and finding the correct problem before putting forward a solution. The first step of the framework is to empathize and develop an understanding of the users that you are designing for. In this project, I did this my using the practical research methods that are detailed in Chapter 3.1.

After having developed a deeper understanding of the project, I found that the needs of the company would likely not be solved by creating another timekeeping solution. Looking at the research I had available, Mennt's requirements for a new timekeeping system would be impossible for this project team to create within the project's timeframe and could potentially be solved by existing solutions. The research also showed that a potential solution could be an e-mail plugin, but this would need to be developed across several platforms, which was also not within the scope of this project. I decided that the best solution available was to create a custom, internal documentation system for Mennt, to improve the access to good, useful documentation for the existing systems.

Ideally, I believe that I should have repeated the *Empathize* and *Define* phases when deciding on a new direction for the project, to better understand the specific user needs relating to a documentation system. The time limitation pushed me to move on with the project, but the user needs that related to both solutions were identified from the research that was done and can be seen in the list in Chapter 5.1.2.

So, were the needs met through the solution that I came up with? From the research, I saw that the issues stem from some of the users lacking an understanding of how the current system functions. This problem can be solved in more than one way. The most obvious one, that I was focused on from the start, was to replace the current system with a new one. This would be too comprehensive for a project of this size and could also create more problems for the company if the new system was not adequate. I would not say that all the related issues can be fixed by the solution that was created, but it should at least be an improvement.

6.1.2 HOW CAN I ENSURE THAT THE WEB SOLUTION IS ACCESSIBLE AND INCLUSIVE FOR USERS WITH DIFFERENT ABILITIES AND BACKGROUNDS?

In the Theory chapter, I talked about accessibility and universal design principles, and how they differ from each other. Throughout this project, I have used principles for universal design to guide the research and design choices that I made. Whenever research was done, the current and *future* employees of Mennt were included. Since I have no way of knowing who the future employees of Mennt could be, I included control groups in the research to see what people outside of the company would think about the same questions and suggestions.

In this project, I have tried to follow the laws and regulations that are in place in Norway, related to universal design (IKT-løsninger, 2013). The laws differ based on whether the company that owns the web solution is private or publicly owned, but they demand that a subset of the Web Content Accessibility Guidelines are followed (The Authority for Universal Design of ICT, no date-b). Norway also has workplace regulations in place that state that "permanent workplaces shall be designed, dimensioned and furnished to accommodate employees with disabilities so that they are able to work in the undertaking (The Workplace Regulations, 2013)."

In Chapter 4.3.4, I have detailed the accessibility features that were included in the implementation of the Menntor MVP. By following the rules and regulations in place, the implemented solution should be accessible to most people. There are, however, still some limitations in the implemented solution. In the chapter previously mentioned, I explained that I had found two issues that break with the regulations that I did not have time to fix – a keyboard issue with the search bar, and not being able to zoom the page to 200% text size.

Not being able to zoom to 200% text size is due to a larger limitation with the project, which is the lack of mobile and tablet layouts. Not having layouts for mobile and tablet devices makes the solution less accessible by default. A study mentioned in the book "Mobile First" mentioned that 64% of the participants used their smartphones at work (Wroblewski, 2011, p. 25). Most people nowadays always have their phones nearby, so not being able to check the Menntor guide through a phone could be a major nuisance.

6.1.3 HOW CAN WE STREAMLINE THE USER EXPERIENCE AND MINIMIZE COGNITIVE LOAD THROUGH INTUITIVE NAVIGATION, CLEAR LABELING, AND EFFECTIVE USE OF VISUAL DESIGN?

To make the navigation as intuitive as possible, I started by using principles of information architecture to lay the foundation for how the users would navigate through the web solution. Several ways to navigate through the web solution were included – menu links in the header, search, and breadcrumbs on the pages that were several levels deep.

The navigation was also a huge focal point in all the usability tests, to ensure that every navigation option would be understandable and efficient for the users. There were changes made based on user feedback that should make the navigation more understandable. In the final usability test, not a single user had issues with the clickable navigation or the search navigation.

As mentioned in Chapter 5.2.1, the naming of the categories still confused some users. I don't see this as a major issue, because the system will be delivered to Mennt without any content, and they can choose the names of all categories and articles themselves. This way, they can choose category names that make the most sense for the people that work there and adjust them as they go.

The feedback from the users in the final usability test that I would consider critical to the user experience was noticed, and fixed, during development. The rest of the feedback that was received was mostly related to the user's personal preferences and should not impact the user experience dramatically.

Using component-driven design in the prototyping and final implementation of the web solution helped me ensure that the visual design and functionality were consistent across all pages. I would consider this to be a more effective way to design web solutions for the developer, as well as a good way to minimize the cognitive load for the users that expect consistency.

6.1.4 HOW CAN WE TEST AND EVALUATE THE USER-FRIENDLINESS AND EFFICIENCY OF THE WEB SOLUTION, AND MAKE ITERATIVE IMPROVEMENTS BASED ON USER FEEDBACK?

To test and evaluate the usability of a web solution, several methods can be employed. In the context of the Design Thinking framework, the *Prototype* and *Test* phases aim to create iterations of your idea that can be tested on real users to gather feedback and make improvements.

In this project, I have made two prototype iterations of the web solution – a low-fidelity prototype and a high-fidelity prototype. To test and evaluate their user-friendliness and efficiency, I performed usability tests on five users between iterations. The idea behind using only five users was explained in Chapter 3.4.1.1. This method is great to use when quickly making several iterations of prototypes, to discover most of the flaws before making improvements and testing again. On the one hand, I did discover a lot of major issues in the first test and the results from the second usability test showed major improvement. On the other hand, I could have discovered more usability issues if I had time to create more iterations of the prototypes.

To test the final prototype, I also asked an expert to evaluate the design. Most of the issues from the Heuristic Evaluation (see Chapter 3.4.3) were fixed in development before I received the results of the evaluation. I caught some mistakes during development that seemed obvious to fix, but these improvements were never tested on real users to get proper feedback. The second usability test also included feedback that correlated with the issues I had changed, but that does not necessarily mean that my implemented fix was the correct one. If any major usability flaws are discovered in the future, they should at least be easy to fix since I have used component-driven design across the solution.

6.1.5 SUMMARY & CONCLUSION TO THE FIRST RESEARCH QUESTION

The research question posed was "*How can the web solution be designed in a way that makes it user-friendly and ensures efficiency for the user?*" In this section, I have looked at some sub-questions that were created to help me answer this question.

The first sub-question – "*What are the specific user needs and preferences that the web solution should address?*" – showed me that to design a solution that fixes the user needs, you should not only focus on fixing the solution but also figure out what the root cause of the issue is. This was done by following the phases in the Design Thinking framework from start to finish.

The second sub-question asked, "*How can I ensure that the web solution is accessible and inclusive for users with different abilities and backgrounds?*" For the web solution to be user-friendly to *all* users, the design process needs to include all of these users. This is what the concept of universal design aims to solve. The best way I could find to follow universal design principles in this project was by including current and future user base in all ideas, including control groups when doing research, and following accessibility guidelines created by experts in the field.

"*How can I streamline the user experience and minimize cognitive load through intuitive navigation, clear labeling, and effective use of visual design?*" was the third sub-question that was asked. The navigation and labeling of the system and the content within the system were created through the use of information architecture design. The usability tests showed that there were still improvements to be made in both the way the users navigated through the pages to the content and in the names that were chosen for the content categories.

Having a good baseline to work off of – the sitemap, the content architecture, and the database architecture – was very useful to guide the design process, but feedback from real users was needed to make the user experience an efficient one.

To create an effective visual design, component-driven design methodology was also used in both the prototyping and the development of the solution.

The fourth and final sub-question – "*How can I test and evaluate the user-friendliness and efficiency of the web solution, and make iterative improvements based on user feedback?*" – was answered by the methods that I used in the project. Iterative improvements were made by prototyping the solution and testing the functionality with the help of real users. To ensure that nothing was missed by the users, an expert was also consulted that analyzed the solution against common usability guidelines.

At the root of everything that was done during the project was the Design Thinking framework (see Chapter 2.1.1). The framework helped by offering a user-centered approach to the design process. It helped by starting the process focused on finding the correct problem instead of jumping straight into a solution and offered a structure in its six phases – *Empathize*, *Define*, *Ideate*, *Prototype*, *Test*, and *Implement*. Although the web solution can always be improved, the combined use of the methods I chose to use within this structure helped design the web solution in a way that was focused on creating a user-friendly and efficient experience for the users.

6.2 Second research question - Sustainability

The second research question I posed was "*What measures can be taken to make the web solution sustainable?*" A few sub-questions were stated in Chapter 2.2.3 that will be addressed in this section.

6.2.1 WHAT ARE THE KEY ENVIRONMENTAL IMPACTS AND CHALLENGES ASSOCIATED WITH WEB DEVELOPMENT?

Through the research I did into sustainability related to web development, detailed in Chapter 2.2, I found that the key impact is the amount of resources being used. Data centers across the world consume large amounts of energy and could be responsible for as much as 3.9% of global greenhouse gas emissions (Freitag *et al.*, 2021).

The use of natural resources that are involved in building the data centers, as well as hardware for servers and anything that consumes the data sent over the internet, are also making an environmental impact that should be considered.

6.2.2 HOW CAN WE ENSURE THAT THE WEB SOLUTION USES ENERGY-EFFICIENT TECHNOLOGIES AND INFRASTRUCTURE, AND AVOIDS UNNECESSARY RESOURCE CONSUMPTION?

To ensure that the web solution avoids unnecessary resource consumption, it should be planned for in the design. Small choices can make an impact, like including dark mode in the design. More TV and computer screens now come with OLED displays, where the screen will consume less energy when the image is darker due to the way the light in the screen comes from each individual pixel instead of a shared backlight, like in LED and LCD displays (Sengupta, 2022).

Using dark mode is a choice that only saves energy in some very specific cases. The biggest thing we can do during the design and development phases is to take steps to reduce the amount of data that will be stored and transferred across the web. We can do this when designing the databases when choosing what content to include on a page, and when deciding on how we manage the data within the developed application. Every step of the design process needs to be involved to make the greatest impact.

Sometimes the best solution to a problem can be to not develop something, as I saw in the case of the timekeeping solution that was the focus at the beginning of my project. By not building something that already exists, I avoided putting more data into the world. However, if I had not come up with an alternative solution, this would require the employees to figure out how to use the systems the same way they have done up until now. They could either go through lots of online documentation separately, which will consume data from sources where they have no control over whether the hosting is "green", or they could ask each other for advice. In my research, I found that the employees sometimes work from different locations, and they often hold meetings and call each other on Microsoft Teams. Streaming video uses a massive amount of data, which quickly becomes more wasteful than storing a web solution that is not being used.

In Chapter 6.1.1, I explained how we considered the needs of the users when designing a solution. The needs of the company were also considered in the solution, by not creating a system that is highly integrated with their current solution, but instead creating a flexible documentation system that can be useful regardless of what software solutions they decide to use. Software is usually not static and will be updated or replaced by the company over time. This way, the software also remains usable over time.

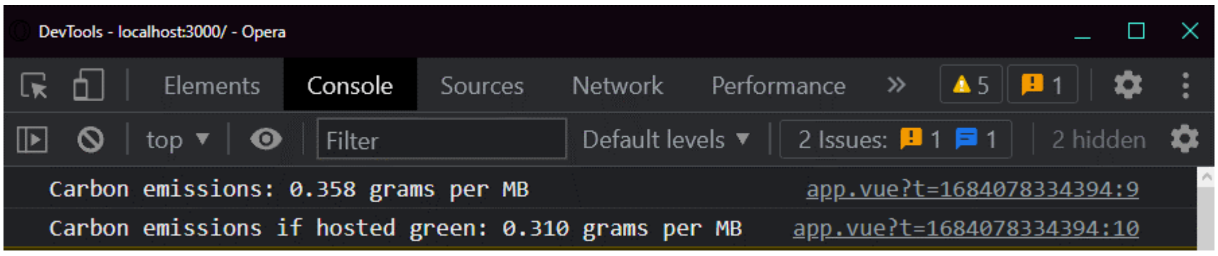
As I explained in the Theory chapter in subchapter 2.2.2.1, data centers are a huge part of the energy drain related to the web. As web developers, we can ensure that our web solutions are hosted in the most sustainable way by researching which data centers are the "greenest" and selecting those over other, potentially cheaper, options.

6.2.3 HOW CAN WE MEASURE AND REPORT ON THE SUSTAINABILITY OF THE WEB SOLUTION?

Through my research, I have found that there are several ways to measure the sustainability of a web solution. Tools like Website Carbon Calculator and Ecograder are created to measure and analyze the sustainability of hosted web solutions by simply pasting in the page URL (Wholegrain Digital, no date; Mightybytes, no date). This is a quick and easy way to get feedback on the state of sustainability in web solutions, which can help catch major resource drains that could potentially be fixed by making changes in the next update of the solution.

Since my project was not going to be hosted, I could not use these tools to measure the sustainability of my project. I used CO2.js, a tool developed by The Green Web Foundation, to measure the carbon emissions per byte sent by the web solution. Every time the application is refreshed, the result will be logged to the console. The final result at the end of the development was 0.358 grams of carbon produced per megabyte (see Figure 50).

The tool also has the option to estimate the result if the hosting environment is a "green host". I included this result in the console to show the difference, and the final result with green hosting would be 0.310 carbon produced per megabyte.



```
Carbon emissions: 0.358 grams per MB                    app.vue?t=1684078334394:9
Carbon emissions if hosted green: 0.310 grams per MB    app.vue?t=1684078334394:10
```

Figure 50 – Console logging the carbon emissions per byte

For the initial page load of the Dashboard, without caching, the developer tool in the browser showed me that 3.9 MB of resources were transferred. On the second page load, with caching, only 1.3 MB was transferred. In the table below, I have presented the results of these tests.

	Page result	First page load	Cached page
Without green hosting	0.358 grams per MB	1.3962 grams of carbon emitted	0.4654 grams of carbon emitted
With green hosting	0.310 grams per MB	1.209g grams of carbon emitted	0.403 grams of carbon emitted

Table 6 – Carbon emission results

In conclusion, there are several ways to measure the sustainability of a web solution. It is complicated, and all the methods presented here are estimates. Getting an accurate reading would be nearly impossible due to all the different factors that affect the environment. However, estimates can still help us discover major problems and take steps in reducing the impact of our web solutions.

6.2.4 CONCLUSION TO THE SECOND RESEARCH QUESTION

The second research question was "*What measures can be taken to make the web solution sustainable?*" In the previous sub-sections, I have discussed the key environmental impacts of web development and how we, as web developers, can work to reduce the impact of our web solutions. Considering that the IT industry is responsible for approximately the same amount of greenhouse gas emissions as the airline industry, sustainable design should be considered in everything we create (Henriksen, 2023).

By including sustainability in our planning, design, and development, we can reduce the impact on the environment by reducing the amount of data that is stored and fetched to our solutions. It is important that we also measure the impact of our solutions, to get an overview of how our solutions will affect the world, and to catch and improve any major resource drains. This is not only good for the environment, but also for the people using them who will see faster download speeds and less unnecessary clutter on the web pages.

6.3 Further development

The Menntor MVP is functional, but there is still work to be done to improve the solution. All the "Must have" features in the prioritization matrix were implemented (see Chapter 4.1, Figure 36), but the features that were not should be prioritized for any further development. This is especially true for the bookmarking functionality, which is visually implemented on the page but lacks functionality. The visual aspects of this being ready could potentially confuse users that are new to the system. The other functionality that should be highly prioritized is the responsive layout, which will make the web solution more accessible.

The feedback that was received from the users showed that they have a lot of ideas for further development. Filtering and sorting of the articles in bookmarks, category pages, and search results, should be done if the company ends up having a lot of articles saved in the system. There was also talk of linking specific articles to current projects that the company is working on, as a way to easily find information related to the projects.

Having a custom system means that the company can develop and add to the system in the future however they would like. The Dashboard page can be customized if the company wishes to add any more functionality to the system. The project could even be built into an intranet for the company, by adding integrations with their tasks in Azure DevOps, important internal messages, or their calendar systems.

There was a lot of talk about the integration of systems they are currently using into our solution, but since the company is in a starting phase where they are frequently changing out their software, I decided that it would not be wise to spend time on this until Mennt has become more established. However, the company is free to develop the solution in any way they wish, and the system is expandable and should be ready for further development.

6.4 Final conclusion

The main problem statement in this thesis was "*How can the issues that Mennt is having due to a growing number of internal software systems be solved through the development of a web solution?*" Through this project, I have learned that there is more than one way to solve the issues that the employees at Mennt were having, and not every solution is viable.

The Design Thinking framework provided me with a great, user-centered approach to figuring out the correct issue instead of jumping to conclusions about what solution would be best for the employees at the company. With the use of universal design principles, the needs of all current and future users will be considered in the design and development process, which gives the best potential to solving issues that are happening now, without creating issues that will need to be solved in future development.

Focusing on researching the issue at hand first and universal design is also a great start to thinking about sustainability in the projects we create. Solving the correct problem leads to less development of unnecessary solutions, which means less data will be created and placed in energy-consuming data centers. Thinking about future users ensures that the solution is as future-proof as possible, which can potentially require less development and upgrading of the solution in future versions.

7 SOURCES

Agile Business Consortium (2014) *Chapter 10: MoSCoW Prioritisation*. Available at: https://www.agilebusiness.org/dsdm-project-framework/moscow-prioririsation.html (Accessed: 29 April 2023).

Andersen, E. S. (2020) *Prosjektarbeid - En veiledning for studenter*. 5 edn. Bergen: Fagbokforlaget.

Balsamiq Studios (no date) *Quick and Easy Wireframing Tool*. Available at: https://balsamiq.com/wireframes/ (Accessed: 25 April 2023).

Budio, R. (2020) *Dark Mode vs. Light Mode: Which Is Better?* Available at: https://www.nngroup.com/articles/dark-mode/ (Accessed: 27 April 2023).

Chromatic (no date) *Component Driven User Interfaces*. Available at: https://www.componentdriven.org (Accessed: 4 May 2023).

Freitag, C. *et al.* (2021) The climate impact of ICT: A review of estimates, trends and regulations, *arXiv preprint arXiv:2102.02622*.

Frost, B. (2016) *Atomic Design*. Pittsburgh: Brad Frost.

G2 (2023) *Best Project Management Software with Time & Expense Capabilities*. Available at: https://www.g2.com/categories/project-management?utf8=✓&order=g2_score&filters%5Bverified_features%5D%5B%5D=9668 (Accessed: 17 March 2023).

Gibbons, S. (2016) *Design Thinking 101*. Available at: https://www.nngroup.com/articles/design-thinking/ (Accessed: 8 March 2023).

Halarewich, D. (2016) *Reducing Cognitive Overload For A Better User Experience*. Available at: https://www.smashingmagazine.com/2016/09/reducing-cognitive-overload-for-a-better-user-experience/ (Accessed: 24 April 2023).

Harley, A. (2014) *Icon Usability*. Available at: https://www.nngroup.com/articles/icon-usability/ (Accessed: 27 April 2023).

Henriksen, H. K. (2023) - *Vi står for omtrent like store utslipp som fly. Men det snakkes det nesten ikke om*. Available at: https://www.kode24.no/artikkel/vi-star-for-omtrent-like-store-utslipp-som-fly-men-det-snakkes-det-nesten-ikke-om/78676338 (Accessed: 12 May 2023).

IKT-løsninger, F. o. u. u. a. (2013) *Forskrift om universell utforming av informasjons- og kommunikasjonsteknologiske (IKT)-løsninger*. Available at: https://lovdata.no/forskrift/2013-06-21-732/§4.

Interaction Design Foundation (no date-a) *User Centered Design*. Available at: https://www.interaction-design.org/literature/topics/user-centered-design (Accessed: 4 May 2023).

Interaction Design Foundation (no date-b) *Heuristic Evaluation*. Available at: https://www.interaction-design.org/literature/topics/heuristic-evaluation (Accessed: 10 May 2023).

Kalbag, L. (2017) *Accessibility for Everyone*. New York: A Book Apart.

Krause, R. (2021) *Maintain Consistency and Adhere to Standards (Usability Heuristic #4)*. Available at: https://www.nngroup.com/articles/consistency-and-standards/ (Accessed: 26 April 2023).

Krug, S. (2014) *Don't Make Me Think, Revisited - A Common Sense Approach to Web Usability*. 3rd edn. Berkeley, Calif: New Riders.

Kuijt, A. (2023) *Which cloud computing platform is the most environmentally-friendly?* Available at: https://www.xomnia.com/post/ai-carbon-footprint/ (Accessed: 6 May 2023).

Lavi, H. (2022) *Carbon Intensity of Data Centres*. Available at: https://public.flourish.studio/visualisation/9338113/ (Accessed: 6 May 2023).

Microsoft (2023) *Sign in with Microsoft: Branding guidelines for applications*. Available at: https://learn.microsoft.com/en-us/azure/active-directory/develop/howto-add-branding-in-apps (Accessed: 13 May 2023).

Mightybytes (no date) *How it Works*. Available at: https://ecograder.com/how-it-works (Accessed: 14 May 2023).

Nielsen, J. (1994) *10 Usability Heuristics for User Interface Design*. Available at: https://www.nngroup.com/articles/ten-usability-heuristics/ (Accessed: 26 April 2023).

Nielsen, J. (1997) *Search and You May Find*. Available at: https://www.nngroup.com/articles/search-and-you-may-find/ (Accessed: 26 April 2023).

Nielsen, J. L., Thomas K. (1993) A mathematical model of the finding of usability problems, *INTERCHI93: Conference on Human Factors in Computing, Amsterdam*.

Norman, D. A. (2013) *The Design of Everyday Things*. Revised and expanded edition. edn. New York: Basic Books.

Oberhaus, D. (2019) *Amazon, Google, Microsoft: Here's Who Has the Greenest Cloud*. Available at: https://www.wired.com/story/amazon-google-microsoft-green-clouds-and-hyperscale-data-centers/ (Accessed: 6 May 2023).

Oberlo (2023) *What Percentage of Internet Traffic is Mobile?* Available at: https://www.oberlo.com/statistics/mobile-internet-traffic (Accessed: 26 April 2023).

Richter, F. (2023) *Big Three Dominate the Global Cloud Market*. Available at: https://www.statista.com/chart/18819/worldwide-market-share-of-leading-cloud-infrastructure-service-providers/ (Accessed: 6 May 2023).

Rosenfeld, L. M., Peter; Arango, Jorge (2015) *Information Architecture: For the Web and Beyond*. 4th edn. Sebastopol: O'Reilly Media, Inc.

Salazar, K. (2016) *The Illusion of Completeness: What It Is and How to Avoid It*. Available at: https://www.nngroup.com/articles/illusion-of-completeness/ (Accessed: 26 April 2023).

Sengupta, A. (2022) *Does The Dark Mode On The Computer Actually Save Electricity?* Available at: https://www.scienceabc.com/innovation/does-the-dark-mode-on-the-computer-actually-save-electricity.html (Accessed: 12 May 2023).

Statista (2021) *Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2020, with forecasts from 2021 to 2025*. Available at: https://www.statista.com/statistics/871513/worldwide-data-created/ (Accessed: 6 May 2023).

Statistisk Sentralbyrå (2019) *Nyetablerte foretaks overlevelse og vekst*. Available at: https://www.ssb.no/virksomheter-foretak-og-regnskap/virksomheter-og-foretak/statistikk/nyetablerte-foretaks-overlevelse-og-vekst (Accessed: 7 March 2023).

Steane, J. (2014) *The Principles & Processes of Interactive Design*. London: Fairchild.

The Authority for Universal Design of ICT (no date-a) *WCAG-standarden*. Available at: https://www.uutilsynet.no/wcag-standarden/wcag-standarden/86?f%5B0%5D=t2%3A130 (Accessed: 12 May 2023).

The Authority for Universal Design of ICT (no date-b) *Gjeldende regelverk og krav*. Available at: https://www.uutilsynet.no/regelverk/gjeldende-regelverk-og-krav/746 (Accessed: 13 May 2023).

The Workplace Regulations (2013) *Regulations concerning the design and layout of workplaces and work premises*. Available at: https://lovdata.no/SFE/forskrift/2011-12-06-1356/§2-4.

Tomitsch, M. *et al.* (2018) *Design. Think. Make. Break. Repeat. A Handbook of Methods*. Amsterdam: BIS Publishers.

United Nations (no date-a) *The 17 Goals | Sustainable Development*. Available at: https://sdgs.un.org/goals (Accessed: 13 March 2023).

United Nations (no date-b) *Sustainability*. Available at: https://www.un.org/en/academic-impact/sustainability (Accessed: 5 May 2023).

United Nations (no date-c) *Article 2 - Definitions*. Available at: https://www.un.org/development/desa/disabilities/convention-on-the-rights-of-persons-with-disabilities/article-2-definitions.html.

Vermandel, C. (2023) *Meilisearch Strapi Plugin*. Available at: https://market.strapi.io/plugins/strapi-plugin-meilisearch (Accessed: 30 April 2023).

Wholegrain Digital (no date) *How does it work? - Website Carbon Calculator*. Available at: https://www.websitecarbon.com/how-does-it-work/ (Accessed: 14 May 2023).

Wroblewski, L. (2011) *Mobile First*. New York: A Book Apart.

8 LIST OF FIGURES AND TABLES

8.1 Figures

8.2 Tables

9 APPENDIX

Appendix A. Project contract

Appendix B. Interview guides

Appendix C. Screenshots from Affinity diagramming

Appendix D. Digital questionnaire for employees

Appendix E. Digital questionnaire for others

Appendix F. Usability test script

Appendix G. Usability testing results

Appendix H. Source code for Menntor

Appendix I. Screenshots from Menntor

NTNU
Norwegian University of
Science and Technology