Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Sunniva Mathea Runde
Kaja Løvsjø Solberg

Nykken: Visualizing Hydrological Data

A Web-Based Application Connected to a Hydrological Measurement Station

Master's thesis in Informatics
Supervisor: George Adrian Stoica
June 2021

NTNU
Norwegian University of
Science and Technology

Sunniva Mathea Runde
Kaja Løvsjø Solberg

Nykken: Visualizing Hydrological Data

A Web-Based Application Connected to a
Hydrological Measurement Station

Master's thesis in Informatics
Supervisor: George Adrian Stoica
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Abstract

With population growth and climate changes, urban drainage and water supply systems are more important than ever. More frequent and heavier rainfalls cause higher water levels in the drainage systems and increase the risk of flooding in cities. Researchers therefore measure and monitor hydrological data to predict the runoff in the sewers, to help facilitate the increased water levels.

Data visualization is presenting data graphically. The goal of data visualization is to convey information effectively and clearly to help people interpret and understand it.

The aim of this thesis was to provide users access to the data recorded at *Risvollan measurement station* (RMS), a hydrological measurement station in Trondheim. Using a design and creation methodology, with data collected through interviews, observations, and questionnaires, this thesis explores the visualization of hydrological data.

The process also included designing, developing, and testing a potential solution, *Nykken*: A web-based visualization tool that makes hydrological data from the RMS accessible to its users, both through data download and visualization.

The results show that Nykken is a good tool to get access to the data from the RMS. It also shows that the users have certain expectations as to how hydrological data should be visualized.

The GitHub repository for this project can be found at `https://github.com/Kajalso/Nykken`.

Nykken is currently deployed at `https://nykken.netlify.app/`.

Sammendrag

Med populasjonsvekst og klimaendringer er vann- og kloakksystemer viktigere enn noen sinne. Økt nedbør fører til mer vann i avløpsrørene og er med på å øke flomfaren i byene. Derfor kontrollerer og overvåker forskere hydrologiske data for å forutsi avløpsmengden i kloakken for å tilrettelegge for de økte vannmassene.

Å presentere data grafisk kalles datavisualisering. Målet med datavisualisering er å overbringe informasjon på en effektiv og tydelig måte, slik at informasjonen er lett å tyde og forstå.

Målet med denne oppgaven var å gi brukere tilgang til datamålinger fra Risvollan målingsstasjon (RMS), en hydrologisk målingsstasjon i Trondheim. Ved hjelp av en *design and creation* metodologi og data samlet inn gjennom intervjuer, observasjoner og spørreskjemaer undersøker denne oppgaven visualisering av hydrologiske data.

Prosessen besto også av å designe, utvikle og teste en potensiell løsning, *Nykken*: et web-basert visualiseringsverktøy som gjør den hydrologiske dataen fra RMS tilgjengelig for sine brukere, både gjennom nedlasting og visualisering.

Resultatene viser at Nykken er et godt verktøy for å få tilgang til data fra RMS: De viser også at brukerne har forventninger til hvordan hydrologisk data skal visualiseres.

GitHub-repoet for dette prosjekt finnes her: `https://github.com/Kajalso/Nykken`.

Nykken finnes på: `https://nykken.netlify.app/`.

Acknowledgements

We would like to sincerely thank our stakeholder, Robert Meier, for his advice and support throughout this project. His insight into and contacts in the Water and wastewater systems engineering group, as well as his positive attitude, have been invaluable.

We would also like to thank all the participants and experts involved in this project. Without their contributions and continuous feedback, we would not have been able to complete our thesis.

Sunniva Mathea Runde & Kaja Løvsjø Solberg

Trondheim, June 11th 2021

Table of Contents

List of Figures

List of Tables

Abbreviations

API	Application Programming Interface
CSS	Cascading Style Sheets
CSV	Comma-separated Values
D3	Data-Driven Documents
DOM	Document Object Model
DQ	Documentation Quality
ERD	Entity Relationship Diagram
HTML	HyperText Markup Language
IDV	Interactive Data Visualization
JPEG	Joint Photographic Experts Group
ML	Machine Learning
MVP	Minimum Viable Product
NVE	Norges vassdrags- og energidirektorat
PDF	Portable Document Format
PPI	Pixels Per Inch
PNG	Portable Network Graphics
RMS	Risvollan measurement station
RQ	Research Question
SRC	Semantically-Resonant Colors
SUS	System Usability Scale
SVG	Scalable Vector Graphics
UCD	User Centered Design
WWSE	Water and Wastewater System Engineering

Part I

Introduction

This part contains the background motivation and a detailed problem description. In addition, the scope of the thesis will be introduced, as well as the target audience of the end product. Lastly, this introduction includes a thorough outline of the remaining parts of the thesis.

Chapter 1

Motivation

1.1 Risvollan Measurement Station

Due to population growth, urbanization, and climate change, flooding in urban areas and disruption in water supply or drainage systems are increasing [10]. In urban environments, two types of water require drainage; wastewater from human water consumption and stormwater from precipitation [3]. Wastewater comes from both private and industrial use, while a range of weather conditions affects stormwater. Monitoring and understanding water flows in urban drainage systems is therefore important to predict and prevent flooding and disruption.

Hydrological data are used to predict future runoff in the urban drainage systems. Hydrology is water science and hydrological data are statistical data related to water [25]. The *Water and wastewater system-engineering* (WWSE) group at the *Norwegian University of Science and Techonology* (NTNU) consists of researchers and engineers working with and analyzing hydrological data to help improve the urban drainage systems in Trondheim [10]. Their research is related to water quality, treatment of wastewater, stormwater management, sustainable rehabilitation of current urban water systems, and treatment of process water from oil and gas exploitation [10].

The *Risvollan measurement station* (RMS) is an urban hydrological station, located at Risvollan, south of Trondheim city. The station has several instruments, including different precipitation gauges, temperature, humidity, radiation and wind speed sensors, as well as instruments to measure runoff, all aimed at collecting hydrological data [28].

Up until 2020, there was no easy access to the data recorded at RMS. Inconsistencies in data files and formats led to extra workloads and confusion among the WWSE-group and students entering the hydrological field. In 2020, a new *Application Programming Interface* (API) was built to give users access to the sensor measurements. Therefore, a need for an application interface that would provide access to this API emerged. To give the users an overview of the data received from the API, some form of data presentation needed to be implemented.

1.2 Data Visualization

Data visualization has always been an important part of hydrology and might contribute to the work conducted by the WWSE-group [33].

Data visualization is presenting data graphically to communicate and make sense of the data [15]. Usually, the data displayed is abstract information, such as statistical information. By presenting this abstract information in a visual way, it can help people better understand and interpret the information.

Knowledge about information processing has proved to be important when designing good visualizations [15]. By knowing how users perceive sensory input and what you want to convey, you can use visualization to ease the users' information processing, thus allowing them a faster and better understanding of the data you are presenting [15].

1.2.1 Processing Data Visualizations

When processing information, and more specifically, when processing data, the presentation of the data can affect the performance of a related task. Several studies have investigated how visual aids affect analysis performance. For example, one study investigated how information presented in charts versus tables affected how students solved the tasks [57]. The study concluded that using charts was more suited for qualitative analysis, than using tables.

Another study compared task solving performance using data presented in tables, charts, or other visual aids [19]. The study found that the participants were more accurate when given well-defined tasks when they were using a *visual network packet analysis tool*, called TNV, than they were when they were using a table. When the participants were given exploratory tasks, researchers found that more insight into the data was gained when the participants were using the TNV. The participants also reported higher user satisfaction, particularly for discovering patterns and irregularities with the TNV [19].

Finally, one study indicated that *interactive data visualization* (IDV) helped non-professional investors understand financial information better [48]. It showed that participants spent less time on their tasks, less time processing the data and that they made better investments than the control group, that did not use IDV. The participants using IDV also reported feeling more confident in their investments and hence were more willing to invest.

1.2.2 Charts

Data can be visualized in a range of ways, and the way the data is presented might affect both how fast the information is processed, and how the information is perceived and interpreted [15].

A chart is a way of displaying data in a simple way, oftentimes as a drawing using lines and curves to show amounts [11].

For the researchers in the WWSE-group, it is therefore important that a visual presentation of the hydrological data from RMS is developed based on their needs and in a way that is best suited for the specific data and use.

Chapter 2

Project Scope

The development and writing of this master's thesis were executed during the Fall semester of 2020 and the Spring semester of 2021. Two students were working on this report, Sunniva Mathea Runde and Kaja Løvsjø Solberg. During the project, the main stakeholder, Robert Meier, contributed like a co-supervisor, attending all of the meetings with the supervisor.

2.1 Project Goal

The goal of this project was to develop and test an application that gave the WWSE-group access to the RMS data and to examine if a visualization of the data could help them in their work with the data.

2.2 Research Questions

For us to create an application aimed at helping the WWSE-group in their daily work through data visualization, we had to investigate and examine what data visualization is and how it is used to display data. This exploration led to the following two research questions:

RQ1: In what ways can interactive visualizations improve/aid the analysis of hydrological data?
This research question seeks to understand the WWSE-groups pain points when working with the data from RMS. It also explores how visual aids can help users process the sensor data and get an overview of the data they want to work with.

RQ2: Which kinds of interactive visualizations of such hydrological data are better suited for the different sensor measurements?
The second research question focuses on choosing the right chart type and design for visualizing different types of data. It will discuss standards for visualization, both in general, but also within this specific user group.

2.3 Target Audience

As this master's thesis is meant to develop an application for the WWSE-group from NTNU, using data from RMS in their research, members of this group are considered our target audience. To represent this group, Robert Meier acted as the main stakeholder, since he was the one developing the API.

Chapter 3

Report Outline

This master's thesis consists of seven parts.

Part I has presented the background motivation and a detailed problem description. In addition, the scope of the thesis will be introduced, as well as the target audience of the end product.

Part II will present the research methodology of the thesis. It will present the different data generation methods used in the different phases of the project, as well as how they were executed.

Part III will present an overview of how and why visualizations should be used. Standards for data visualization are discussed, along with guidelines for visualizing hydrological data. Lastly, three visualization applications are presented.

Part IV will present Nykken, a web application for visualizing and downloading hydrological data from RMS. Firstly, the part will describe the design and development process from the very first idea and concept, the prototyping phase, and lastly the final concept.

Part V will present a detailed description of the results gathered from the different data generation methods.

Part VI will present a discussion of the research methods used in this project. An assessment of the application's visualization design is also included. Lastly, a more detailed evaluation of the entire project will be given, including answers to the research questions and project goal.

Part VII will present a conclusion of the project. Lastly, the future work chapter will discuss recommendations for further work.

Part II

Research Methodology

The following chapter describes the research methodology of the thesis. It will present the different data generation methods used in the different phases of the project, as well as how they were executed.

Chapter 4

Research Method

This chapter presents the methods used during the different phases of this project. We will first describe the overall process itself, before giving a more detailed description of the literature review and the data collection methods used, and how they were executed.

4.1 The Process

This master's thesis was conducted using a design and creation research strategy. Data was generated through multiple methods, such as interviews, observations, and questionnaires. The data were analyzed both qualitatively and quantitatively. Figure 4.1 shows the chosen methods, framed in red.

We started our project by examining different libraries created to support data visualization. A list of criteria was created to compare the available options we decided to test. We then compared the libraries and chose the one that fulfilled the most criteria. This process is further described in Chapter 7.

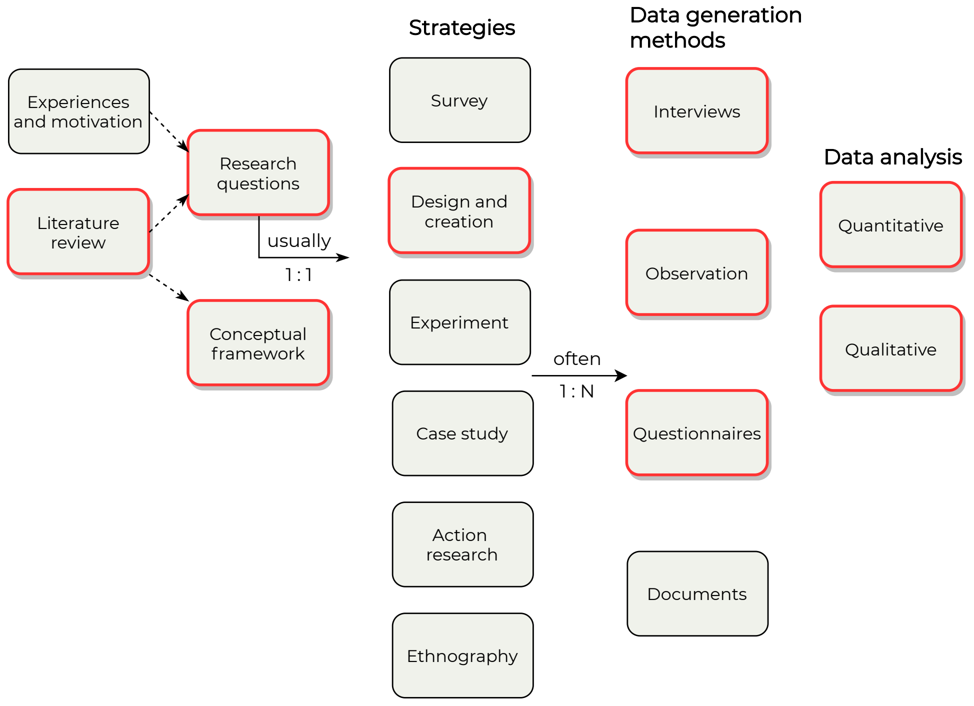


Figure 4.1: The Research Process [43]

Simultaneously with the library testing, we found relevant literature and work to get an understanding of important aspects of visualization theory and data visualization.

The design and creation process was inspired by the steps from *user centered design* (UCD) [71]. UCD is a four-step design methodology that focuses on users and their needs. The four steps in UCD are presented in Figure 4.2.

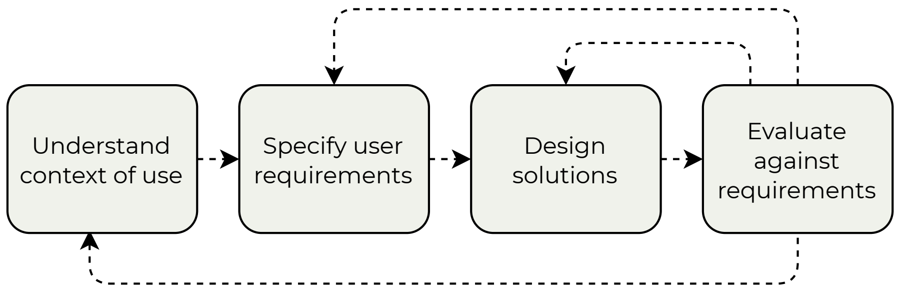

Understand context of use
Specify user requirements
Design solutions
Evaluate against requirements

Figure 4.2: The Four Steps of User Centered Design

We first conducted user interviews with the WWSE-group and experts from *Norges vassdrags- og energidirektorat* (NVE) to learn about their current practices and experiences. We also asked them about their preferred visualizations of hydrological data and hydrological visualization standards.

We then analyzed the interviews to identify the core problems that the WWSE-group was facing when working with the data. Using this information we created personas, scenarios, user stories, and requirements.

Based on these user stories and requirements, we created a prototype. We then invited the participants from the interviews back to test the prototype. After completing these four steps, we analyzed the data collected in the user test. From these data, we evaluated whether our concept and design satisfied the participants' needs. We revisited the different steps to make a final concept and design, prioritized the user stories, and divided them into sprints before the implementation phase.

The final concept was tested in a second user test, where we invited the same participants, using observation, questionnaires, and follow-up interviews to evaluate it.

The implementation phase used elements from the Scrum framework, such as daily stand-ups, sprints with demos at the end, and retrospectives. We used the task board on GitLab as our Scrum Board. We used the three labels *backlog*, *started*, and *completed*. The tasks listed were created after breaking down the requirements for the final concept into smaller issues. This is further explained in Chapter 15.

4.2 Participants

The participants chosen for this project were suggested by the main stakeholder. Since he was working closely with the data from RMS, he had a good insight into whom we should talk to and whom we were developing the application for. This consisted of PhD Candidates and professors from the WWSE-group. He also suggested that we talked to one of the engineers at RMS, as well as experts from NVE, which also had their own sensors at the station. We hoped that this could serve as an inspiration for our own design and development process. For the final user testing, we also asked a bachelor's student working with the same data to participate. An overview of the participants is presented in Table 4.1.

Code	Background	Place of Work
Prof1	Professor	NTNU
Prof2	Professor	NTNU
PhD1	PhD Candidate	NTNU
PhD2	PhD Candidate	NTNU
PhD3	PhD Candidate	NTNU
Tech	Engineer at RMS	NTNU
NVE1	Hydrologist	NVE
NVE2	System developer	NVE
BS	Bachelor's student	NTNU

Table 4.1: Participants

4.3 Data Collection

To improve the quality of our research, we chose to use multiple data gathering methods, also known as method triangulation [43]. This enabled us to compare and corroborate the data collected in one method, with findings from another data gathering method. To further enhance the quality of the data gathered, we also used both qualitative and quantitative analysis. We will now present the different data gathering methods used in this thesis, the aim of each method, and how each method was executed.

The NSD was notified of the project, and the data collection has been done in conformity with their guidelines.

4.3.1 Interview

Interviews are a suitable data generation method when you want to obtain detailed information [43]. In our case, we wanted information from our user groups to better understand their needs. By talking to an experienced system developer at NVE, we could also obtain details from someone with first-hand experience with data visualization, something that is difficult to obtain from reading scientific literature.

The interviews were conducted as part of our initial research phase. The interviews were semi-structured to allow for follow-up questions and to enable the participants to talk more freely in their answers [43]. We also decided to record the interviews to further allow for a better flow in the conversation. When the interviews were being planned, we planned to conduct them physically. As a part of the planning process, we conducted a test interview with the main stakeholder and learned that we could obtain better explanations by allowing the participants to draw out their ideas. However, due to the ongoing pandemic, the interviews were conducted online. To still allow the participants to explain and show their thoughts, we encouraged them to share their screens and let us record it along with the audio.

We also had a short follow-up interview with the participants after each user test. This was used to follow up problems that occurred during the tests, and in case we wanted the users to elaborate on something they did during the user test. At this stage, we chose an unstructured interview to allow for an open discussion with the participants and follow up on their particular experiences with the test.

The information about the audio and screen recording was given to the participants beforehand, along with an information sheet about how these recordings were being saved and stored, and their purpose. The interviews were transcribed before starting the analysis.

Since the interviews with the WWSW-group and the experts from NVE served different purposes, we will discuss their execution separately in the following two subsections.

WWSE-group

The main purpose of these interviews was to gain insight into the current situation of those working with the data from RMS, as well their needs in a potential new visualization application. The questions were divided into four main categories:

- Current situation
- Existing solutions
- Features and functionality
- Motivation and expectation

The full interview guide can be found in Appendix A.

We started every interview by asking the participants to talk a bit about themselves and their daily work. We also asked them to include how they worked with the sensor data from RMS. This was to get an understanding of how their current situation was and if there were any pain points in their daily work. This was also done to help get the interviewees talking, as suggested by Oates [43].

We then moved on to existing solutions. Here we asked what they currently did to make the data usable for their tasks, and which programs they used to do this. We also asked them if they had used any other tools to visualize or look at data, and which features these included.

The last two parts were aimed at our project and how they thought a new application could be of use to them. We asked them if they had a preferred way of seeing the different types of data, and which features they thought were important in a visualization tool.

NVE

The main purpose of the interviews with the experts from NVE was to gain knowledge about the development process and their approach to creating data visualization applications. We also aimed to learn about existing visualization tools and applications for inspiration.

These interviews were more unstructured than the user interviews. As with the user interviews, we had thought about some overall topics to discuss, but these were more directed towards the actual development process. Because of this, we decided against having too many questions prepared, but rather let the participants speak freely about their experience with visualizing hydrological data.

Some topics we discussed were:

- Current situation
- Existing solutions
- Experiences from similar projects

4.3.2 Observation

Observations were done during both user tests, to evaluate the prototype and the final concept. The observations were done by looking at the participants' screen during the user tests as well as seeing the participants on camera as they were interacting with the prototype and final concept. Because of covid-19, physical observations were not possible. We therefore chose this solution instead, as it would best simulate a physical observation.

The aim of observing the users during the user tests was to look for common behaviors among the users. This could provide us with a good indication of how the application flow should be.

The footage of both the screen and the participants' faces were recorded along with the audio, to enable us to retrace comments and problems that occurred during the user tests, as well as the feedback given after the test was finished. The participants were informed about this and gave their consent before the observation started.

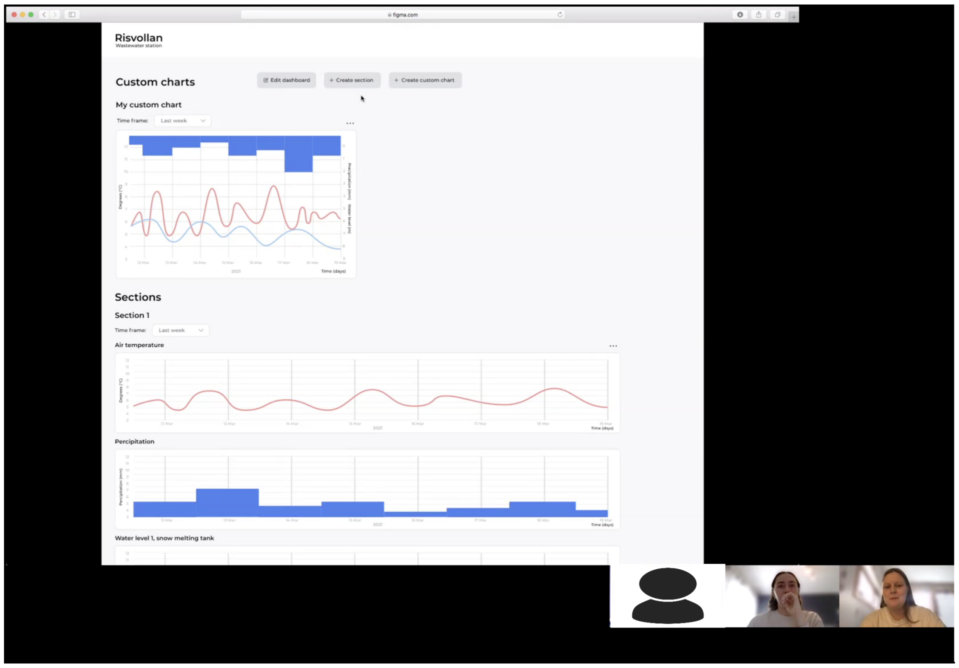


Figure 4.3: A Screenshot from the User Test. The Participant's Face Has Been Replaced.

The participants were given a set of scenarios with different tasks to see how they would use the application to solve the given tasks. The tasks were planned and were both read out loud and sent to the participants in the meeting chat, so they could take their time to read them properly before trying to solve them. Because the observations were planned and the participants knew we were observing them, the observations can be explained as systematic and overt [43].

During the user tests, we used the ten points for user testing described by Svanæs [64]. These steps include, among other things, a brief description of the purpose of the test, teaching the participants how to think out loud, explaining that we are unable to provide any help during the testing, and explaining the limitations of the prototype and final concept. A more detailed description of the two user tests can be found in the following sections.

1st User Test

In the first user test, the users were given five scenarios with coherent tasks they were asked to complete. The goal of this user test was to examine if our prototype's features could help solve some of the pain points uncovered during the user interviews. We also wanted to test how the users liked the presentation of the included features, as well as the application flow. The tasks given during the user test were the following:

1. You can assume for all tasks that today is March 19th, 2021. You are on the Risvollan wastewater station website and you want to see the air temperature measured on the 17th and 18th of March. What do you do?
2. Instead of seeing the air temperature from the latest week, you are more interested in seeing the temperature from the last two hours recorded instead. What do you do now?
3. You want to group air temperature, precipitation, and water level 1 from the snow melting tank together to be able to change their resolution simultaneously to show the last week. What do you do?
4. You now want to export the same data in one chart as a PNG file before adding it to your dashboard with the name "My custom chart". What do you do?
5. You also want the raw data files used in the newly created graph. What do you do?

2nd User Test

In the second user test, the participants were given similar scenarios as in the first user test, but with some tweaks. Since the prototype and final concept ended up sharing most of the same application flow, we wanted to prevent the participant from acting by their memory from the last test, but rather as if this was a new situation. The tasks the participants had to complete in the second user test, were the following:

1. You are writing a paper and need data from Risvollan. Before you do anything, you therefore want to check when Nykken was last updated. What do you do?

2. You have just started using Nykken, and you want to customize your dashboard to see the charts of the sensors you mostly interact with. Those sensors are the Air Temperature, Humidity, and one Wind speed. What do you do?
3. You want to take a closer look at the Air temperature measurements from the first day of April 2021, before you download the raw data for further use. What do you do?
4. In your paper you want to include a picture of the Air Temperature chart showing the measurements from May 1st, but only showing the hourly average measurements. What do you do?
5. You realize that you often want to see both temperature measurements (Temperature and Air Temperature) in the same time frame, so you want to group them together and control them using just one time frame picker. Before you add this group to your dashboard, you want to give it a suitable name. How do you do that?

During the observations, we focused on seeing how the participants interacted with the available features. Combined with the participants thinking out loud, this provided valuable insight into their imagined workflow. After the participants had completed all the tasks, we had a short discussion about any additional thoughts they had during the completion of the tasks. We also asked the participants to give their honest feedback and stated that all feedback was good feedback.

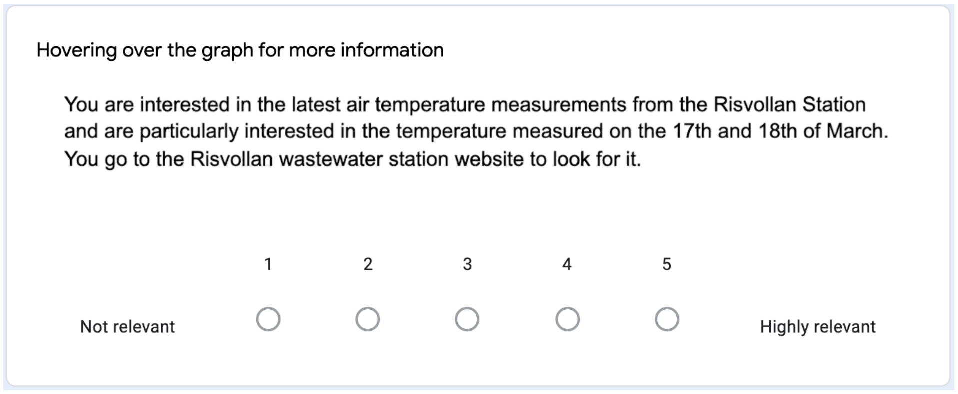
4.3.3 Questionnaire

In conjunction with the user test, the participants were also asked to fill out a short questionnaire. This was done to collect more quantitative data about how the users experienced interacting with the prototype and final concept and how they liked the features presented in the prototype.

The first questionnaire was divided into three sections: demographic data, relevancy of the presented features in the prototype, as well as the *System Usability Scale* (SUS) [60]. The entire first questionnaire can be found in Appendix C.

The questions in the first section included occupation, age, and gender, to see if this affected how the users rated each feature's relevancy, and how they perceived the system's usability.

The following section in the first questionnaire used the *Likert Scale* to rate the different features in regards to their relevancy for the user [43]. These features were linked to the different tasks the user was given during the user test. The scale had a range of five responses ranging from *Low relevance* to *High relevance*. We chose to include these questions because we wanted to see if the presented features were the desired features for the user. We also wanted to know if any of the features were more desired than others, to know what to prioritize when developing the final concept. An example of this can be seen in Figure 4.4.


Hovering over the graph for more information

You are interested in the latest air temperature measurements from the Risvollan Station and are particularly interested in the temperature measured on the 17th and 18th of March. You go to the Risvollan wastewater station website to look for it.

	1	2	3	4	5	
Not relevant	○	○	○	○	○	Highly relevant

Figure 4.4: Example of a Question from the First Questionnaire

To measure the usability of our solution, we included the SUS in both questionnaires. The SUS is described as a "quick and dirty" scale to measure usability, as its results are considered useful and reliable even for smaller sample sizes [60]. Since our test population consisted of six people or less, we found it to be an appropriate method to examine the system's usability. We also wanted to see if there were any changes in the SUS score between the prototype and the final concept. The SUS statements and the scores from both questionnaires can be found in Table 18.2 in Part V.

We also decided to include a short text input field, for any additional comments. This allowed the user to share their honest opinion without having to say it directly to our faces, but also to add something if they forgot to mention it during the user tests.

4.4 Data Analysis

4.4.1 Interview

All the user interviews were fully transcribed. After each interview was transcribed, it was summarized into a table, with the four overall themes from the interviews as the overall categories. As more and more interviews had been transcribed and summarized, we used an inductive approach to create sub-categories in a bigger table with all the information derived from the interviews. We used this matrix to look for themes and patterns in the user responses. These themes and patterns served as the foundation of our personas and user stories, that were created before we started prototyping. The full matrix can be found in Appendix B.

The interviews with the experts from NVE were not fully transcribed. Instead, we took notes during the interview. We later re-watched the interviews to add additional notes, in case something was missed during the first note-taking.

4.4.2 Observation

The observations were analyzed by studying the video recordings of the user tests. The participants' mouse movements, actions, comments, and expressions were described. By observing how the participants executed the different tasks, we uncovered common practices and behaviors among the participants. This gave us insight into how the participants interacted with the different features in the prototype and final concept, as well as which option they would choose if several were available.

4.4.3 Questionnaire

Both questionnaires were analyzed by using the SUS-scoring method to calculate the SUS score. The SUS consists of 10 statements. It is calculated by subtracting 1 from each odd statement and by subtracting the user's response from 5 for each even statement. This means that for each statement, the top score is 4. The scores are added together and multiplied by 2.5, to get a score from 1 to 100.

According to Jeff Sauro, a score higher than 68 is considered *above average*, and "a SUS score of 74 has higher perceived usability than 70% of all products tested" [60]. How these scores were calculated can be found in Section 18.4.

The first questionnaire also contained a rating of each feature's relevancy. This was analyzed by looking at the distribution of the ratings and ordering them from highest rated to lowest rated, giving us an indication of which features to prioritize.

4.5 Validity and Reliability

There is always a risk that the level of validity and reliability of the data collected is not sufficiently high. Oates mentions that the *Hawthorn Effect* can affect the outcome of the respondents' answers or participants' behavior because they know they are being observed [43]. This effect can result in a more positive outcome than what is the case.

Since the testing period was limited, the results from the observation and questionnaire do not provide evidence to conclude that the solution will work long term or that it will be used in the ways we imagined. We can, however, based on the results, determine if the concept is perceived as a good tool to access the data from RMS and which features are the most desired by the different user groups that have tested the solution.

Part III

Preliminary Study

The preliminary study presents an overview of how and why visualizations should be used. Standards for data visualization are discussed, along with guidelines for visualizing hydrological data. Lastly, three visualization applications are presented.

Chapter 5

Data Visualization

In this chapter, a detailed description of how data can be visualized will be presented. Firstly, we will present different chart types and their usage, before discussing different approaches to data visualization. Lastly, the guidelines for visualizing hydrological data will be discussed.

5.1 Chart Types

To best visualize your data, you need to know the type of data you are visualizing and what your visualization seeks to achieve [8, 18, 21]. We will now present the most common chart types and which analysis tasks they are best suited to solve.

Line Chart

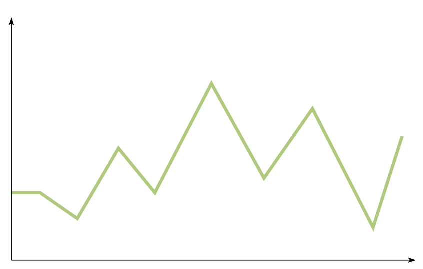
The line chart is a very common chart type [21]. It shows how two continuous variables are related to each other, with one of the variables often being time [8]. Therefore, the line chart is commonly used to show trends over time, by connecting several distinct data points using the line to show the continuous evolution of the data [21].

Bar Chart

Bar charts are an effective way of comparing information and are also one of the more common chart types [21]. This can be done, either by showing values of a single continuous variable for multiple separate entities or by sampling one variable in a given interval [8]. This allows the user to quickly highlight differences, spot outliers, and high/low points in the data [21].

Pie Chart

Another commonly used chart is the pie chart. Pie charts are used to show the relative distribution of data or categories that make up a whole, which is also the only recommended instance of where the pie chart should be used [8]. Since people are not


(a) Line Chart

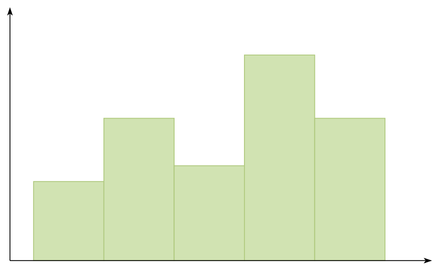
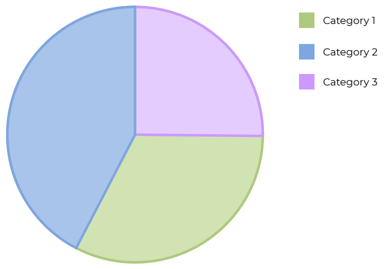

(b) Bar Chart

Figure 5.1: Chart Types

usually that good at comparing sizes, they could have a hard time comparing wedges of similar sizes and key points from the data can consequently be missed [21, 32].

Scatter Chart

Similar to the line chart, the scatter chart can show the correlation between two continuous variables, as points or plots in the chart, like a scatter. The scatter chart can give a sense of trends, concentrations, and outliers that could direct the interpreter where to focus their investigation efforts further [21].



(a) Pie Chart

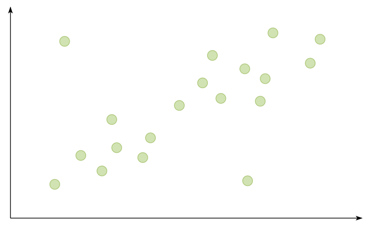
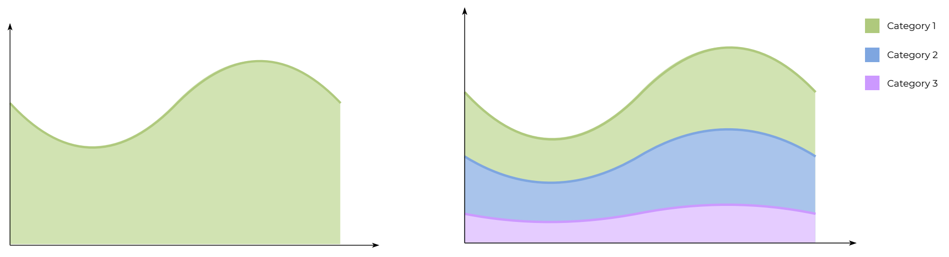

(b) Scatter Chart

Figure 5.2: Chart Types

Area Chart

Area charts are a form of line chart, but with a shaded area between the line and the horizontal axis [8]. They are commonly used when multiple line graphs are stacked on top of each other to represent a whole, where each area represents its own category [8]. Examples of single and multiple area charts can be seen in Figure 5.3.



(a) Area Chart (b) Area Chart with Multiple Lines

Figure 5.3: Area Chart Types

5.2 Standards

According to Charles Kostelnick, there have been several attempts to create standards for data displays in the last two centuries [32]. Four different approaches to this that have gained popularity according to him are the *Conventional*, *Perceptual*, *Informational* and *Aesthetic* approaches. While they all aim to visualize data in the most effective and persuasive way, their motivation, reasoning, and approach vary.

5.2.1 Conventional

The conventional standard is divided into two general types, both bound to their own rules by nature, to meet the reader's expectation through its design practices [32].

Genres

This convention is one of the most widespread conventions [32]. It is based on a lexicon of data displays genres, from now on referred to as charts, created by Robert L. Harris [22]. These are similar to the ones mentioned in Section 5.1. With these charts come guidelines on how to use them. These include rules for choosing the best-suited chart for visualizing your data, similar to the guidelines presented in the previously mentioned section. Other rules state how the charts should be designed. A pie chart should for instance always begin at the 12 o'clock position and the y-axis should start at 0 [32]. With technology offering tools to help visualize data, these charts have grown in popularity since they are commonly offered and reinforced by these tools.

Disciplines

Within a community or group, unspoken rules for visualizations can emerge [32]. This also happens within disciplines and organizations. Rules such as color usage and stylistic choices made in-house in a company may apply to the designers creating the visualizations for that company. This can also apply to specific chart usages for certain types of data [32].

5.2.2 Perceptual

The perceptual standard is derived from empirical research, through observation and testing [32]. Researchers have studied the relationship between the eye, brain, and image, and their findings corroborate universal design principles. As mentioned in Section 5.1 about pie charts, people are not usually good at comparing sizes, something that the research within the perceptual standard field has proven. While people can easily identify which wedge is the biggest and smallest, they are unable to tell how much bigger the biggest is compared to the smallest. The same applies to volume, color scales, and depth. They can be easily distinguished but are hard to compare. Therefore, if you want to compare data, you should position it along a common scale, using for instance bars, lines, or dots [32].

Ideas from the gestalt principles are also mentioned in the perceptual standard. The figure-ground principle states that people instinctively separate objects into being in the foreground or being in the background [66]. This can be enhanced by making the foreground stand out from the background, by increasing the contrast between the two. In a chart, this means having a transparent background to draw the attention and focus to the actual data displayed in the chart [32].

5.2.3 Informational

The informational standard focuses on, as implied by its name, the information to be visualized [32]. It emphasizes the clear and efficient uptake of data, thus sharing many similarities with the perceptual standard. The reasoning behind it, however, differs. The idea is that visualized data should be as transparent and objective as possible, and to achieve that, one must remove all unnecessary information from the view [32].

One activist behind this standard was Edward Tufte. He referred to everything that cluttered the chart design as “chartjunk” [69]. This so-called chartjunk can be 3D effects, color where color is not needed, and too much text. By enforcing these principles, the charts that are created following the informational standards end up looking similar to charts created by the perceptual standard.

5.2.4 Aesthetic

The last standard described by Kostelnick is the aesthetic standard. This standard does not have any rules or guidelines like the other three standards but focuses more on the fact that designers are affected by and make decisions based on trends and what is seen as visually pleasing [32]. Some designers may choose a simplistic view, similar to the informational standard, while others may choose to include colors, figures, and frames, depending on their audience. A sales presentation trying to encourage people to buy a summer house may add some palm trees and bright colors to evoke summer feelings [32].

5.3 Additions to Improve Readability

This section presents ways of improving chart readability, which refers to how easily an interpreter can read and understand the information presented in a chart.

5.3.1 Interactive Data Visualization

Making data visualizations interactive, gives the user the ability to change the visualization to fit their specific needs or tasks, hence helping them with decision making [29]. As previously mentioned, IDV can aid non-professional investors by reducing their uncertainty level when doing investments [49].

All it takes for a visualization to be interactive is that it allows for human input [63]. Examples of this could be a button click, moving a slider, or zooming in and out in a chart. Figure 5.4 shows an interactive scatter chart where the user can compare different flower data by interacting with the drop-down menus.

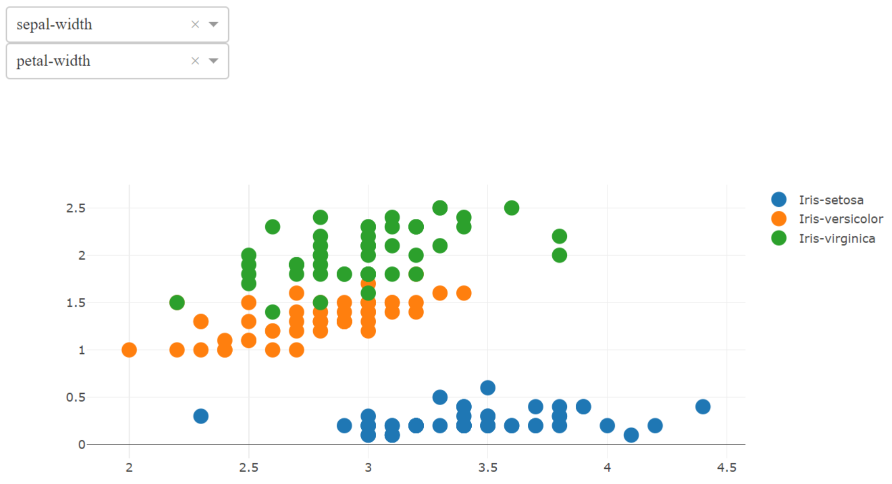


Figure 5.4: Example of Interactive Data Visualization [1]

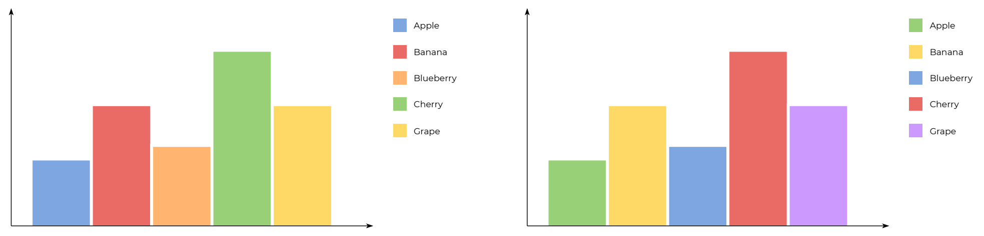
5.3.2 Color

Color is one of the first things we notice when we look at a chart and if chosen wisely, it can both create contrast between and highlight specific data. By using basic color theory and reflection to make the color decisions in a chart, you can significantly improve its readability [12, 18].

Semantically-Resonant Colors

Concepts can be related to color. This could either be because of their physical appearance like “bananas are yellow” or common metaphors, such as “love is red”. When concepts are paired with a fitting concept, they are called *semantically-resonant colors* (SRC) [34].

Figure 5.5 shows two ways of assigning colors to charts: Using default colors or semantically-resonant color assignment. According to research made by PhD student Sharon Lin and computer science professor Jeffrey Heer, people will use on average 10% less time on a task using bar charts with semantically-resonant colors versus bar charts using non-resonant colors [34]. One reason for this is that these colors can enable the user to take advantage of existing familiar relationships, which require less information processing. Additionally, time can be saved by not having to re-check the color legends every time a user wants to look at the chart to remember which bar represents which concept.



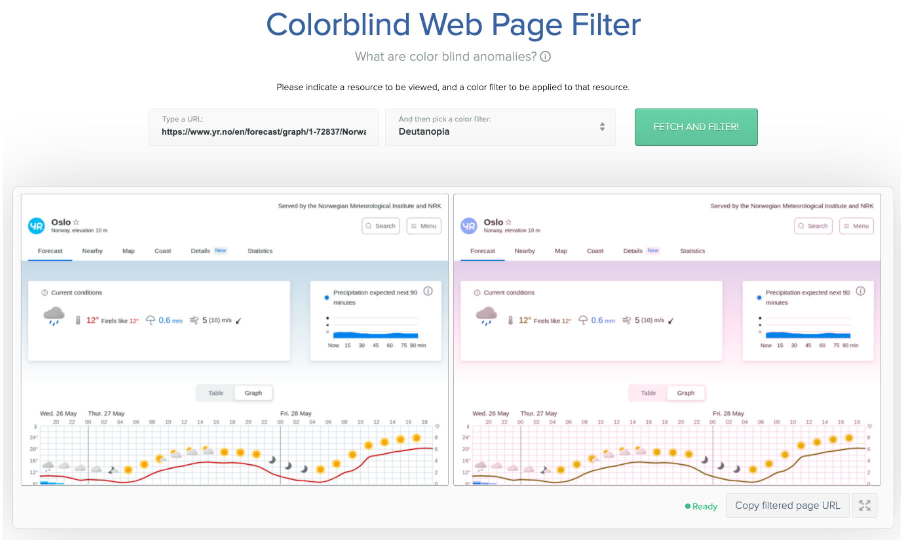
(a) Default Colors Approach (b) SRC Approach

Figure 5.5: Different Color Choice Approaches

Color Universal Design

In a process where the users are in focus, it is natural to consider *universal design*. Universal design means designing solutions in such a way that they do not hinder people with varying functional capabilities.

Around 8% of men and 1% of women see color differently from the ordinary person. *Color Universal Design* (CUD) was created to be a user-oriented design system that would address the issues that these people face when they are absorbing information through color. This would allow for a more accurate conveying of information for as many individuals as possible [47].



Colorblind Web Page Filter

What are color blind anomalies?

Please indicate a resource to be viewed, and a color filter to be applied to that resource.

Type a URL: https://www.yr.no/en/forecast/graph/1-72837/Norwa

And then pick a color filter: Deutanopia

Figure 5.6: Colorblind Web Page Filter by Toptal [67]

Colorblind Web Page Filter is a tool created by the freelance company Toptal [67]. Its goal is to show how people with different variants of color vision deficiency perceive web pages. This tool can be used to discover potential problem areas in an application's design. Figure 5.6 shows the tool in action using the website Yr.no with the deficiency *deuteranopia* chosen, which is the most common of the color vision deficiencies.

There are multiple ways of preventing interpretation issues for the color deficient when visualizing data. One option is to check the color contrasts in a greyscale, to make sure that colors that represent different values still can be differentiated when the colors themselves are not perceivable [47]. Another way of approaching the issue is to give the graphs distinct patterns. In that way, individuals with all levels of color deficiency can still be able to differentiate the values from each other. An example of a chart using patterns in its visualization can be seen in Figure 5.7.

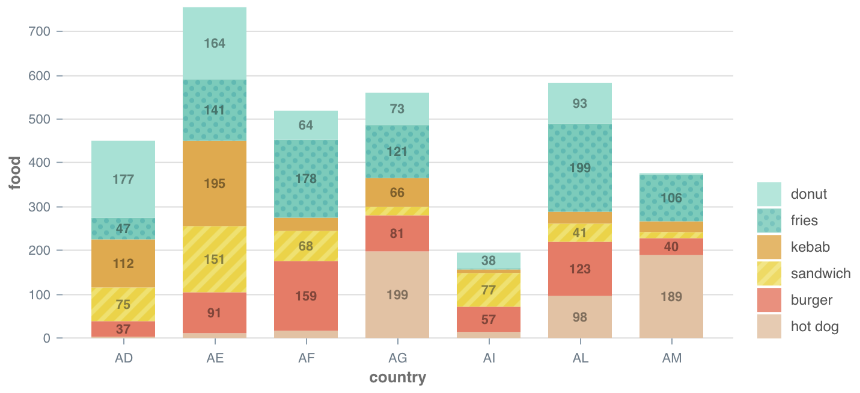


Figure 5.7: Bar Chart Using Patterns [40]

5.3.3 Reference Points

One way of aiding users when they are interacting with data visualizations is to give them reference points when they are looking at the data. There are two main ways of doing this: using reference lines or band charts.

Reference Lines

When a user is interacting with a chart, they may not have anything to compare the data to or anything that tells them what the data usually looks like. Reference lines can be added to show the user where the "normal" or "expected" value for the data usually is located. Figure 5.8 shows how the meteorological weather website Yr.no gives the user insight into the amount of rainfall that is normal for each month. Knowing this information, the user can easily tell which data are above or below the normal value.

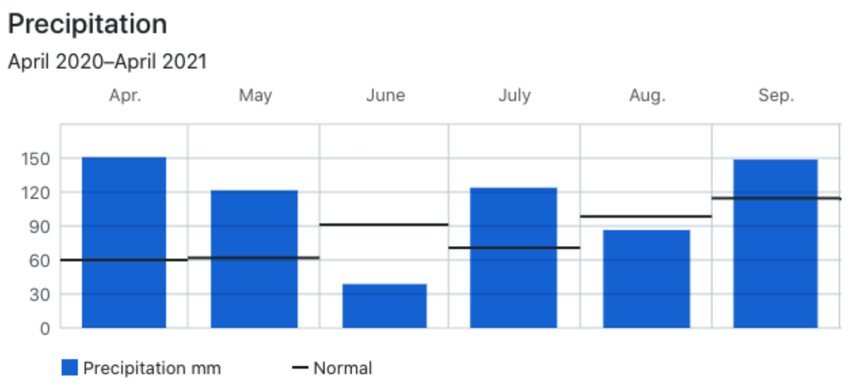


Figure 5.8: Example of Reference Lines Used in Bar Chart

Band Charts

Band charts are essentially enhanced line charts, with an added shaded area showing upper and lowers boundaries of an average value [38]. These charts provide the user with insight into the data, by adding context to the visualization.

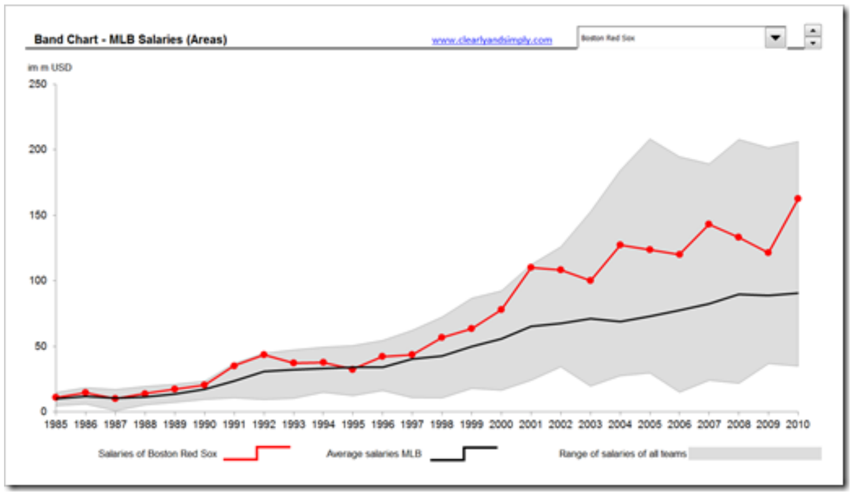


Figure 5.9: Example of Band Chart [38]

5.3.4 Axis Modification

Axis modification, especially modifications to the y-axis, can be used to manipulate the interpreter in various ways. Big intervals between the steps can make changes look small, while small intervals can make changes look big. Studies show that people tend to overlook changes in the chart axes more than changes to other factors, such as the color of graphical shapes or the describing text [37]. Consequently, it is important to be aware of how the axes are created when visualizing data.

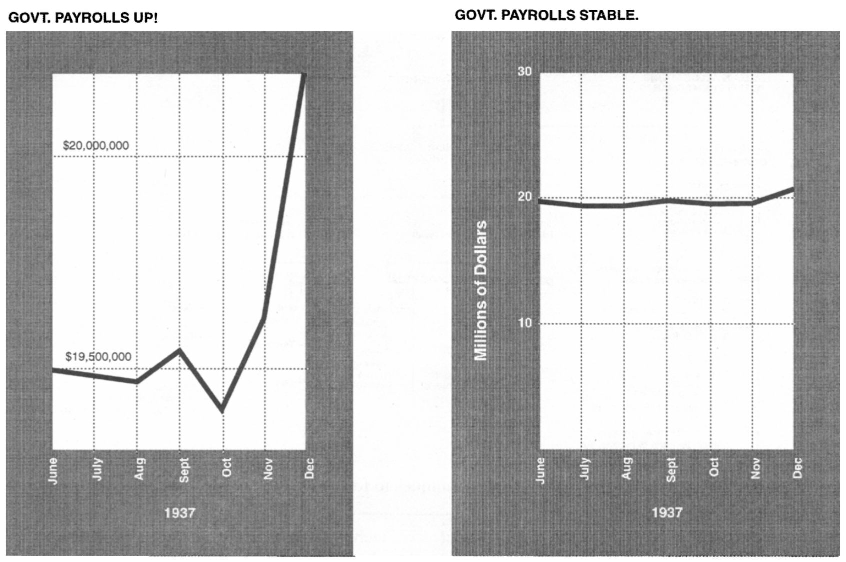


Figure 5.10: Two Ways of Visualizing the same Data [37]

5.4 Visualizing Hydrological Data

Data visualization has always been an important part of hydrology [33]. In the present day, modern computers are responsible for the majority of these visualizations. Such visualizations are made to present data in easily readable formats, as well as uncover trends and relationships between data in simple ways.

5.4.1 Commonly Used Chart Types

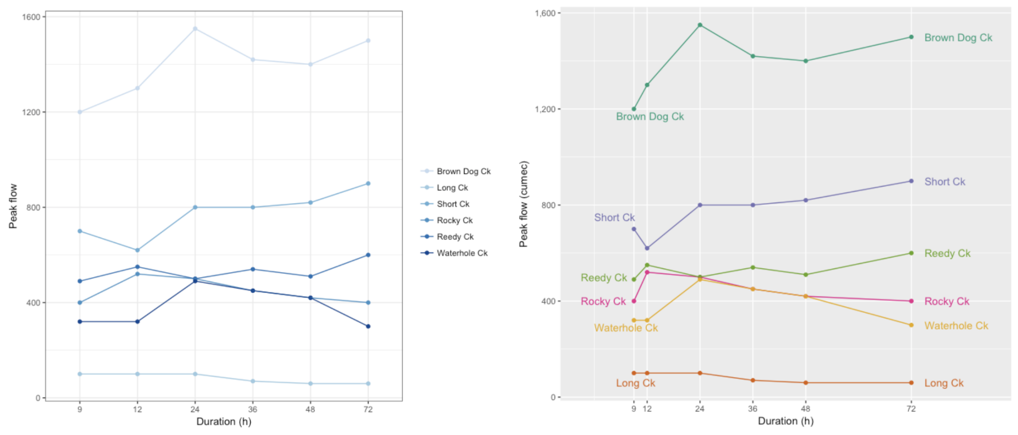
Commonly used chart types for visualizing hydrological data, include line charts, bar charts, and scatter charts. This section will explain how these chart types can be used to present hydrological values and how color can enhance these visualizations.

Line Chart

When visualizing hydrological data, line charts can be used to compare different data. However, some pitfalls should be avoided.

Some common pitfalls are including too many lines, not using distinct colors, and having inconsistencies in the axes [33]. Line charts with too many lines could potentially make the data difficult to interpret since they make it harder to differentiate the lines from each other. Not using distinct colors for each line could also make it difficult to unambiguously determine which color legend belongs to which line. For axes, it is important to keep the values and the spacing consistent to improve readability. If the same amount of space between two values on the x-axis represents two completely different time periods, this could easily lead to misinterpretations by the reader. Multiple of these pitfalls can be seen in the example in Figure 5.11a, while an improved version of the same chart can be seen in

Figure 5.11b. It is worth noting that the improved chart uses text directly on the lines, instead of using color legends which further emphasize which line represents which data.

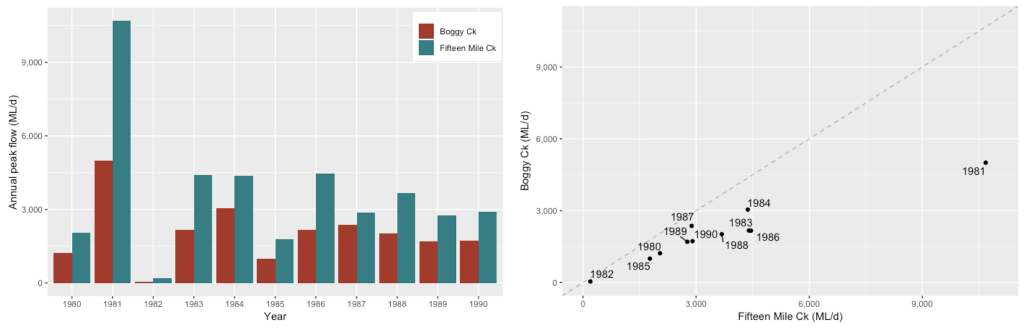


(a) Line Chart (b) Improved Line Chart

Figure 5.11: Hydrological Data Visualized in Line Charts [33]

Bar Chart and Scatter Chart

Similar to line charts, bar charts can also be used for data comparison. One example of use is when the user wants to compare data from two different sensors in the same time frame, for instance, a year. The same task can also be done using a scatter chart. One example of how hydrological data can be compared this way is peak values for sensors within a time frame. Figure 5.12 shows how this type of data can be visualized using a bar or scatter chart.



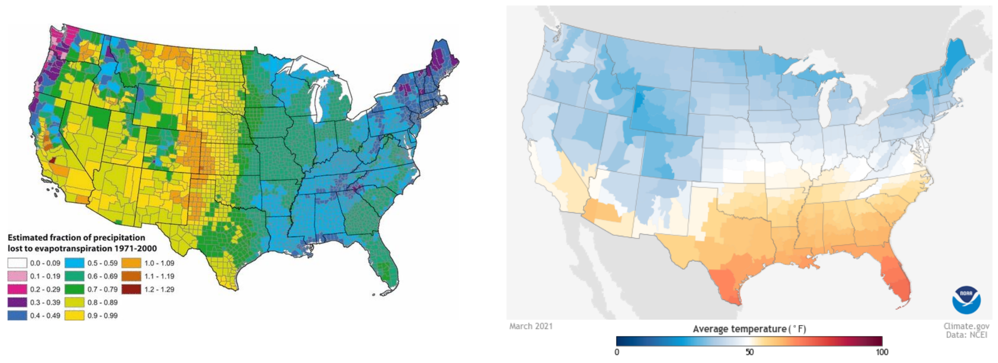
(a) Bar Chart (b) Scatter Chart

Figure 5.12: Comparing Peak Values Using Charts [33]

5.4.2 Color Scales

Choosing which colors should be used in each chart is an important decision, also when dealing with hydrological data [33]. One option for choosing colors is to go for a rainbow or a spectral color scale. This means that the color of the chart elements do not have any semantically-resonant meaning, but are rather chosen in a specific order. An example of a chart using one of these scales can be seen in Figure 5.13a. This way of choosing colors has been criticized for distorting, misleading, and confusing the chart reader by creating false boundaries between values [33].

Rainbow or spectral color scales are likely to increase error since there is no intuitive way of seeing if a value is high or low. People will generally expect a darker color to mean more. This means that it would be more helpful to choose a color scheme that goes from light to dark, or a more natural color order, for instance from blue to red to signal temperature changes. Figure 5.13 show two different map charts that show different types of data, with different color scales. Figure 5.13b uses blue and red coloring to show the temperatures across the United States.



(a) Spectral Color Scale [59] (b) Temperature Color Scale [7]

Figure 5.13: Map Charts with Different Color Scales

When visualizing values that can be either positive or negative (or above or below a specific value), it is helpful to opt for a diverging color palette where the color drastically changes at a certain value [33].

5.4.3 Less Commonly Used Chart Types

This section will briefly discuss some of the lesser-known chart types that can be used to visualize hydrological data. A visual representation of these chart types can be seen in Figure 5.14.

Waterfall Chart

Waterfall charts are a common approach when presenting financial statements to show how components contribute to an overall result, but they can also be used for hydrological purposes. One of the ways it can be used is to determine the driest and the wettest year of an area when it comes to precipitation [33]. An example of a waterfall chart used to visualize hydrological data can be seen in Figure 5.14a.

Cut and Stack Plot

When plotting long series of data in one chart, it can be difficult to fit all of the data in the chart without losing important information. One way of approaching this problem is to cut a longer graph element into segments of equal widths, and stack them on top of each other. This chart type can for instance be used to show the average daily flow in a river, to be able to spot longer dry periods [33]. An example of a cut and stack plot can be seen in Figure 5.14b.

Facet Plot

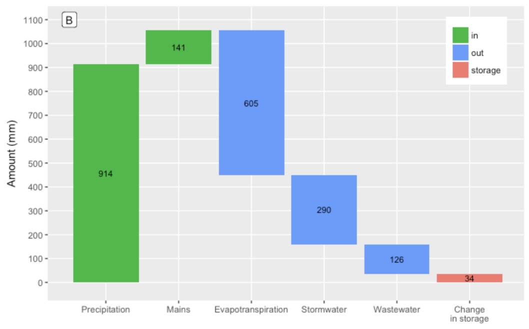
In a facet plot, each year can be displayed as a separate panel. Similar to the cut and stack plot, this chart type can be used to determine a river's dry periods [33]. Using a fixed y-axis allows for an easier comparison between the years. An example of a facet plot can be seen in Figure 5.14c.

Circular Plots

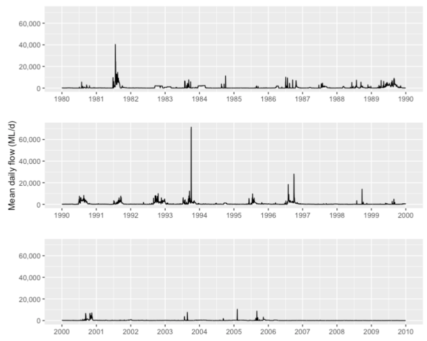
A circle plot, or a “clock face”, can be used to represent a year, where January 1st is placed at 12 o'clock and December 31st is placed at the end of the circle [33]. The area in between represents the rest of the year. These charts can be used to highlight events that occur annually, at certain parts of the year, or to show when the largest floods generally occur in an area. An example of a circular plot can be seen in Figure 5.14d.

Tile Plots

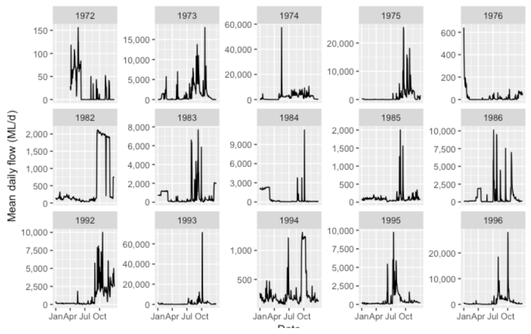
Tile plots can be used to visualize the impact that urbanization has on *flow*. Flow in this context means the water that infiltrates the ground and is discharged into stream channels, springs, or drainage pipes. These charts can highlight the times of the year when the flow is *high* (above average) or *low* (below average) in an area. Normally, these plots show that higher flows are clustered in the winter in rural areas, due to higher rainfall and reduced evaporation, while in urban areas, there are bursts of high flow throughout the entire year. This happens because the rainfall does not get absorbed by the ground, due to the *impervious* surfaces [33]. Impervious surfaces are surfaces that do not allow fluid to pass through. An example of a tile plot can be seen in Figure 5.14e.



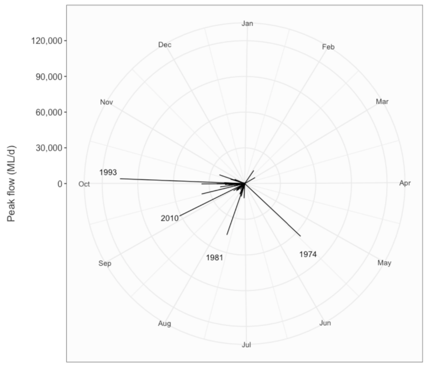
(a) Waterfall Chart



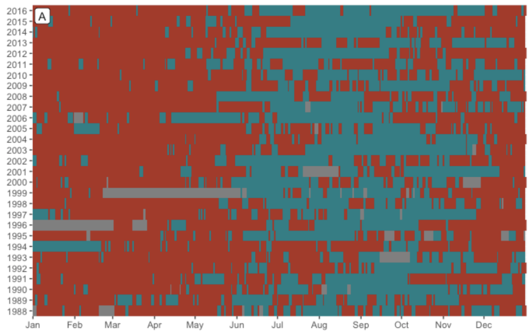
(b) Cut and Stack Chart



(c) Facet Plot



(d) Circular Plot



(e) Tile Plot

Figure 5.14: Less Frequently Used Chart Types [33]

Chapter 6

Relevant Work

This chapter discusses relevant work that has been used as inspiration throughout this project. The chosen relevant work was three applications: NVE Sildre (BETA), Yr, and VG Coronaviruset. These three were chosen either because the data they visualized were similar to the data provided by RMS, or because they had other relevant data visualization features.

Other relevant applications worth mentioning are Storm.no and AccuWeather.com. However, they are not discussed further in this thesis.

6.1 NVE Sildre (BETA)

NVE Sildre (BETA), henceforth referred to as Sildre, is a data visualization application created by NVE for visualizing hydrological data from various wastewater stations in Norway [42]. Its main goal is to deliver hydrological data to the public in a simple and user-friendly manner. While studying this application, it was still under BETA-testing, which meant that it was not yet a final solution, but was open for the audience with limited functionality.

NVE is an established directorate in Norway with 100 years of operation [41]. The initiative from NVE to create a service like Sildre therefore helps to validate the relevance of this project.

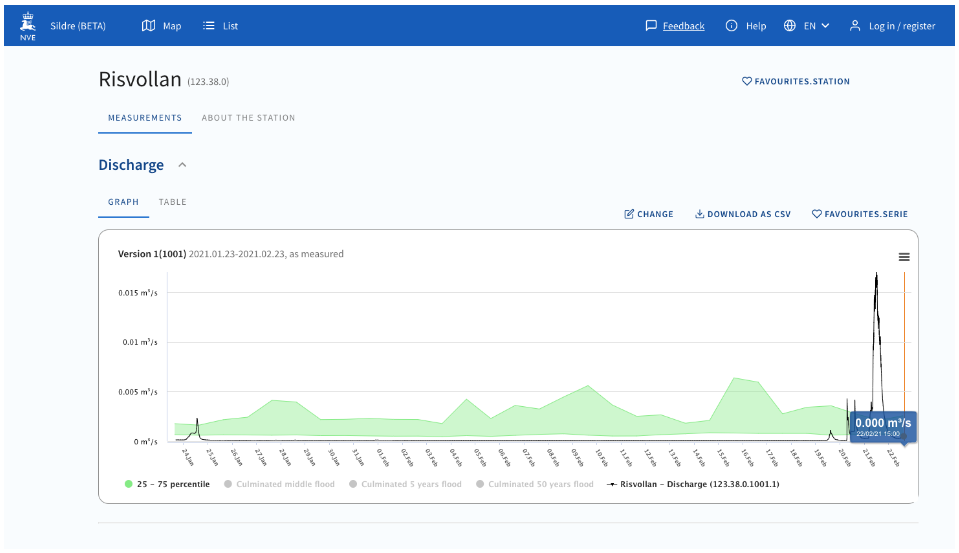


Figure 6.1: NVE Sildre

In this case, we will discuss how Sildre displays data from RMS.

6.1.1 Features

In this section, the features in Sildre that were considered the most relevant to this project will be discussed.

Displaying Time Series

Sildre displays data in the form of time series using charts and tables. For each data type, the user will have the option of seeing the data either in a chart or in a table. The time series consist of data gathered by NVE sensors and are displayed almost in real-time, which means that the data is displayed as soon as it is available, normally every hour. This feature can be seen as tabs in Figure 6.1.

Interactive charts

Several charts in Sildre allow for user interaction. They all have hover effects that give the user more detailed information about the data points in the chart, while some charts also allow for adding and removing data information from the chart. The chart shown in Figure 6.2 shows the hover effect on a data point.

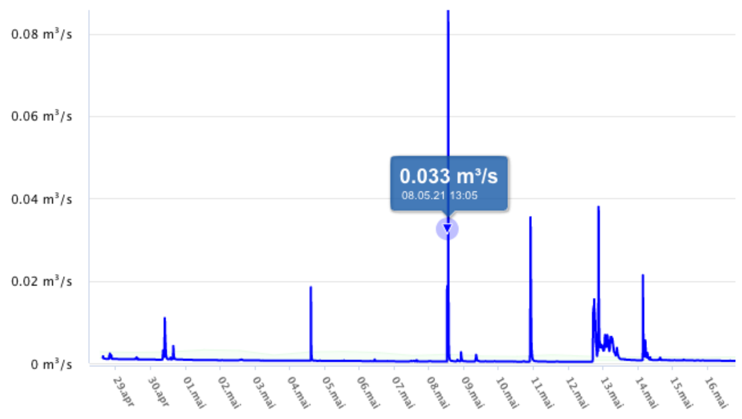


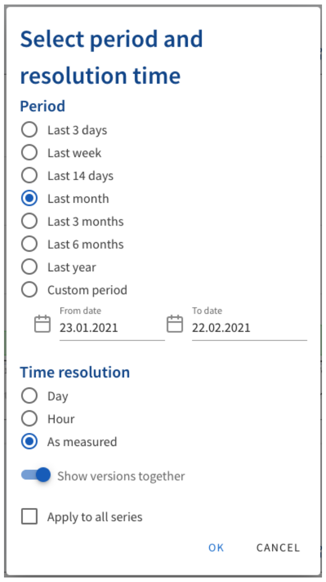
Figure 6.2: Sildre: Hover Effects

Customizing Charts

When displaying a chart, the user can click a "Change"-button, in the top right corner, to be able to customize the chart. This customization consists of selecting the time period and time resolution for the data to be displayed. The feature can be seen in Figure 6.3a.

Downloading Time Series

In Sildre, the user has the option of downloading the data displayed in either the chart or table as a CSV file. When downloading, the user has the option of downloading the entire time series of the data or choosing a custom time period, which can be seen in Figure 6.3b. The user can also select the time resolution of the data, either as *days*, *hours*, or *as measured*.



Select period and resolution time

Period

- Last 3 days
- Last week
- Last 14 days
- [✓ Last month]
- Last 3 months
- Last 6 months
- Last year
- Custom period

From date 23.01.2021 To date 22.02.2021

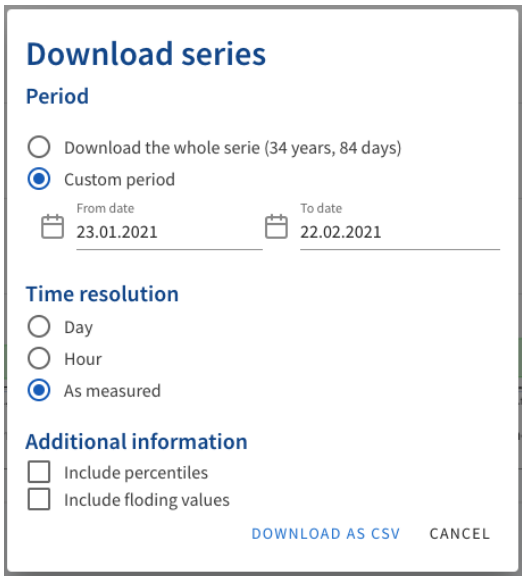
Time resolution

- Day
- Hour
- [✓ As measured]

[✓ Show versions together]

Apply to all series

(a) Change Chart



Download series

Period

- Download the whole serie (34 years, 84 days)
- [✓ Custom period]

From date 23.01.2021 To date 22.02.2021

Time resolution

- Day
- Hour
- [✓ As measured]

Additional information

- Include percentiles
- Include floding values

(b) Download Time Series as CSV

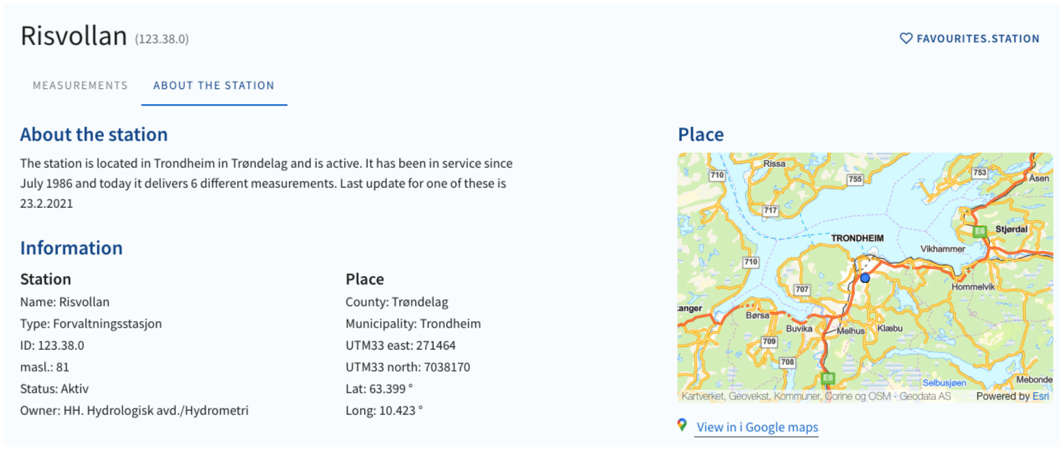
Figure 6.3: Sildre: Functionality

Export Charts as an Image

When displaying the time series in a chart, the user can choose to export and download the chart image. The file types available are PNG, PDF, SVG, and JPEG.

About the Station

Sildre includes a section with information about the Risvollan wastewater station. It states how long the station has been in service, how many measurement types it delivers, as well as when the last data update was. Additional information includes name, type, ID, location, and a map view where the station is pinned. This page can be seen in Figure 6.4.


Risvollan (123.38.0)

FAVOURITES.STATION

MEASUREMENTS ABOUT THE STATION

About the station

The station is located in Trondheim in Trøndelag and is active. It has been in service since July 1986 and today it delivers 6 different measurements. Last update for one of these is 23.2.2021

Information

Station	Place
Name: Risvollan	County: Trøndelag
Type: Forvaltningsstasjon	Municipality: Trondheim
ID: 123.38.0	UTM33 east: 271464
masl.: 81	UTM33 north: 7038170
Status: Aktiv	Lat: 63.399 °
Owner: HH. Hydrologisk avd./Hydrometri	Long: 10.423 °

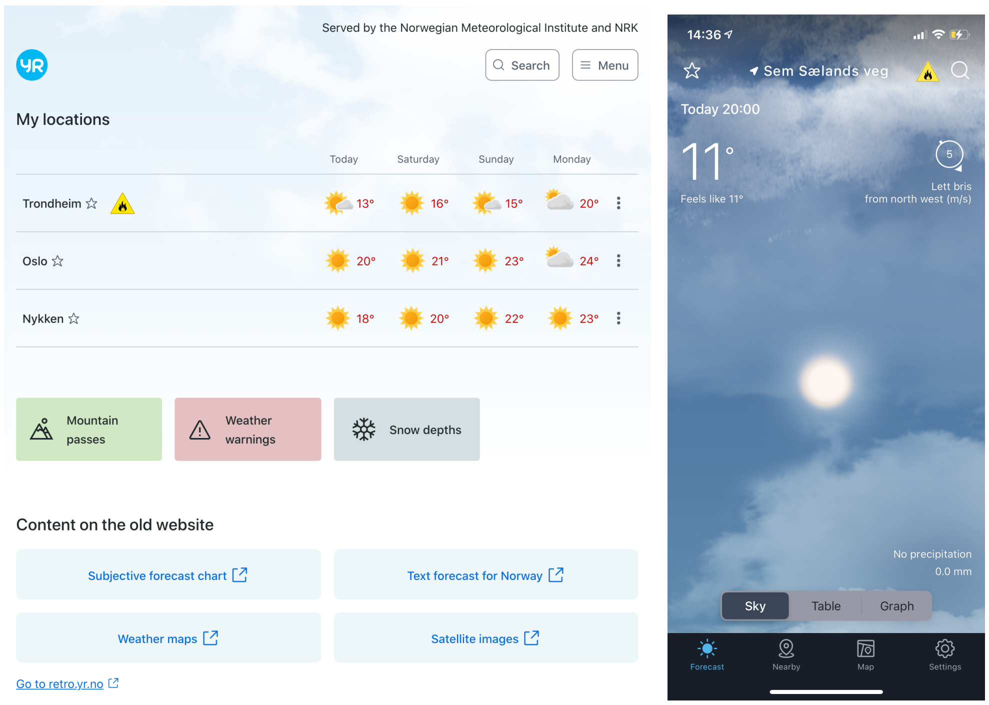
Place

View in i Google maps

Figure 6.4: Sildre: About the Station

6.2 Yr

Yr is a meteorological service provided by *The Norwegian Meteorological Institute and the Norwegian Broadcasting Corporation* (NRK). It provides both a website and a mobile application for displaying meteorological data to its users. According to themselves, Yr has millions of daily users and is one of the most popular weather services in the world [14]. The website version Yr.no is the version that will be discussed in this section, since it is the most relevant platform for this project, and will henceforth be referred to as Yr.



	Today	Saturday	Sunday	Monday
Trondheim ☆	13°	16°	15°	20°
Oslo ☆	20°	21°	23°	24°
Nykken ☆	18°	20°	22°	23°

(a) Yr.no Website [14] (b) Mobile Application

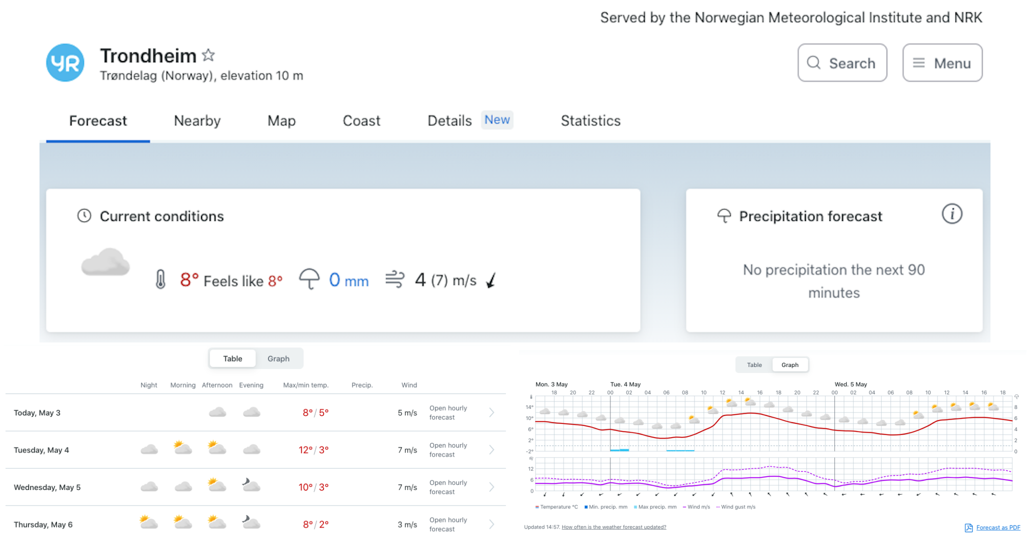
Figure 6.5: Yr

6.2.1 Feature

In this section, the functionalities of Yr that were considered the most relevant to this project will be briefly discussed.

Forecast

The main functionality that Yr provides, is its weather forecast. When choosing a particular area on the home page, the user is redirected to a forecast page for that area. This forecast page tells the user about the current weather conditions of the area, with details about the temperature, precipitation, and wind speed. Further, the details about the upcoming week are available. It is possible to view this information as a table or a chart. If the user wants to download the forecast as a PDF file, there is a link in the bottom right corner of the page, which can be seen in Figure 6.6b.


Served by the Norwegian Meteorological Institute and NRK

Trondheim ☆
Trøndelag (Norway), elevation 10 m

Current conditions

8° Feels like 8° 0 mm 4 (7) m/s

Precipitation forecast

No precipitation the next 90 minutes

	Night	Morning	Afternoon	Evening	Max/min temp.	Precip.	Wind	
Today, May 3			☁	☁	8° / 5°		5 m/s	Open hourly forecast
Tuesday, May 4	☁	⛅	⛅	☁	12° / 3°		7 m/s	Open hourly forecast
Wednesday, May 5	☁	☁	⛅	☾	10° / 3°		7 m/s	Open hourly forecast
Thursday, May 6	⛅	⛅	⛅	☾	8° / 2°		3 m/s	Open hourly forecast

Updated 14:57 How often is the weather forecast updated?

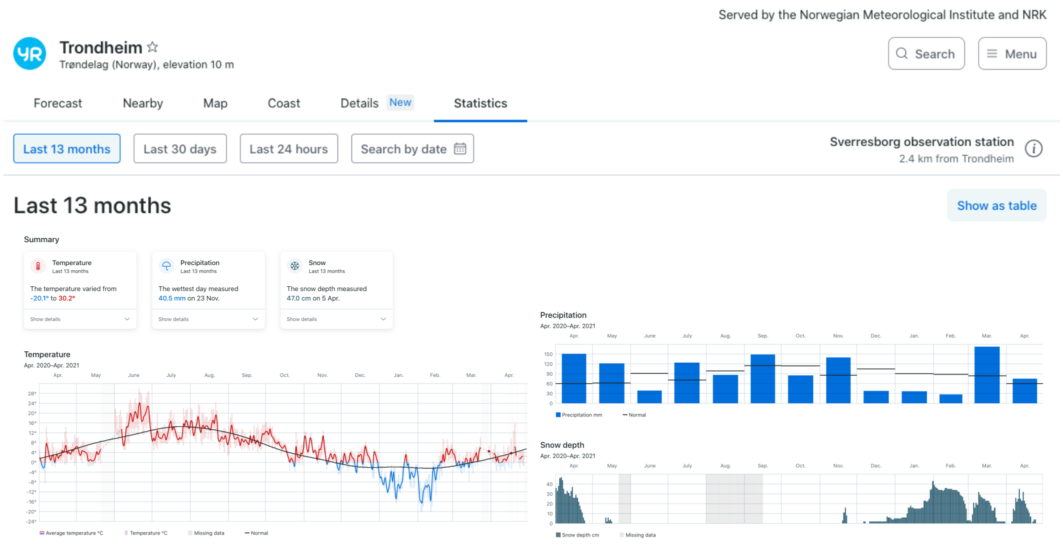
(a) Table (b) Chart

Figure 6.6: Yr: Forecast Page for Trondheim

Statistics

The statistics section is one of Yr's newer features. Yr launched a new version of its website in December 2020. This feature was first released under the name "Historical data", and lets you go back as far as 1930 to see detailed data and information about the weather at that time.

Yr's statistics page displays temperature, precipitation, and snowfall data from a specific time period. The user can either choose to see the data from the last year, month, 24 hours or choose a custom time frame. If the user wants to view the statistics data as a table, this is also possible by clicking the "Show as table"-button visible in 6.7b.



The temperature varied from -20.1° to 30.2°

The wettest day measured 40.5 mm on 23 Nov.

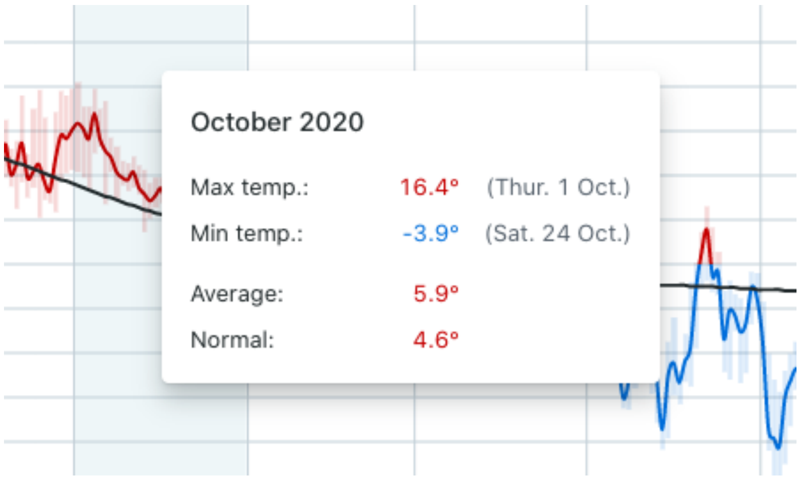
The snow depth measured 47.0 cm on 5 Apr.

(a) Summary and Temperature (b) Precipitation and Snow Fall

Figure 6.7: Yr: Statistics

Interactive Charts

All of the charts on Yr's website have the usual interactivity where the user gets more information about the chart while hovering over it. Yr's charts are divided into time periods which also have their own hover effects. This means that if you view a chart with e.g. temperature data from the last year, you can hover over each specific month to gain more insight into its temperature data. This includes maximum and minimum temperature, average, and normal values. An example of this can be seen in Figure 6.8.



October 2020

Max temp.:	16.4°	(Thur. 1 Oct.)
Min temp.:	-3.9°	(Sat. 24 Oct.)
Average:	5.9°	
Normal:	4.6°	

Figure 6.8: Yr: Chart Hover Effects

MET Weather API

There is no obvious way for users to download the hydrological data from Yr. However, there is a way for users to use it if they are more technically capable. Yr has a page that is created specifically for developers, which gives the visitor a brief tutorial of how to use the MET Weather API to be able to access the weather data that is used in Yr's services [76].

6.3 VG Coronaviruset

VG Coronaviruset is a visualization tool created by VG to visualize covid-19 data mainly from Norway, but also includes data from the rest of the world [72]. It is updated daily and has, since its first release in March 2020, kept adding new features. Henceforth, VG Coronaviruset will be referred to as Coronaviruset

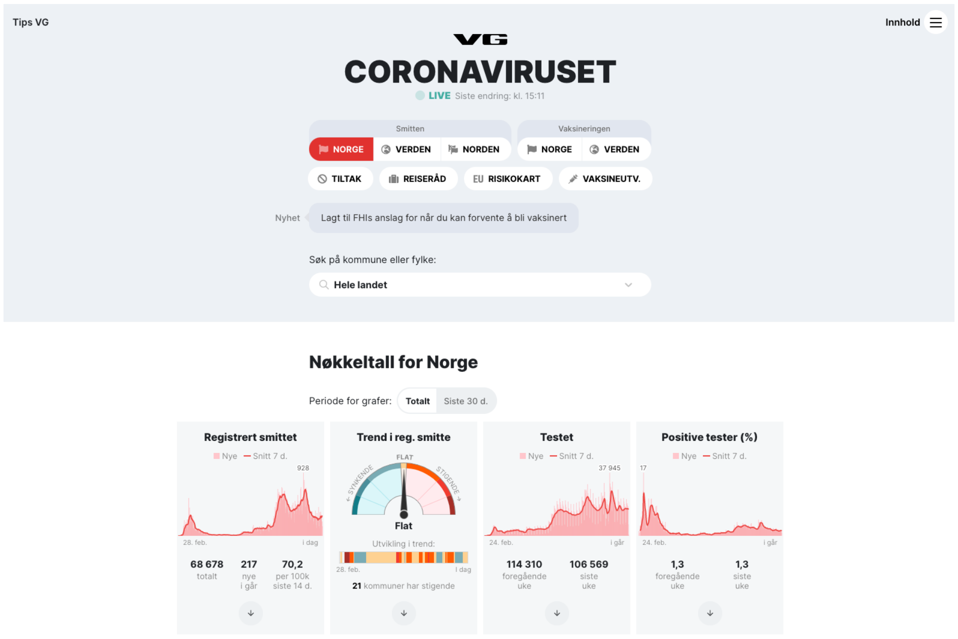


Figure 6.9: VG Coronaviruset

6.3.1 Features

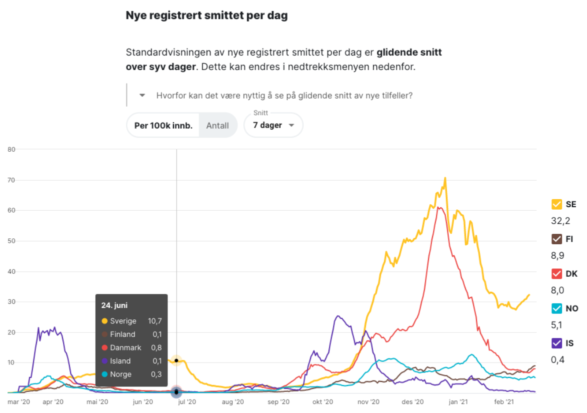
In this section, the functionalities of VG Coronaviruset that were considered the most relevant to this project will be briefly discussed.

Visualize Covid-19 Data

Coronaviruset displays a vast amount of covid-19 data from all over the world, but mainly focuses on data from Norway. The main types of charts used in the application are line charts, bar charts, area charts, and map charts. Coronaviruset, in some cases, also displays data in tables.

Interactive Charts

As previously mentioned, Coronaviruset displays a lot of its data in charts. Most of them include hover effects on the chart elements, that provide the user with more detailed information about a specific data point. Most often, this detailed information includes the number of infected or dead and the registration date of the recorded data. Figure 6.10a shows another type of hover effect, where the Nordic countries and the number of infected for each country are shown.



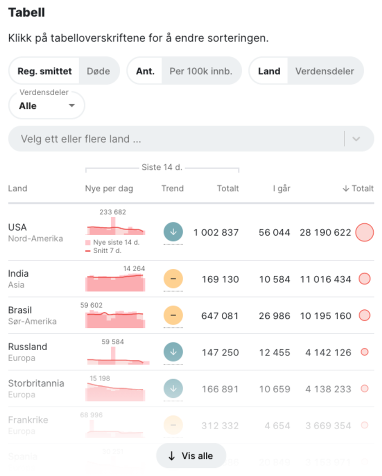
Nye registrert smittet per dag

Standardvisningen av nye registrert smittet per dag er **glidende snitt over syv dager**. Dette kan endres i nedtrekksmenyen nedenfor.

24. juni

● Sverige	10,7
● Finland	0,1
● Danmark	0,8
● Island	0,1
● Norge	0,3

✓ SE	32,2
✓ FI	8,9
✓ DK	8,0
✓ NO	5,1
✓ IS	0,4



Tabell

Klikk på tabelloverskriftene for å endre sorteringen.

	Siste 14 d.				
Land	Nye per dag	Trend	Totalt	I går	↓ Totalt
USA Nord-Amerika	233 682	↓	1 002 837	56 044	28 190 622
India Asia	14 264	–	169 130	10 584	11 016 434
Brasil Sør-Amerika	59 602	–	647 081	26 986	10 195 160
Russland Europa	59 584	↓	147 250	12 455	4 142 126
Storbritannia Europa	15 198	↓	166 891	10 659	4 138 233
Frankrike Europa	68 996	–	312 332	4 654	3 669 354

(a) Interactive Chart   (b) Countries Overview

Figure 6.10: VG Coronaviruset: Functionality

Filtering the Data

As seen in Figure 6.10, there is also the ability to filter the displayed data. This filtering functionality includes the ability to decide if you want to see new or collective data, as well as the ability to choose if you want to see the daily or weekly data. Most of the charts also include the option of seeing the data visualized with a logarithmic scale.

An additional filtering feature in the Coronavirus application is the ability to filter data depending on its country when looking at the world data, and filtering the data on municipalities, when looking at the data for Norway. An example of how this is done when can be seen in Figure 6.10b.

Chapter 7

Exploring and Testing Libraries

This chapter explains the hows and whys of exploring and testing libraries that could assist the implementation of Nykken.

7.1 Libraries

A library is a collection of functions, object files, and/or reusable code snippets, that is often used in software development. The library format depends on the language it was written in. Libraries can either be open source, which means that everyone can use them, or they can be protected by a closed source license.

7.1.1 Library Advantages

There are many advantages to using libraries in software development. This subsection will describe a couple of these main advantages.

Save Time

When building a website for visualizing data, it can be a huge time saver to use a framework or a library to create parts of the functionality, instead of building everything from scratch.

It is also possible to write a custom library for building the ideal visualizations. However, as there are many visualization libraries already available, the more efficient way is to look for an existing library that could realize the desired visualizations.

Avoid Reinventing the Wheel

It is generally unnecessary, and a waste of time and energy, to create something that has already been created multiple times over. This brings us to one of the most important advantages of using libraries: you do not have to reinvent the wheel every time you need a solution if it already exists.

7.1.2 Potential Downsides

Although there are many positive aspects to using libraries, there are also negative aspects that should be mentioned.

Dealing with Limitations

One single library will most likely not cover every possible scenario that is needed in an application. This means that there are limitations that have to be dealt with as a developer. Using third-party libraries means that one has to buy into their approach, which sometimes requires the developer to structure their code in a certain way.

Loss of Support

When using a library, there is a certain chance that the library might break or stop being updated. The developer therefore needs to put in the effort to keep up with the selected libraries and be quick to act if an update breaks important code in their application.

7.2 Selection Process

The process of exploring and testing libraries, to find the most suitable library, can be divided into seven steps, which are visualized in Figure 7.1.:

1. Find potential libraries
2. Define library criteria
3. Evaluate libraries based on the library criteria
4. Select top-rated libraries
5. Define testing criteria
6. Test the top-rated libraries based on the testing criteria
7. Select the most suitable library to use for implementation

How the first four steps of the process were completed is explained in detail in Appendix C.

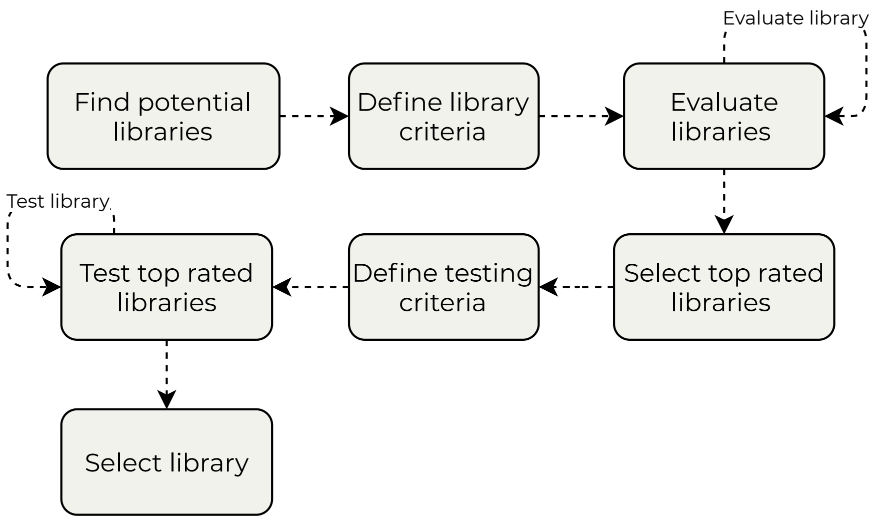


Figure 7.1: 7 Steps of the Library Testing Process

7.2.1 Libraries Selected for Testing

Based on the evaluation criteria presented in Appendix C.0.1, four libraries were selected for testing. These libraries were *Chart.js*, *D3.js*, *Nivo*, and *Victory*. They were selected based on their high evaluation scores, compared to other potential libraries.

7.3 Testing Libraries

Before testing the libraries, a testing application was set up using create-react-app [51] to display the testing results. Afterward, the testing criteria were defined and each selected library was tested based on these criteria.

7.3.1 Testing Criteria

This subsection will present the evaluation criteria that were used to evaluate the four potential libraries. An overview of the definitions of the criteria categories can be seen in Table 7.1.

Setting up and Getting Started

When testing a library, the process of setting up and getting started was evaluated. Each library got a setup score of either *Easy*, *Medium*, or *Hard*, depending on how easy it was to set up and get started.

Building a Chart

Since the main goal of the application was to visualize data with charts, it was essential to look at how straightforward the process of building a chart using the library was.

Documentation

The documentation was inspected more in-depth during the testing. How well the documentation helped the learning process of the library was evaluated.

Understandability

When defining the understandability scores of the libraries, the values given were either *Easy*, *Medium*, or *Hard*, depending on how easy the libraries were to understand.

Responsiveness

Responsiveness was important when testing the libraries. It was looked at how the charts responded to browser window resizing and how easy it was to fix any problems that would occur.

A library could either be categorized as having *Low*, *Medium*, or *High* responsiveness.

Customizability

Customizability was an important factor to evaluate for each library, especially when it came to customizing the charts themselves.

The libraries were either categorized as having *Low*, *Medium*, or *High* customizability.

Ability to Add Multiple Y-Axes

Each library had mentioned that they included the ability to add multiple y-axes. In this step, we evaluated how these additional axes were added to charts using the library. The ability of adding multiple y-axes was evaluated as being *Easy*, *Medium*, or *Hard.*

Export Possibilities

It could be helpful for users to be able to download image versions of charts. It was therefore decided to evaluate the export possibilities of the libraries.

During evaluation, each library would either fit into the *Yes* or the *No* category. ”No” meant that *export to image* was not a library functionality.

Integration with the Back-end API

It was evaluated how easy it was to display API data using the library. Each library integration with the API was evaluated as either *Easy*, *Medium*, or *Hard.*

Criteria	**Categories**					
	1	Definition	2	Definition	3	Definition
Setup	Easy	It is easy to set up and get started.	Medium	It is somewhat easy to set up and get started.	Hard	It is difficult to set up and get started.
Build a chart	Easy	Building a chart is easy and does not require a lot of insight.	Medium	Building a chart is somewhat easy, but requires some insight.	Hard	Building a chart is difficult and requires a lot of insight.
Under-stand-ability	Easy	It is easy to understand how the library works.	Medium	It is somewhat easy to understand how the library works.	Hard	It is difficult to understand how the library works.
Respon-siveness	High	The charts are responsive.	Medium	The charts are responsive, but with some limitations.	Low	The charts are not responsive.
Custom-izability	High	The charts are easily customizable.	Medium	The charts are somewhat customizable.	Low	The chart are not easily customizable.
Multiple Y-axes	Easy	It is easy to add multiple y-axes.	Medium	It is possible to add multiple y-axes, but there are some limitations.	Hard	It is difficult or impossible to add multiple y-axes.
Export image	Yes	The library functionality includes export to image.	—	—	No	The library functionality does not include export to image.
Integr-ation with API	Easy	It is easy to visualize data from the API, using the library.	Medium	It is somewhat easy to visualize data from the API, using the library.	Hard	It is difficult to visualize data from the API, using the library.

Table 7.1: Definitions of Criteria Categories

7.4 D3.js

D3.js is a JavaScript library that uses HTML, SVG, and CSS to visualize data. It allows the user to bind data to a *Document Object Model* (DOM) element with the possibility of applying data-driven transformations to the document. D3.js uses data to display various SVG elements in the DOM [9]. From this point onwards, D3.js will be referred to as D3.

7.4.1 Evaluation Criteria

This subsection describes the evaluation criteria that were used to evaluate each testing measure for D3.

Installation

After following a simple YouTube tutorial [35], installing D3 was no issue. A simple `npm install d3` in the terminal was all that was needed to install the library.

Building a Chart

To get to know D3 as a library, a free online course called *Datavis 2020*, ran by data visualization specialist Curran Kelleher [30], was followed. The course was created to give participants an introduction to data visualization, using React and D3. The D3 library was different from all the other libraries. Every D3 element was built from the ground up using only SVG elements. As a result of this, it was not possible to skip ahead to building a chart. First, a more thorough understanding of how SVG elements worked had to be established.

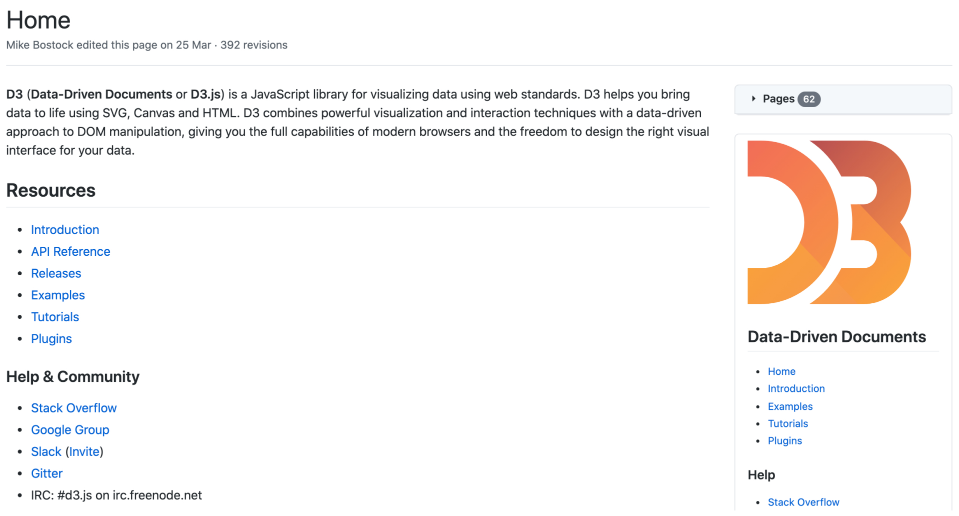
After understanding the basic concepts of SVGs, it was possible to continue learning about the CSV parsing of data, loading data in React, creating axes with SVG lines and text, and eventually create charts with SVG elements such as circles, lines, and rectangles.

Customization

With D3, there was no real limit to the customization. Since there were close to no abstractions to the library, nothing was predefined about the way a chart should be presented. Everything about the charts could additionally be tweaked in CSS to achieve the desired visualizations.

Documentation

The D3 documentation was located on the D3 GitHub wiki [9]. This wiki included many resources that could be useful when building charts using the library. It also linked to examples, but these were fully-fledged examples other people had created using the library, so D3 did not provide a simple "LineChart example" in its own documentation. To find this, the user would have to find external sources, such as the Datavis 2020 course.



Home

Mike Bostock edited this page on 25 Mar · 392 revisions

D3 (**Data-Driven Documents** or **D3.js**) is a JavaScript library for visualizing data using web standards. D3 helps you bring data to life using SVG, Canvas and HTML. D3 combines powerful visualization and interaction techniques with a data-driven approach to DOM manipulation, giving you the full capabilities of modern browsers and the freedom to design the right visual interface for your data.

Resources

- Introduction
- API Reference
- Releases
- Examples
- Tutorials
- Plugins

Help & Community

- Stack Overflow
- Google Group
- Slack (Invite)
- Gitter
- IRC: #d3.js on irc.freenode.net

Figure 7.2: D3: Documentation

Ability to Add Multiple Y-Axes

Since there were no real limitations to customizing charts in D3, there were also no limitations to the number of axes that could be added to a chart. The axes had to be built from the bottom up with SVG line and text elements, but after following a few examples from Datavis 2020, the axes eventually became less complicated to create.

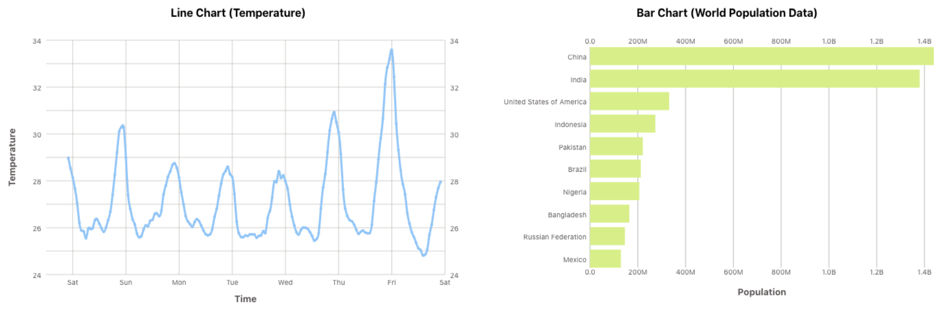


Figure 7.3: D3: Multiple Axes

Export Possibilities

D3's documentation did not provide any direct way of exporting charts to different file types. To do this, other external third-party libraries would have to be used.

7.4.2 Positive and Negative Aspects

This subsection presents the positive and negative aspects that were noted during the testing process of the D3 library.

Positive Aspects

- The developer had full customizability and control.
- The developer could decide where the abstractions would go, especially when using React in conjunction with D3.
- D3 is not limited to hidden implementations of other libraries.
- A lot of supporting material available.
- D3's documentation provided a lot of helpful built-in functions.
- It was possible to create a LineChart component using React and D3 together, that can later be reused for all line chart instances, even if they are different from each other.

Negative Aspects

- A significant amount of time was needed to learn the library and confidently use it outside of tutorials.
- Creating a chart is not as straightforward as with other libraries, since every part of the chart has to be created from scratch.
- The developer could potentially end up with a significant amount of code in comparison to other libraries, hence the lack of abstractions.

7.5 Chart.js

Chart.js is a small open-source JavaScript library. It uses canvas elements from HTML5 to render its chart types: *Bar*, *Line*, *Area*, *Pie*, *Bubble*, *Radar*, *Polar*, and *Scatter plot.*

7.5.1 Evaluation Criteria

Setting up and Getting Started

The installation process was straightforward. The official Chart.js installation guide was used to set up and get started [5].

Building a Chart

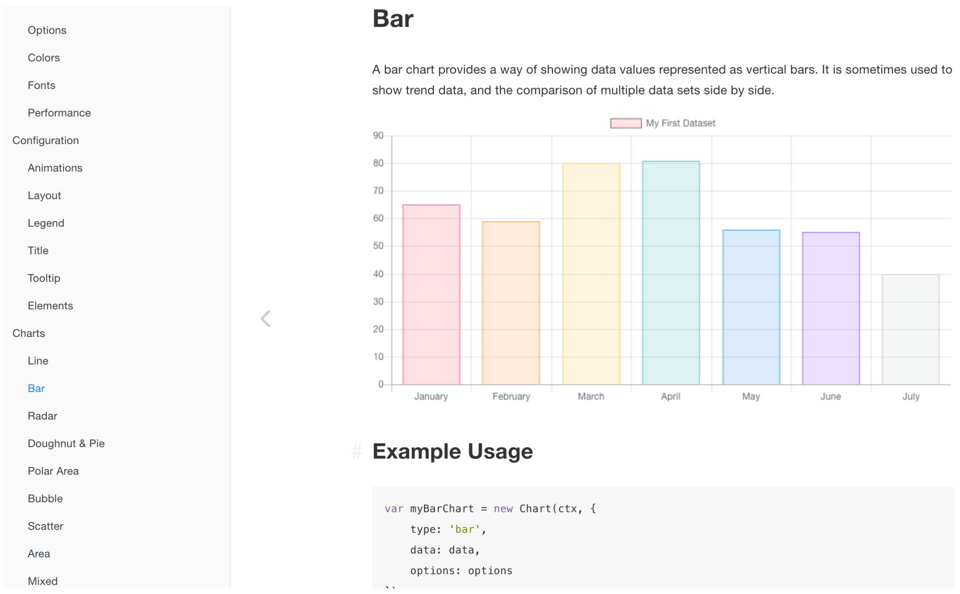
Building a chart with Chart.js was simple. Following a video tutorial [68], a chart was built in minutes without any problems. It was intuitive how the charts were structured since the documentation explained everything in a simple, easy-to-understand manner.

Documentation

Chart.js had a documentation that was easy to navigate. It had a search bar, as well as many sections and categories to explore [5].

The categories were neatly organized with subcategories that gathered related information together. The "Chart Types" category had a lot of subcategories of different chart types.

Each of these subcategories had its own examples and instructions of how to set them up, as well as presenting all of the possible parameters each chart had. An excerpt of how the "Bar" page looked can be seen in Figure 7.4.



Bar

A bar chart provides a way of showing data values represented as vertical bars. It is sometimes used to show trend data, and the comparison of multiple data sets side by side.

Example Usage

```
var myBarChart = new Chart(ctx, {
    type: 'bar',
    data: data,
    options: options
```

Figure 7.4: Chart.js: Documentation

After using the Chart.js documentation throughout the testing process of the library, its quality was evaluated to be good. An additional advantage was that it was possible to search the documentation freely.

Ability to Add Multiple Y-Axes

Chart.js had the possibility of adding as many x- and y-axes as the developer could want. Adding these axes was both simple and straightforward to do. The result of following the instructions for multiple axes can be seen in Figure 7.5.

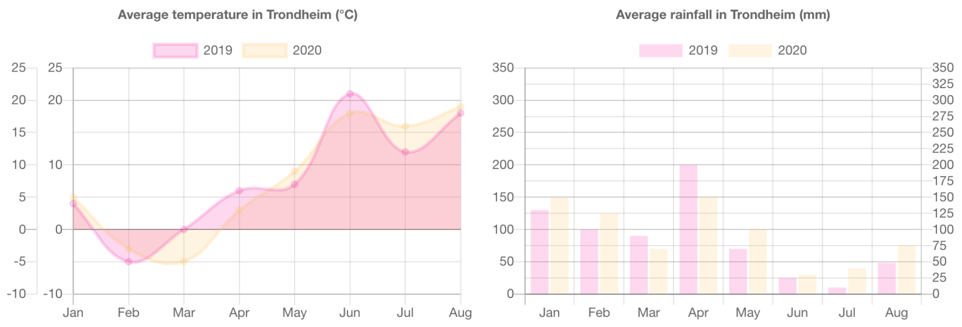


Figure 7.5: Chart.js: Multiple Axes

Export Possibilities

In Chart.js' own API, there was a function called `.toBase64Image()`, which was described to return a PNG data URL of the image on the canvas [4]. However, it ended up being difficult to figure out how to use this function. After multiple attempts of trying to make it work, it ended up not being possible after all, either due to poor documentation of the function or the function itself being flawed.

7.5.2 Positive and Negative Aspects

Positive Aspects

- It was easy to understand the data structure of a chart.
- The charts had an aesthetically pleasing look.
- Customizing the colors of the charts was a straightforward process.
- It was easy to add interactive color legends that could be used to show and hide the chart elements.

Negative Aspects

- For the charts to behave responsively, the page had to be refreshed every time the browser window was resized.

7.6 Victory

Victory is described as an ecosystem of React and React Native chart components for building interactive data visualizations [73]. Victory describes itself as a tool for creating charts, without sacrificing flexibility.

7.6.1 Evaluation Criteria

Installation

The Victory documentation presented the developer with a tutorial to help them get started with the framework. The first step was to set up a basic React project. It was possible to choose between doing it on your own or cloning the example project.

Building a Chart

The easiest part of Victory was to build the initial chart. After a simple import of the React component VictoryBar, a bar chart was created without having to input any dummy data.

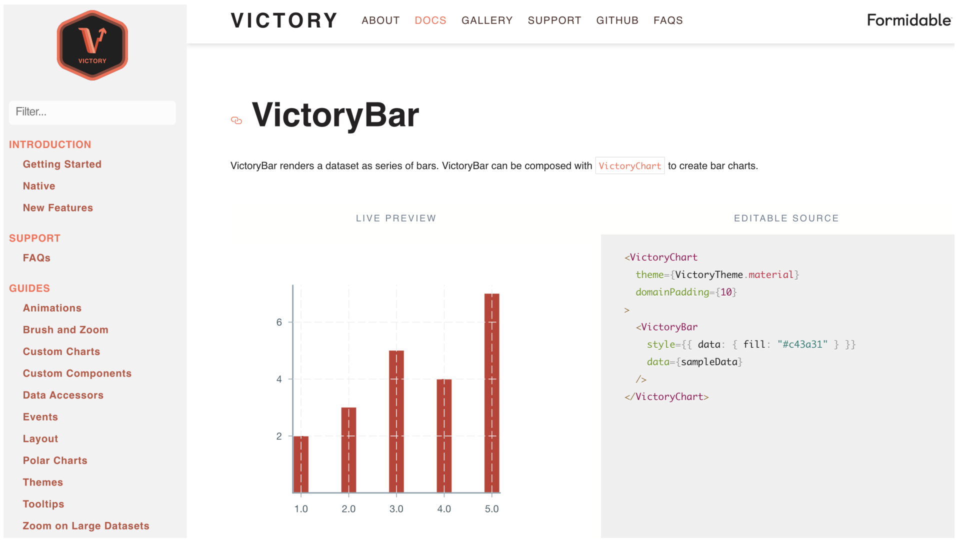
Customization

It was easy to add themes to charts since the VictoryChart component had a property called theme. This property accepted a VictoryTheme component as input, which could include as many theme values as desired, including custom colors and fonts. The general language and structure of the tutorial made it easy to follow.

Documentation

Victory's documentation was very well organized. It was divided into several sections and subsections. In addition to the usual "Charts" section, it also included a "Guides" section that explained how to alter default components, as well as creating custom ones.

Most of the functionality that was documented also included an example chart with the associated code. An example of this can be seen in Figure 7.6. One thing that was missed in the documentation was how to customize the additional axes to place them more effectively.



VictoryBar

VictoryBar renders a dataset as series of bars. VictoryBar can be composed with `VictoryChart` to create bar charts.

```
<VictoryChart
  theme={VictoryTheme.material}
  domainPadding={10}
>
  <VictoryBar
    style={{ data: { fill: "#c43a31" } }}
    data={sampleData}
  />
</VictoryChart>
```

Figure 7.6: Victory: Documentation

Ability to Add Multiple Y-Axes

To add multiple axes, multiple components had to be added. Adding a right axis needed an x offset, which was a bit tricky to set since the exact number that would place the axis in the desired spot had to be found through trial and error. Flipping the text when creating the new axis was difficult as well, something that made it almost impossible to avoid overlapping between the text and the axis. An example of this can be seen in Figure 7.7.

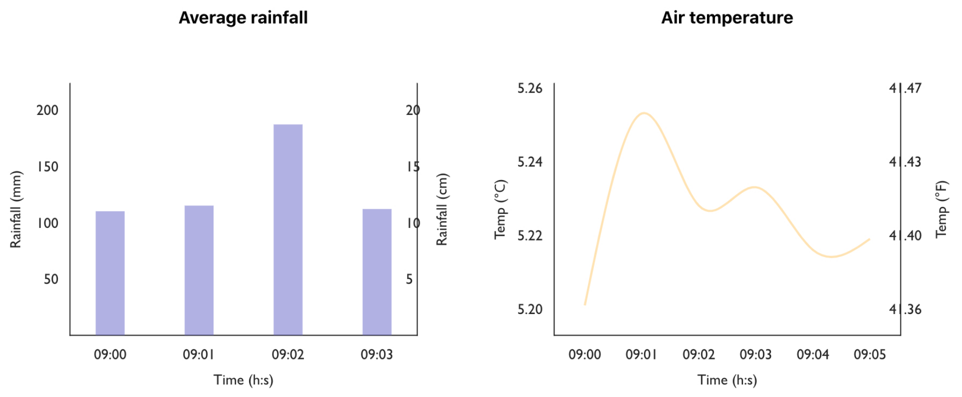


Figure 7.7: Victory: Multiple Axes

Export Possibilities

Victory did not include any possibilities for exporting the charts to image or PDF files. As mentioned, there were ways to do this outside of the library itself, which lead us to believe that this was not a deal-breaker when it came to choosing which library to use in the project.

7.6.2 Positive and Negative Aspects

Positive Aspects

- It was intuitive and easy to set up and get started with Victory.
- It was possible to render a chart even without dummy data.
- The axes were automatically generated when adding a VictoryChart component, which meant that min and max values did not have to be manually set.
- Victory had good documentation quality with the ability to easily search through everything.
- The documentation included a helpful tutorial for getting started.
- It was easy to customize the axes with many ticks and tick formats.

Negative Aspects

- It was difficult to place text on additional axes without having experiencing overlaps between axes and text.

7.7 Nivo

Nivo is a library consisting of React components that are built on top of D3. It markets itself as being highly customizable and able to create both SVG, HTML, and Canvas-based charts [40].

7.7.1 Evaluation Criteria

Installation

There was no installation guide on the official Nivo website. Instead, a Medium article was used to get started with the library [62]. There did not seem to be an easy way of installing all of Nivo's functionality at once, for example, @nivo/line and @nivo/bar had to be installed separately.

Building a Chart

It took more time to build a chart with Nivo than it did with for example Chart.js. This was mostly due to the immense documentation, and that the charts were supposed to have two separate files called config.js and data.js for them to render. When this was ultimately understood, the rest of the process of building a chart went smoother.

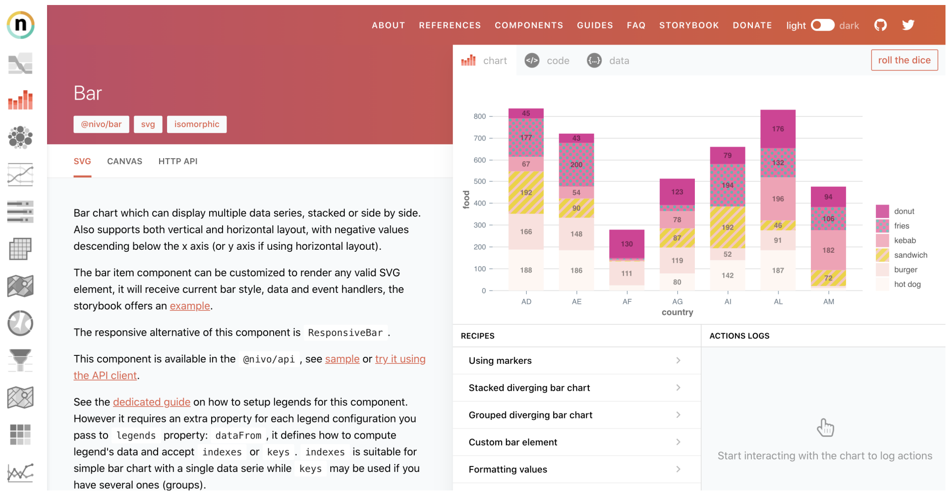
Customization

With Nivo, the customization possibilities were many. However, it was not as straightforward as the documentation first made it appear. When trying to use the same customization as the documentation had used in its own examples, the outcome was not the same. This was ultimately possible to fix, but the customization process was not as intuitive as first expected.

Documentation

Nivo had immense documentation. Although it seemed overwhelming at first, the documentation itself was structured in a good way. Nearly all variables of each chart type had an interactive example, which made it possible to view changes directly in the documentation. It was also noted which variables were required and which ones were optional.

One downside to the documentation was the fact that all of the sections in the menu only had image icons, and no supplementing text to explain what section described what chart type. This led to a lot of trial and error when trying to find what we were looking for in the documentation. There was also no obvious way of searching the documentation. An excerpt from the ”Bar” page in the Nivo documentation can be seen in Figure 7.8.



Bar

@nivo/bar svg isomorphic

SVG CANVAS HTTP API

Bar chart which can display multiple data series, stacked or side by side. Also supports both vertical and horizontal layout, with negative values descending below the x axis (or y axis if using horizontal layout).

The bar item component can be customized to render any valid SVG element, it will receive current bar style, data and event handlers, the storybook offers an example.

The responsive alternative of this component is `ResponsiveBar` .

This component is available in the `@nivo/api` , see sample or try it using the API client.

See the dedicated guide on how to setup legends for this component. However it requires an extra property for each legend configuration you pass to `legends` property: `dataFrom` , it defines how to compute legend's data and accept `indexes` or `keys` . `indexes` is suitable for simple bar chart with a single data serie while `keys` may be used if you have several ones (groups).

RECIPES

- Using markers
- Stacked diverging bar chart
- Grouped diverging bar chart
- Custom bar element
- Formatting values

ACTIONS LOGS

Start interacting with the chart to log actions

Figure 7.8: Nivo: Documentation

Ability to Add Multiple Y-Axes

In Nivo, it was possible to add a top, bottom, right, and left axis. Each added axis was an object with six optional variables: *enable*, *tickSize*, *tickPadding*, *tickRotation*, *legend*, and *legendOffset*. The ticks represented a collection of lines and numbers that would go along the axes. The legends would show which colors corresponded to what value in the chart. Both variables were optional to show. The result of adding multiple axes can be seen in Figure 7.9.

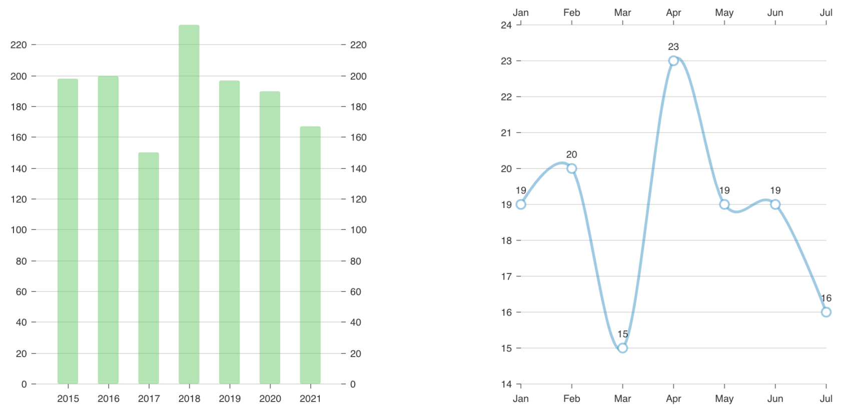


Figure 7.9: Nivo: Multiple Axes

Export Possibilities

Nivo did not include the possibility of exporting charts to images. However, after some research, multiple other ways of doing it with a third-party library were discovered. Since it was not included in the specific library that was tested, it was not furthered explored at this stage.

7.7.2 Positive and Negative Aspects

Positive Aspects

- Nivo was highly customizable when it came to color, patterns, fonts, and sizes.
- It was easy to add multiple axes.
- Multiple optional variables were available for each chart type.
- Interactive documentation.

Negative Aspects

- Non-searchable documentation with only icons to present categories.
- Lack of supporting material.
- It was difficult to manually adjust the axis range.
- Strange behavior when resizing browser window. The chart would randomly take up the entire width of the browser window.
- Unclear structure of input data.

7.8 Results of Library Testing

During the library testing process, the results were noted down and eventually put into a table that can be seen in Table 7.2.

Criteria	**Library**			
	Chart.js	Nivo	Victory	D3.js
Setup	Easy	Medium	Easy	Easy
Build a chart	Easy	Medium	Easy	Medium
Understandability	Easy	Easy	Easy	Medium
Responsiveness	Medium	Medium	High	Medium
Customizability	Medium	Medium	Medium	High
Multiple Y-axes	Easy	Medium	Hard	Easy
Export image	No	No	No	No
Integration with API	Easy	Hard	Easy	Easy

Table 7.2: Results from Library Testing

7.9 Reflection

During and after the library testing, there was a lot of information to absorb and evaluate. As the testing process went on, the testing criteria that were more or less relevant to each library became clearer.

The most valuable information that was learned during the library testing process included:

- Each library had its own blueprint of how a data set should be structured, for its work with the library components.
  - Example: Most libraries accepted a JavaScript object, but the names of the values varied (some had to have x- and y-values, while others could use names such as ”temperature” and ”timestamp”).
- The documentation quality heavily influenced how easy each library was to test.
- When a library was more difficult to test, it also had weaker results, which made the library in question less convenient to use.
- Gaining testing experience meant that it became clearer what was important to test and not for each library. As a result, the more testing that was done, the more thorough the testing became.
- Similar to the previous point, the more testing was done, the more clear it became that it was important to write down how every step of the testing process went. Hence, the last libraries ended up having more notes than the first libraries that were tested.
- It worked better to write down the experiences before filling in the table, and not the other way around.
  - The first two libraries had the table as the primary documentation of testing results, but as the testing continued and became more thorough, writing everything down first made more sense.
- Before testing, *export possibilities* was chosen as a testing criteria. However, it was found that the majority of the libraries did not mention this as a part of their documentation, and in the rare case that they did, the functionality was flawed. Due to further investigation, it was found that other people had made this possible by using other third-party libraries or stand-alone plug-ins. Therefore, it was decided to not put as much focus on this criteria when evaluating the libraries.

7.10 Deciding on a Library

After testing the four libraries that were ranked the highest, the last step in the library testing process, was to decide on the library that was going to be used in the project.

The decision was ultimately made based on the results from the library testing that can be seen in Table 7.2, as well as looking deeper into the criteria that were prioritized: *Customizability*, *relevancy*, and *availability of supporting material*.

7.10.1 Deciding on D3

In the end, D3 was chosen as the library that was going to be used in the project.

There were three main reasons why D3 ended up being chosen as the preferred visualization library:

- **Customizable**
  D3's visualizations were built by simple SVG elements. In other words, D3 had very few abstractions, which made it especially customizable.
- **Relevant**
  D3 had the most amount of GitHub stars out of all of the tested frameworks. In addition to this, D3 was regularly updated.
- **Supporting material**
  The vast amount of supporting material available about D3, meant that if we were to get stuck on a task, there would most likely exist a solution online.

Even though it was not as straightforward to build and understand the D3 charts right away, the three previously mentioned reasons made D3 stand out as the best option compared to the other libraries. In the end, D3 would be the library that would be used to develop the final solution, *Nykken*.

Part IV

Nykken

This part presents Nykken, a web application for visualizing and downloading hydrological data from RMS. Firstly, the part will describe the design and development process from the very first idea and concept, the prototyping phase, and lastly the final concept.

Chapter 8

Conceptual Model

This chapter presents the conceptual model, which was based on the results from the interviews and prototype testing. It will describe the metaphor, mapping, concepts, and the relations between the concepts. Lastly, a minor change between the initial and final conceptual model will be discussed.

8.1 Dashboard Metaphor

The application can be looked at as a dashboard of graphs, each with the option of being changed with controls.

A dashboard is originally a term used to describe the panel inside of a car, which provides information about the current state of the car's main functions, as well as some of the controls to manage certain aspects of the car. In this project, the dashboard represents an overview of the sensor charts, while the controls represent each sensor and its data.

8.2 Concepts

The concepts in the updated conceptual model were:

- Sensor data - Sensor data retrieved from the API
- Time frame - A start and end time
- Sensor chart - A visualization of sensor data in a time frame
- Chart group - A group of sensor charts with different sensor data within the same time frame
- Custom chart - A combined sensor chart with multiple sensor data within the same time frame
- Sensor measurements - All of the added sensor charts displayed in one place
- Chart groups - All of the chart groups displayed in one place

- Custom charts - All of the custom charts displayed in one place
- Dashboard - A display of sensor measurements, chart groups, and custom charts

Concept	**Attributes**	**Operations**
Sensor data	Sensor ID, time frame, granularity, measurements	Get sensor data
Time frame	Start date time, end date time	Set time frame
Sensor chart	Sensor data	Edit time frame, change granularity, download as CSV, download as PNG
Chart group	(Multiple) sensor data, time frame, group name	Edit time frame, change granularity, delete group
Custom chart	Sensor data (multiple), time frame, custom chart name	Edit time frame, change granularity, delete custom chart
Sensor measurements	Sensor charts	Add/remove sensor chart
Chart groups	Chart groups	—
Custom charts	Custom charts	—
Dashboard	Sensor measurements, chart groups, custom charts	Add chart group, add custom charts

Table 8.1: Concepts

8.3 Relations

Concept relations show how the concepts are connected. Existing relations between the concepts are visualized in Figure 8.1 using an *Entity-Relationship Diagram* (ERD) where each concept is represented as an entity and each relation is represented as a relationship.

Concept	**Relations**
Sensor data	Contains time frame, granularity, and measurements
Time frame	Contains start date time, end date time
Sensor chart	Visualizes sensor data in a time frame
Chart group	Contains multiple sensor charts in the same time frame
Custom chart	Visualizes multiple sensor data in the same time frame
Sensor measurements	Contains all selected sensor charts
Chart groups	Contains all created chart groups
Custom charts	Contains all created custom charts
Dashboard	Contains sensor measurements, chart groups and custom charts

Table 8.2: Concept Relations

8.4 Changes from Initial Conceptual Model

In the initial conceptual model, groups were referred to as *sections*. During the prototype testing, users found this wording disorienting. To clarify the concept, sections were renamed to *groups*.

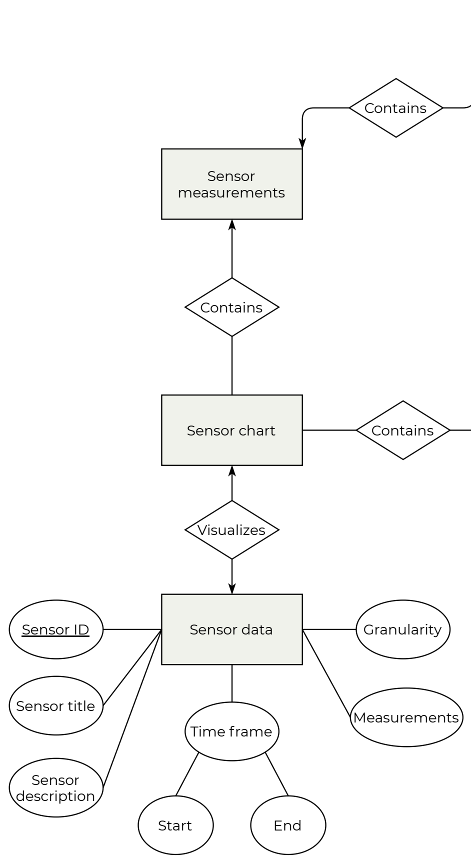


Figure 8.1: ERD of Concept Relations

Chapter 9

Personas

Personas are fictive characters developed based on research, to represent the users of the application [56]. They exist to help designers and developers understand what their users' needs and expectations are, as well as their experiences.

Using the insight from the interviews, which is described in Subsection 4.3.1, the three main target groups within the WWSE-group emerged. *PhD Candidates* using the data in their work, *professors* at the Department of Civil and Environmental Engineering, and lastly *technicians* monitoring the sensors at RMS.

9.1 Persona: PhD Candidate

Figure 9.1: PhD Candidate: Osan [50]

Name: Osan
Age: 24
Occupation: PhD Candidate at the Department of Civil and Environmental Engineering
Motivation: Wants to plot measurement data into his models.
Pain points:

- Has to ask professors or other sources to access the data from the RMS.
- Data files have to be cleaned after receiving them.

Quote: "Downloading the data should be easy, but it's very often a challenge"

9.2 Persona: Professor

Figure 9.2: Professor: Ingvild [50]

Name: Ingvild
Age: 47
Occupation: Professor at the Department of Civil and Environmental Engineering
Motivation: Wants to show students hydrological data in an effective manner.
Pain points:

- No easy way to quickly visualize data from RMS.
- Has to provide students with a lot of data.

Quote: "When students need data, I have to sort through 500 text files to find the data they want"

9.3 Persona: Technician

Figure 9.3: Technician: Jonas [70]

Name: Jonas
Age: 32
Occupation: Chief Engineer at the Department of Civil and Environmental Engineering
Motivation: Wants to check the status of the sensors in the measurement station.
Pain points:

- Not easy to determine if the data is faulty or not.
- Tedious process to visualize data from RMS.

Quote: "It's not always that easy to see if the data that comes in makes sense"

Chapter 10

Scenarios

User scenarios are developed by designers to show how user groups could use the application in a realistic setting [17]. Scenarios are used to understand the motivation and needs of the user, and to show how the user could have their needs met by using the developed application. These scenarios are developed with our three personas in mind.

10.1 PhD Candidate Scenario

Osan, the PhD Candidate, wants to plot sensor data into one of his models for a paper that he has due. Osan usually uses a specific program to do the modeling. To get the data for his models, he visits Nykken, the RMS website, to download the data as a CSV file. To do this, Osan adds the temperature sensor to his dashboard. He then selects a time frame that is suitable for his model and a chart with this data appears. Osan is not particularly interested in the chart but wants to download the data displayed in the chart as a CSV file. He therefore clicks the "Download as CSV"-button by the chart and uses it in his models.

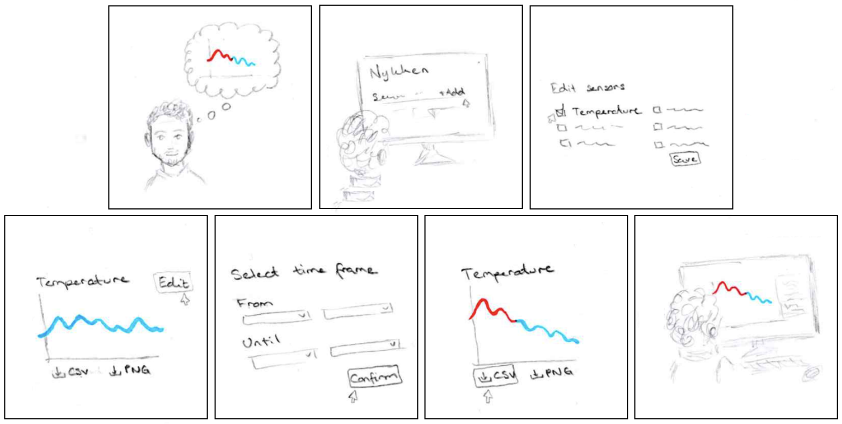


Figure 10.1: PhD Candidate Scenario

10.2 Professor Scenario

Ingvild, the professor, is going to have a lecture for some first-year students studying Civil and Environmental Engineering. The lecture is about hydrological data. In this particular lecture, Ingvild wants to show the students the correlation between rainfall, humidity, and wastewater. To do this, she needs a specific chart that displays this data in a combined chart. She decides to visit Nykken to create the chart for her lecture. She clicks the "Create custom chart"-button and selects the three data types she wants to display and edits the time frame to show the variation during a specific day. After she had created the custom chart, Ingvild clicks the "Download as a PNG"-button. She then adds the PNG image to her Powerpoint presentation.

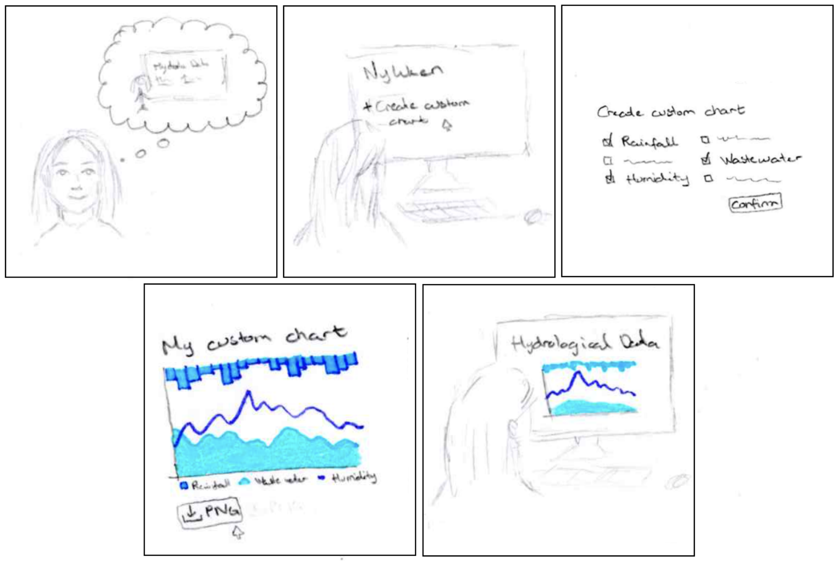


Figure 10.2: Professor Scenario

10.3 Technician Scenario

Jonas, the technician, installed a new rainfall gauge at RMS yesterday. Today, he wants to control the sensor data to see if the sensor was successfully installed. Jonas enters Nykken and sees the charts that he has previously added to his dashboard. One of them is rainfall. The rainfall chart shows data from the last hour. Jonas decides to change the time frame to show the last 24 hours. Jonas has written down when he installed the new sensor and finds in it the chart. The rainfall data from this specific time period shows negative values. Since rainfall data cannot be negative, Jonas interprets this as there being something wrong with the rainfall gauge that was installed and decides to control it.

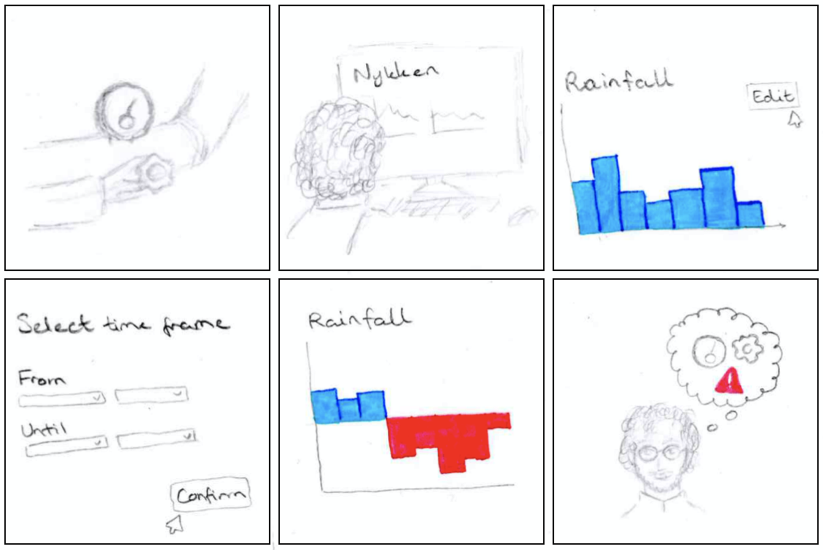


Figure 10.3: Technician Scenario

Chapter 11

Requirements and User Stories

This chapter will present the application's functional and non-functional requirements, and its user stories. These were defined based on the conceptual model, personas, and the scenarios, which in turn were created based on the user interviews.

11.1 Functional Requirements

In this section, the functional requirements for the application are presented in Table 11.1. Functional requirements describe what the software must offer to its users [13] [20]. A functional requirement can describe the function of the software system itself, or one of its components.

11.2 Non-Functional Requirements

Non-functional requirements involve any other requirements that describe the application in attributes such as *portability*, *usability*, *security*, *performance*, and *availability* [13]. The non-functional requirements are presented in Table 11.2.

ID	Description	Priority
F1	The application shall have a dashboard	High
F2	The application shall visualize sensor data	High
F3	The application shall provide data download	High
F4	The application shall allow users to combine multiple sensor data in a visualization	High
F5	The application shall allow users to choose which sensor data they want to see visualized	High
F6	The application shall allow users to select which time frame they want in the visualizations	High
F7	The application shall allow users to download image versions of sensor data visualizations	Medium
F8	The application shall give users information about existing sensors	Medium
F9	The application shall allow users to remove added visualizations from the dashboard	Medium
F10	The application shall provide the latest data available	Medium
F11	The application shall provide the date for when the data was last updated	Low

Table 11.1: Functional Requirements

ID	Description	Attribute	Priority
N1	The application shall function and be available on modern browsers	Portability	High
N2	The application shall be easy and intuitive to use, with minimal uncertainty	Usability	High
N3	The application's main functionality shall be easily accessible, with minimal distractions or clutter	Usability	High
N4	The application users shall find the application secure, without being unsure about personal information leaking without permission	Security	High
N5	The application shall have short response time when retrieving data for visualizations	Performance	Medium
N6	The application shall allow for extension of the API, for example adding more sensors	Scalability	Medium
N7	The application shall be available and not have much down time	Availability	Medium

Table 11.2: Non-Functional Requirements

11.3 User Stories

After the requirements were defined, user stories were developed to ensure that the requirements were met during the development process. User stories are informal, general explanations of software features that are written from the end user's perspective [54]. Their purpose is to articulate how software features can provide value to the stakeholder.

The developed user stories are shown in Table 11.4. These user stories were defined by *ID*, *priority*, *name*, *description*, *user type*, and *size estimate*.

11.3.1 Definitions

The user story priorities were first marked as *High* (red), *Medium* (orange/yellow), or *Low* (green). Afterward, they were ranked in order and put into a backlog. The size estimates were set based on how much time it was anticipated that they would take to implement in the application. These estimates were categorized into *Small*, *Medium*, and *Large*. A closer look at the time estimate and priority definitions can be seen in Table 11.3.

Priorities	
🟥 High	Must be implemented
🟨 Medium	Should be implemented
🟩 Low	Can be implemented

Time Estimates	
Small	0-2 hours
Medium	2-6 hours
Large	6+ hours

Table 11.3: Definitions: Priorities and Size Estimates

11.3.2 User Stories Table

ID	**P**	**Name**	**Description**	**User type**	**Size est.**
1	1	Data in chart	As a user I want to be able to see sensor data in a chart.	Student, professor	Medium
2	4	Add/remove sensors	As a user I want to be able to choose which sensor charts I want to see on my dashboard.	Student, professor	Medium
3	3	Custom time frames	As a user I want to be able to select custom time frames for each chart.	Student, professor	Medium
4	8	Chart zoom	As a user I want to be able to zoom in and out of the chart.	Student, professor	Medium
5	5	Hover details	As a user I want to be able to hover to see more detailed information about data points.	Student	Small
6	2	Download CSV	As a user I want to be able to download the data in a CSV-file.	Student, professor	Medium
7	9	Download PNG	As a user I want to be able to download the chart as a PNG-file.	Student, professor	Medium
8	7	Custom chart	As a user I want to be able to see multiple sensor data in one custom chart.	Student	Large
9	10	Name custom chart	As a user I want to be able to give my custom chart a name.	Student	Small
10	11	Customize colors	As a user I want to be able to customize the colors of the graphs inside of my custom chart.	Student, professor	Small
11	16	Receive updates	As a user I want to be able to receive updates on selected charts.	Student	Large
12	6	Create group	As a user I want to be able to group charts together and set a time series for the whole group.	Student	Medium
13	12	Latest API update	As a user I want to be able to see when the API was updated last.	Student	Small
14	13	Sensor information	As a user I want to be able to access the sensor information.	Professor	Small
15	14	Data status	As a user I want to be able to see the status of the data (faulty or good).	Technician	Medium
16	15	Discrepancy notification	As a user I want to be notified when discrepancies occur.	Technician	Medium

Table 11.4: User Stories

Chapter 12

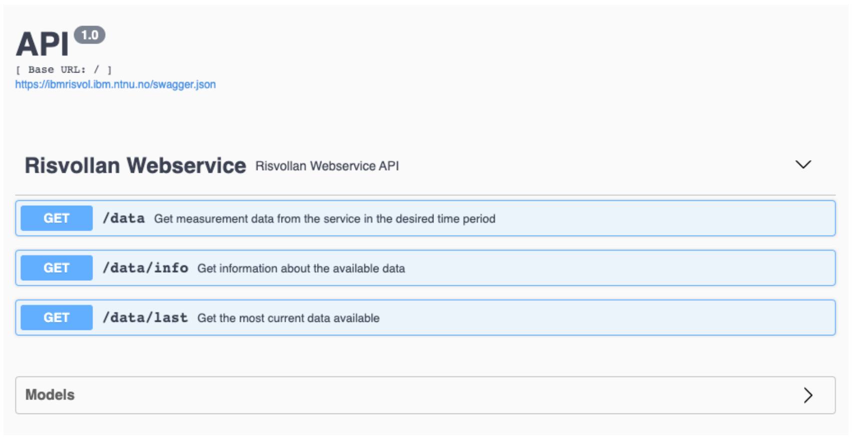
Software Architecture

This chapter will explain the software architecture of Nykken. Firstly, it will present the API used for providing data to Nykken will be described. Afterward, it will be explained how this influenced the choice of software architecture for the application.

12.1 Risvollan Webservice API

Nykken was developed and based on the functionality made available by the Risvollan Webservice API available at `https://ibmrisvol.ibm.ntnu.no/`, which will henceforth be referred to as the API. The API was developed in 2020/2021 by the main stakeholder. Since the API was the main source of data fueling the web application, it also had to be a main component in the software architecture.

One of the core purposes of Nykken was to display data and information from the API. This data and information would be displayed in various forms, such as in charts and groups. For each chart and group created by the user, a client request would have to be sent to the API through the application UI. When receiving requests, the API would provide data from one of its service components to the UI. This could for example be a `/data/info` request, which would give the user information about a sensor, or it could be a `/data` request, which would give the user a series of data measurements from a specific sensor in a specific time frame.


API 1.0
[Base URL: /]
https://ibmrisvol.ibm.ntnu.no/swagger.json

Risvollan Webservice Risvollan Webservice API

GET /data Get measurement data from the service in the desired time period

GET /data/info Get information about the available data

GET /data/last Get the most current data available

Figure 12.1: Risvollan Webservice API

12.2 Nykken Architecture

Since the API provides the user with various types of services, a *microservices architecture pattern* would best describe this type of architecture. There are many ways to implement a microservices architecture pattern, but two of the most popular ways are the *API REST-based* topology and the *application REST-based* topology.

API REST-based

The API REST-based topology has proven itself useful for web applications that expose small, self-contained individual serviced through an API [55]. In the API REST-based topology, the users typically access these services through a REST-based interface implemented through a separately deployed web-based API layer.

Application REST-based

While there are many similarities between the API REST-based topology and the application REST-based topology, the main difference is that in the application REST-based one, the client requests are received through a traditional web application, instead of just the API layer [55]. The service components associated with this type of topology are usually larger and do not represent as fine-grained services as in the API REST-based topology.

The software architecture of Nykken is based on a combination between the API REST-based topology and the application REST-based topology. Nykken works as the UI layer that connects the users with the API. The architecture is visualized in Figure 12.2.

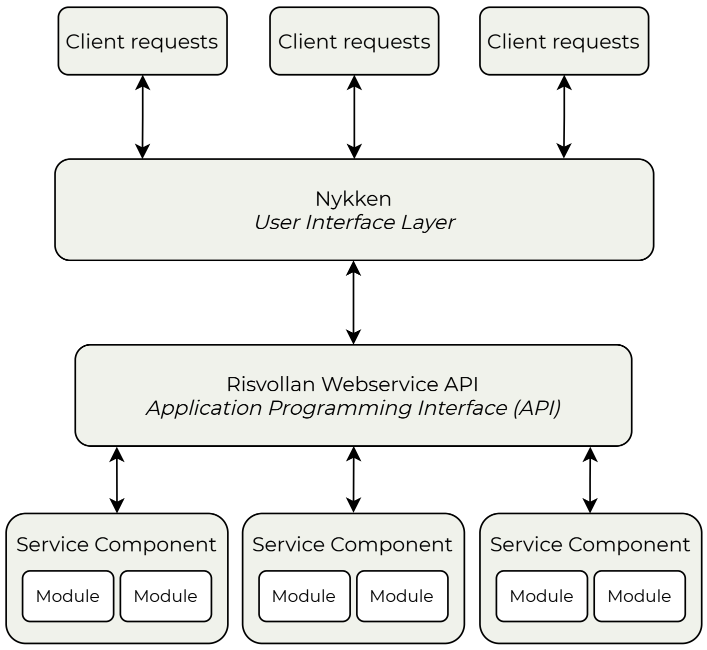


Figure 12.2: Nykken: Software Architecture

Chapter 13

Technologies

In this chapter, we will present the most important technologies used to develop Nykken: Figma, GitLab, React, and D3.js. In addition to this, the reasons for choosing these technologies will be discussed. Not all of the technologies used will be presented, but all of them are listed in Appendix D.

(a) Figma (b) GitLab (c) React (d) D3.js

Figure 13.1: Technologies

13.1 Figma

Figma was the technology used to create prototypes during the prototyping phase of this project. Figma is a web-based user interface design application that can be used for wireframing websites, UX design, interaction prototyping, creating social media posts among other things [16].

The main reasons for choosing Figma as the prototyping technology were:

- **Collaborative**
  Figma is excellent for collaborating since it is possible to share Figma files with other users. When a file is shared between users, it can be edited by multiple people simultaneously. This was one of the main reasons why this technology was chosen.
- **Web-based**
  Figma is a web-based application, which means that it will run on any OS that has a web browser. Because of this, all updates made to the Figma file are automatically

updated for all users that have access to the file. That way, there is no need for sending files back and forth every time there is a change in the prototype.

- **Reusable components**
  When creating a button in Figma, you usually want to use this button on multiple parts of your application. Figma makes this easy with reusable components. If a color change is needed on all buttons, the user only has to change the color of the button component, which will update the color on all instances of this component. When developing a prototype, this feature could potentially save you a lot of time.

13.2 GitLab

GitLab is a DevOps tool that lets users collaborate and manage their repositories [26]. It provides features like issue-tracking and continuous integration. In addition to these reasons, it was recommended by the supervisor.

In this project, GitLab was used for managing the Nykken repository and for its issue-tracking during the development of Nykken.

When it was time to deploy Nykken, the hosting service Netlify was used. Netlify worked best when using GitHub as the code source. For this reason, the GitLab repository was cloned to GitHub to connect the project with the hosting service.

13.3 D3.js

D3.js was used to create the visualizations of the sensor data in Nykken [9]. The main reasons for choosing D3.js were discussed in Subsection 7.10.1.

13.4 React

In addition to using D3, we decided on using React to build our application. React is an open-source JavaScript library, built by Facebook, for building user interfaces [52].

The main reasons for choosing React in combination with D3 were:

- **Reusable components**
  React is built in a way, that lets the developer easily create reusable components. These components can have various properties to make many variations of the same component, for instance, a graph with different data properties. In a project like this, React would therefore be incredibly useful and efficient to use.
- **Widely used**
  React has a wide user base with over 6.4 million GitHub users as of May 2021. This means that there is a lot of user support available, as well as frequent updates.
- **Compact and easy to understand**
  React does not provide an API of a million different functions but rather consists of a small set of functions that describe the core concepts of React. This makes it less confusing and overwhelming to use together with D3, which has a larger amount of functions.

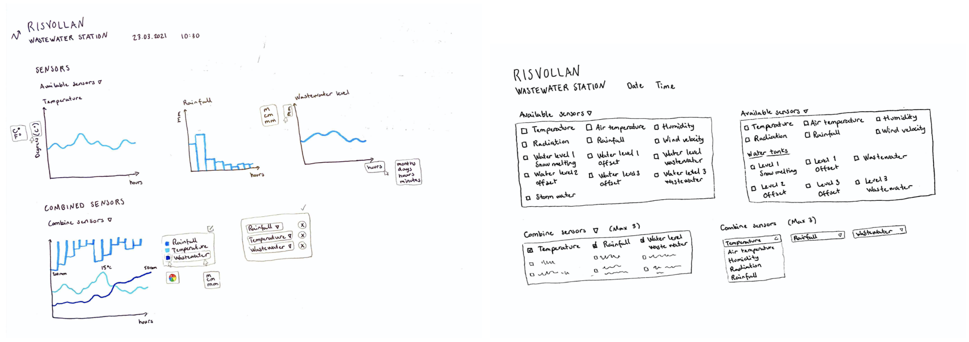
Chapter 14

Prototyping

Before settling on the final Nykken concept, there were multiple iterations of prototyping. This chapter presents the prototyping process, with early sketches and a description of the final Figma prototype.

14.1 Early Sketches

After interviewing the potential users, a lot of the important points that were mentioned were translated into paper sketches. These were early ideas of solutions that could potentially meet the user needs.



(a) Dashboard (b) Select Sensors

Figure 14.1: Prototype: Early Sketches

14.2 Discarding the Paper Prototype

A standard procedure during a design and development process is paper prototyping. Due to the covid-19 situation at the time of prototyping, it was not possible to arrange a physical meeting with the potential users. As a result of this, we decided not to create a paper prototype in the first iteration, but rather use the early sketches as inspiration for a high fidelity prototype.

14.3 High Fidelity Prototype

The high-fidelity prototypes were created using Figma and were based on user needs retrieved from the interviews. This included selecting which sensor data they wanted to see, combining multiple sensor data together, and customizing the charts.

14.3.1 Dashboard

The dashboard is where the user has all of their sensor measurements, custom charts, and sections in one place. From the dashboard, the user has three main options of action: *Edit dashboard*, *create section*, and *create custom chart*.

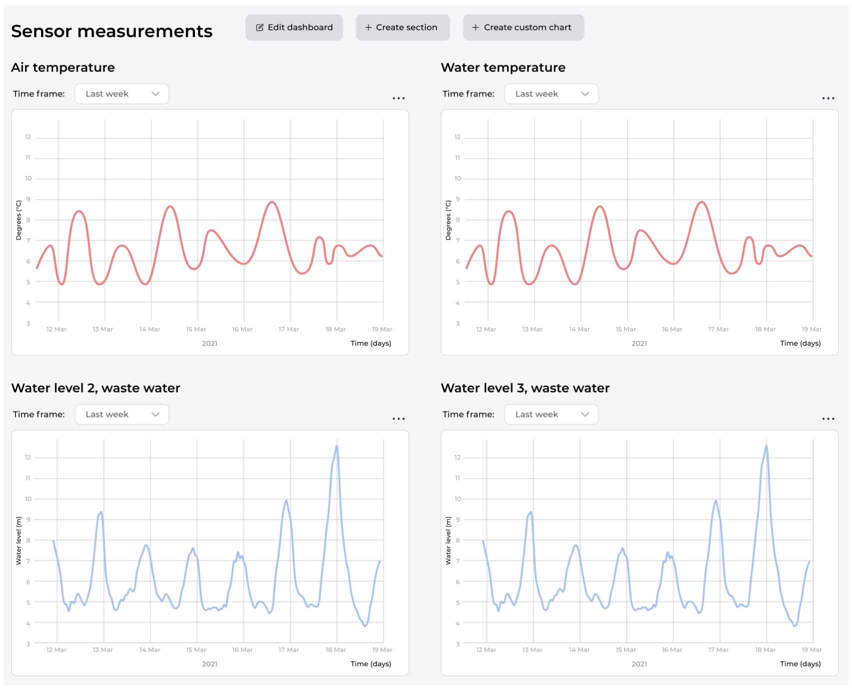
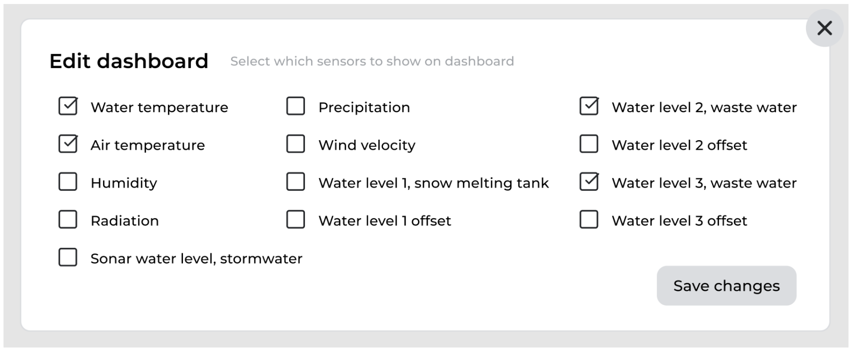


Figure 14.2: Prototype: Dashboard

14.3.2 Edit Dashboard

If the user wants to add or remove a sensor chart from the sensor measurements, they can click the ”Edit dashboard”-button from the dashboard. When they do, the modal shown in Figure 14.3 pops up. In this modal, the user can choose which sensor charts to show with checkboxes. When they are happy with the sensors selected, the ”Save changes”-button will bring the user back to an updated dashboard, with sensor measurements showing only the sensor charts with the selected sensors.



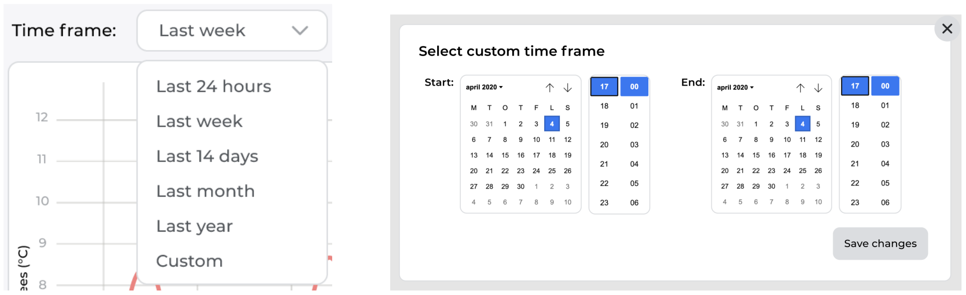
Edit dashboard Select which sensors to show on dashboard

- [✓] Water temperature
- [✓] Air temperature
- [ ] Humidity
- [ ] Radiation
- [ ] Sonar water level, stormwater
- [ ] Precipitation
- [ ] Wind velocity
- [ ] Water level 1, snow melting tank
- [ ] Water level 1 offset
- [✓] Water level 2, waste water
- [ ] Water level 2 offset
- [✓] Water level 3, waste water
- [ ] Water level 3 offset

Figure 14.3: Prototype: Edit Dashboard

14.3.3 Edit Time Frame

When a user wants to change the time frame of the sensor chart, this can be done using the *Edit time frame* feature. Every chart on the dashboard has a time frame picker that controls the start and end date and time for that specific data. This time frame picker has multiple different options for time frames to choose from, such as the last 24 hours, last year, or custom.



Time frame: Last week

(a) Time Frame Picker (b) Custom Time Frame

Figure 14.4: Prototype: Edit Time Frame

14.3.4 Hover Details

When a user is looking at a sensor chart, they might want to see more detailed information about the data points in the chart. If the user chooses to hover over one of the data points, a hover detail will appear. The hover detail includes the date, hour, and measurement value. An example of this can be seen in Figure 14.5.

14.3.5 Create Section

If the user wants to control multiple sensor charts at once with only one time frame picker, the user can click the "Create section" button to create a section. A section includes an elective amount of sensor charts within the same time frame. When the user has selected

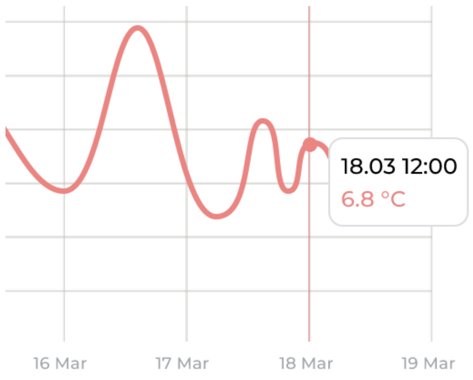


Figure 14.5: Prototype: Hover Details

which sensor data they want in a section, as well as the time frame, they can click the "Add section"-button, and the section will be added to the dashboard.

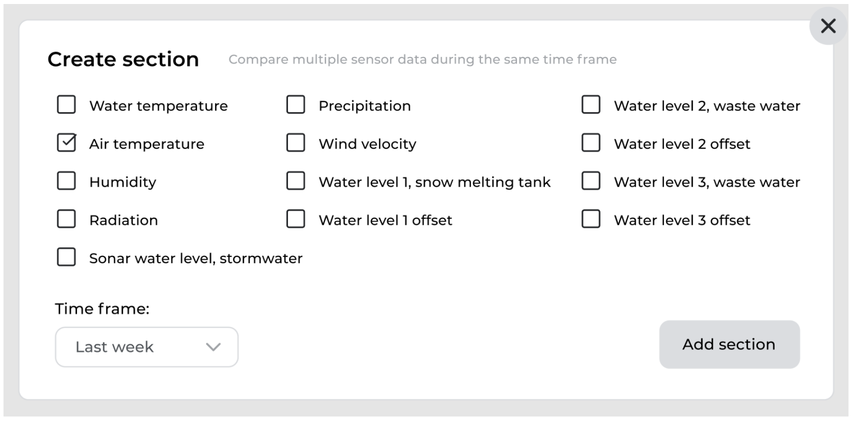


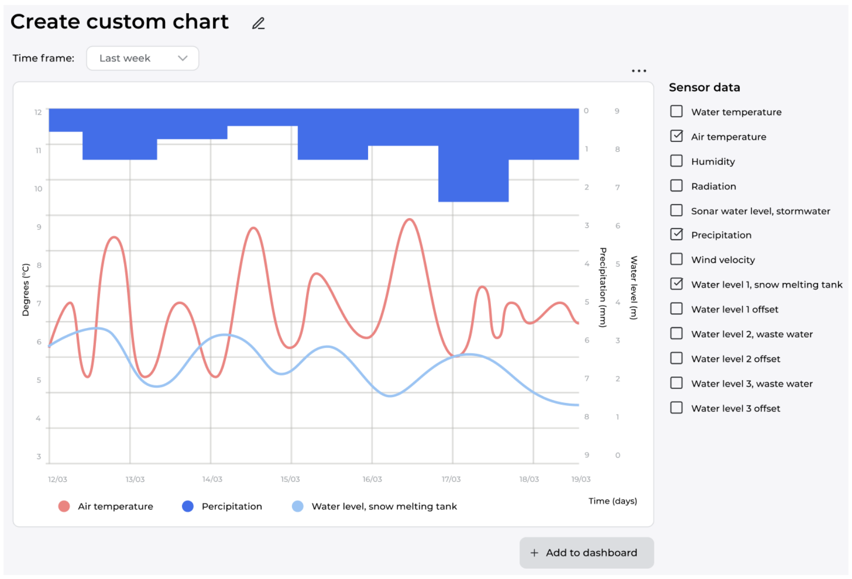
Figure 14.6: Prototype: Create Section

14.3.6 Create Custom Chart

Combining multiple sensor data within one collective chart is made possible through the "Create custom chart"-button from the dashboard. This will lead the user to a separate page where the user can customize a chart. This customization includes choosing which sensor data they want in the chart, choosing a time frame, and giving it a suitable name. After the user is happy with the custom chart, it can be added to the dashboard by clicking the "Add to dashboard-button".

14.3.7 Download as PNG or CSV

Sensor charts, sections, and custom charts all have the option of downloading a PNG image or downloading a CSV file. These options are visible when the user clicks the three dots on top of each chart, which can be seen in Figure 14.8. The PNG image is an image



Time frame: Last week

Sensor data

- ☐ Water temperature
- ☑ Air temperature
- ☐ Humidity
- ☐ Radiation
- ☐ Sonar water level, stormwater
- ☑ Precipitation
- ☐ Wind velocity
- ☑ Water level 1, snow melting tank
- ☐ Water level 1 offset
- ☐ Water level 2, waste water
- ☐ Water level 2 offset
- ☐ Water level 3, waste water
- ☐ Water level 3 offset

Figure 14.7: Prototype: Create Custom Chart

of the chart, which can be useful if the user is writing a paper and needs a visualization of the sensor data. The CSV file includes the raw data measurements that are displayed in the chart. If the user wants to plot the data themselves, this is possible using the CSV file.

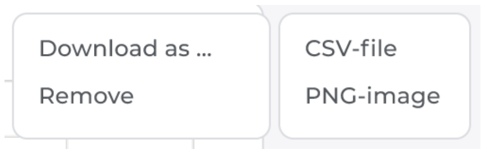


Figure 14.8: Prototype: Download as CSV or PNG

Chapter 15

Implementation

In this chapter, the implementation process will be discussed. The focus is on the method applied as well as its principles. The final solution, which was developed during the implementation, is presented in the following Chapter 16.

15.1 Scrum

The implementation process was influenced by the agile software development method, Scrum [31]. This section will list the most important principles that were adopted.

15.1.1 Product Backlog

A product backlog is an ordered list of improvements that can be done to the product [61]. The initial product backlog was the ordered list of user stories that can be seen in Table 11.4.

15.1.2 Sprints

The implementation was divided into three sprints. Because of the time constraint, each sprint was about one week long. The first sprint was longer than the two following sprints since some extra time was considered necessary to be able to present the first *minimum viable product* (MVP). The sprint cycle for each sprint can be seen in Figure 15.1.

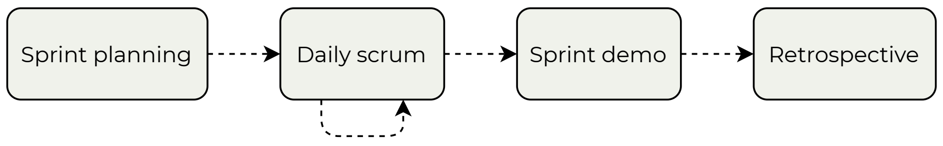
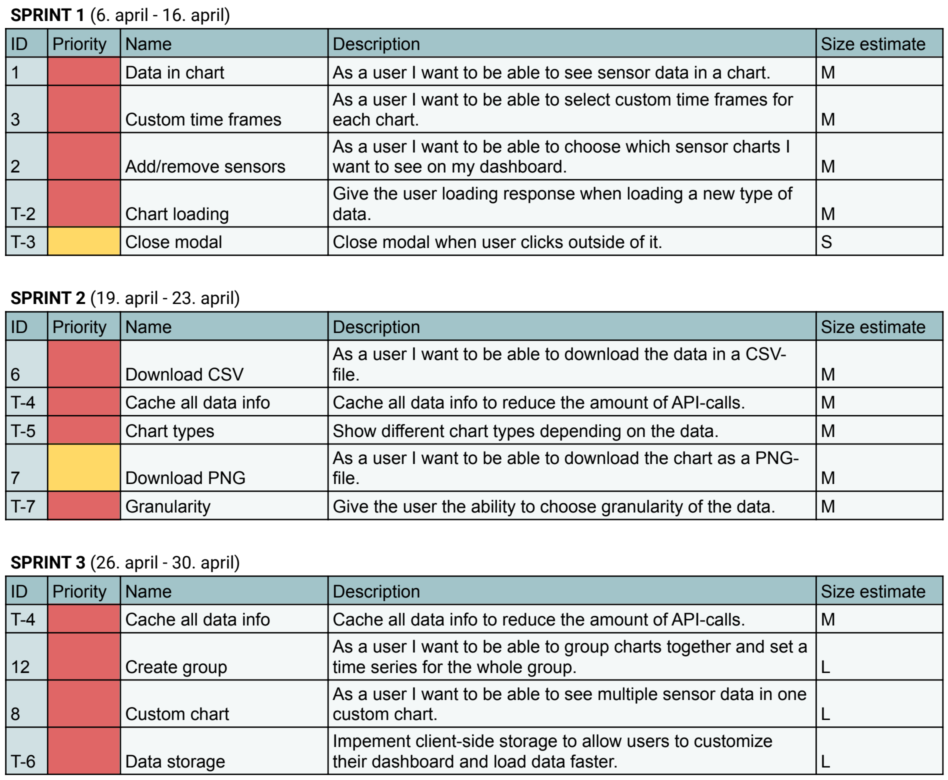


Figure 15.1: Sprint Cycle

Sprint Planning

Before each sprint, a sprint planning meeting was arranged, to select which user stories to include in the current sprint.

The result of the three sprint planning meetings can be seen in Figure 15.2. User stories with IDs that start with *T-* are technical stories that were created as the implementation process went on. Their purpose was to deal with issues that should be solved, but that were not a part of any particular user story.



SPRINT 1 (6. april - 16. april)

ID	Priority	Name	Description	Size estimate
1	🟥	Data in chart	As a user I want to be able to see sensor data in a chart.	M
3	🟥	Custom time frames	As a user I want to be able to select custom time frames for each chart.	M
2	🟥	Add/remove sensors	As a user I want to be able to choose which sensor charts I want to see on my dashboard.	M
T-2	🟥	Chart loading	Give the user loading response when loading a new type of data.	M
T-3	🟨	Close modal	Close modal when user clicks outside of it.	S

SPRINT 2 (19. april - 23. april)

ID	Priority	Name	Description	Size estimate
6	🟥	Download CSV	As a user I want to be able to download the data in a CSV-file.	M
T-4	🟥	Cache all data info	Cache all data info to reduce the amount of API-calls.	M
T-5	🟥	Chart types	Show different chart types depending on the data.	M
7	🟨	Download PNG	As a user I want to be able to download the chart as a PNG-file.	M
T-7	🟥	Granularity	Give the user the ability to choose granularity of the data.	M

SPRINT 3 (26. april - 30. april)

ID	Priority	Name	Description	Size estimate
T-4	🟥	Cache all data info	Cache all data info to reduce the amount of API-calls.	M
12	🟥	Create group	As a user I want to be able to group charts together and set a time series for the whole group.	L
8	🟥	Custom chart	As a user I want to be able to see multiple sensor data in one custom chart.	L
T-6	🟥	Data storage	Impement client-side storage to allow users to customize their dashboard and load data faster.	L

Figure 15.2: Result of Three Sprint Planning Meetings

Daily Scrum

To keep us continuously up-to-date on the implementation status, the daily scrum was adopted as a principle. Every day during the implementation phase, 15 minutes were set aside to discuss what had been done the day before, and the plans for the current day. The daily scrum was also an opportunity to bring up any potential issues that needed solving.

Sprint Demo

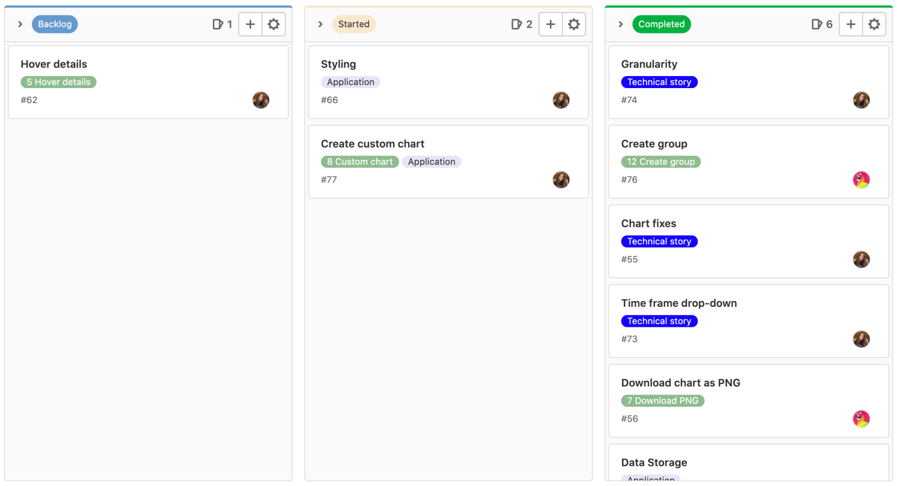
Each sprint ended with a sprint demo in front of the stakeholder and the supervisor. The demos were a good opportunity to get feedback from the stakeholder.

Sprint Retrospective

Sprint retrospectives exist to increase quality and effectiveness within the team as the sprints go on [31]. It is usually held at the end of the sprint to discuss how it went. The important elements to discuss are what went well during the sprint, what could have been better, and how similar problems could be solved in the future.

15.1.3 Scrum Board

Boards on GitLab were used for organizing the backlog issues and keeping track of the progress during the sprint. Each user story was either given its own issue or broken down into multiple smaller issues, depending on its size. The technical stories and issues regarding design refinements were also placed on the boards. An excerpt from the board is shown in Figure 15.3.



Backlog 1

Hover details
5 Hover details
#62

Started 2

Styling
Application
#66

Create custom chart
8 Custom chart Application
#77

Completed 6

Granularity
Technical story
#74

Create group
12 Create group
#76

Chart fixes
Technical story
#55

Time frame drop-down
Technical story
#73

Download chart as PNG
7 Download PNG
#56

Data Storage

Figure 15.3: Scrum Board on GitLab

15.1.4 Pair Programming

Pair programming involves working together as a pair during programming [31]. As a practice, it can both improve code quality and knowledge within the team. In this project, pair programming was especially used if one of us was stuck on a task and needed help to solve a problem.

15.1.5 Code Review

Code review was used in combination with pair programming to improve the code and to ensure that new code would not break or override important code. The code reviews were done directly in GitLab, as it is an integrated feature.

Chapter 16

Final Concept

This chapter presents the final concept, which is the web application Nykken. Firstly, it will present the changes that were made after the prototype testing. Afterward, the application's main features and core purpose will be described.

16.1 Changes from the Prototype

In this section, the changes made from the prototype to the developed solution, Nykken, will be presented.

16.1.1 New Name and Logo: Nykken

Initially, the webpage was referred to as *Risvollan wastewater station*. Many of the users commented on this, saying that it was a hydrological station, measuring more than just wastewater. It was therefore decided that the application needed its own name. After a brainstorming session, the name *Nykken* was chosen as the application name. Nykken is Nynorsk for "Nøkken", which is a mythological water creature from Norwegian mythology.

Since the application had gotten a new name, it was also designed a logo. The logo can be seen in Figur 16.1.

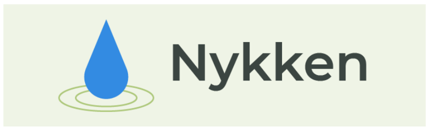


Figure 16.1: Nykken: Logo

16.1.2 New Color Scheme

When designing the Figma prototype, no cohesive design with a font and color palette existed. Before creating the final solution, some key design changes were made. The

Figma prototype had a completely grey color scheme, which put the focus mainly on the charts but also gave it a sterile feel.

A new color scheme was created to be pleasant to look at, with colors that could harmonize, but not take focus away from the colored charts. This color scheme can be seen in Figure 16.2.

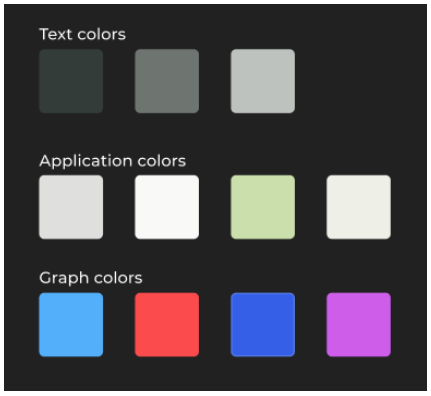


Figure 16.2: Nykken: Color Scheme

16.1.3 Renaming Sections

When user testing the prototype, there was a lot of misunderstanding regarding the concept of sections. As a result of this, it was decided to rename *sections* to *groups*. A more in-depth explanation of how the users found this confusing can be found in Section 18.3.1.

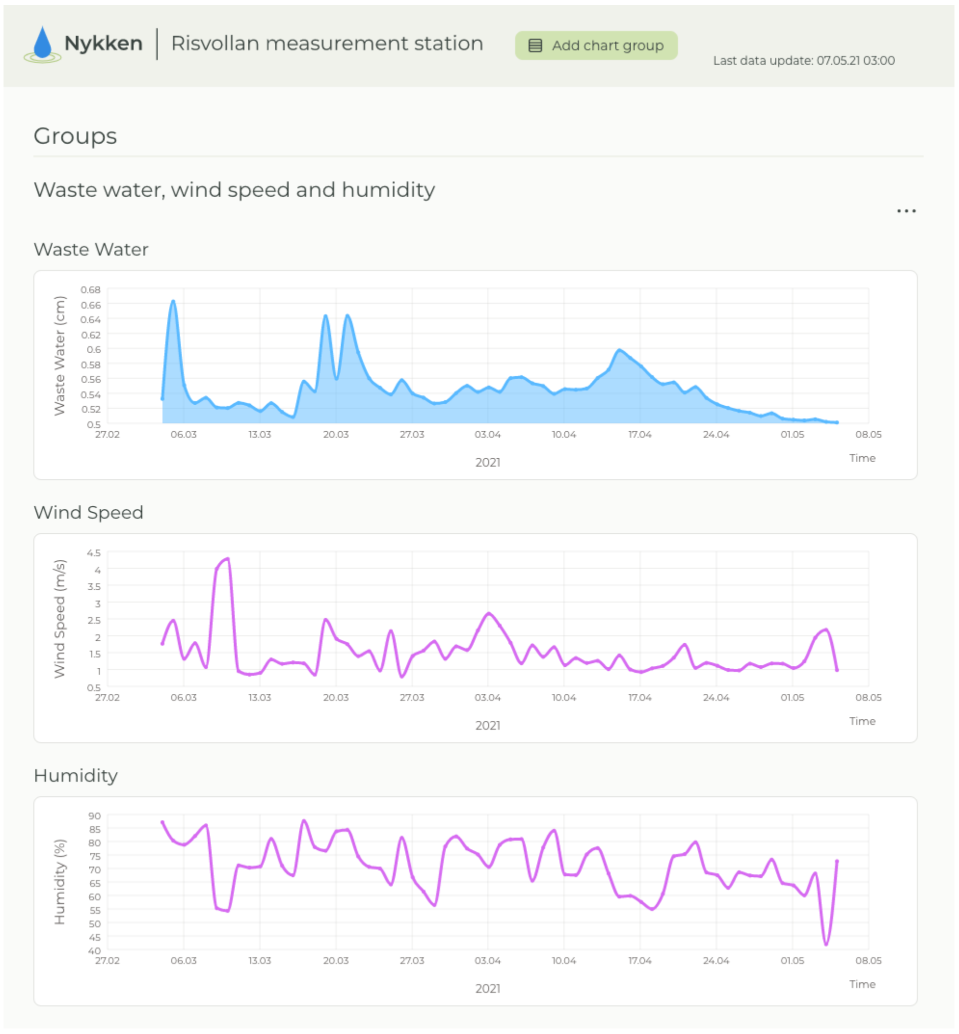


Figure 16.3: Nykken: Groups

16.1.4 Dashboard Changes

There were some changes made to the application dashboard. The main changes included:

1. Placing the "Add chart group"-button in the header for easy access to the functionality
2. Adding latest data update to the header to inform users of when the data was last updated
3. Changing "Edit dashboard"-button to "Add/remove sensors" to make it more clear what the functionality would involve
4. Placing the "Add/remove sensors"-button in the Sensor measurements part of the application

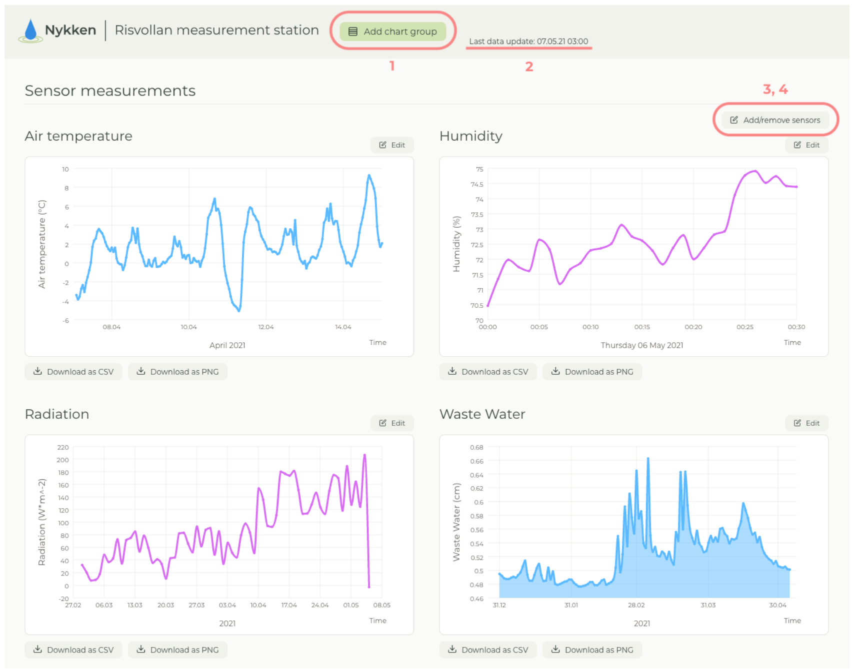


Figure 16.4: Nykken: Dashboard Changes

16.1.5 Edit Time Frame

Custom Time Frame

During the user testing of the prototype, the users had tasks that involved changing the time frames of the sensor measurement charts. All of the users chose to use the ”Custom” option in the time frame picker, instead of using the ”Last X hours/days/months/year” options. This was the case, even when the tasks specifically involved a time frame that would fit into one of these other options. Since this feature did not stand out as interesting or helpful for the users, it was decided that during the development of the final concept, we would focus on the ”Custom” time frame picker.

Granularity

When a user wanted to see sensor measurements from a longer time period in a chart, the amount of data retrieved from the API would sometimes be so large that it would cause the page to crash. During the development process, it was therefore decided to implement a ”Granularity”-select, where the user would have some options to choose between. Depending on the time period, these granularity options could include ”Hourly”, ”Daily”, ”Weekly”, ”Monthly”, ”Yearly” or ”As measured”. The option ”As measured” would be the only option that showed all of the sensor data, which in most cases would be a measurement every minute.

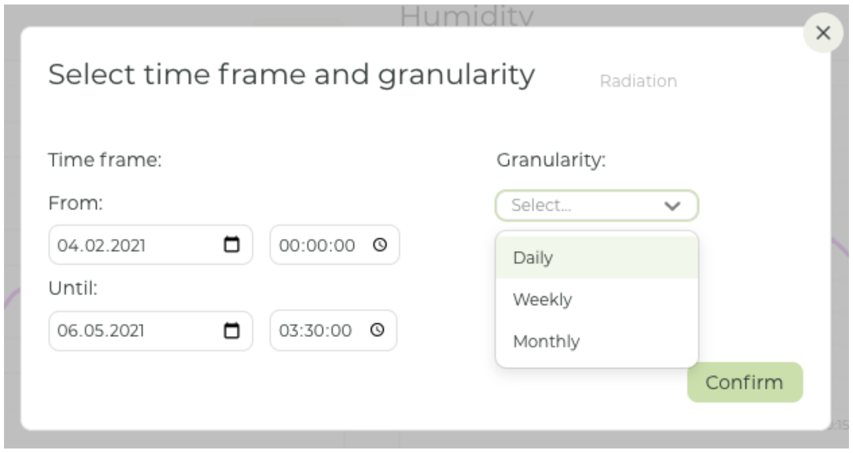


Figure 16.5: Nykken: Select Time Frame and Granularity

Download Buttons

In the Figma prototype, the options of downloading PNG images and CSV files of the sensor measurement charts were placed in a drop-down menu at the top right corner of the charts. However, in the final concept, it was decided that these two download options should rather be two buttons at the bottom of each chart. The main reason for this was that downloading raw CSV-data was one of the main features that the users wanted, and by having buttons instead of drop-down options, the users were one click less away from this feature, and by making it more visible, it would also be more accessible to the users.

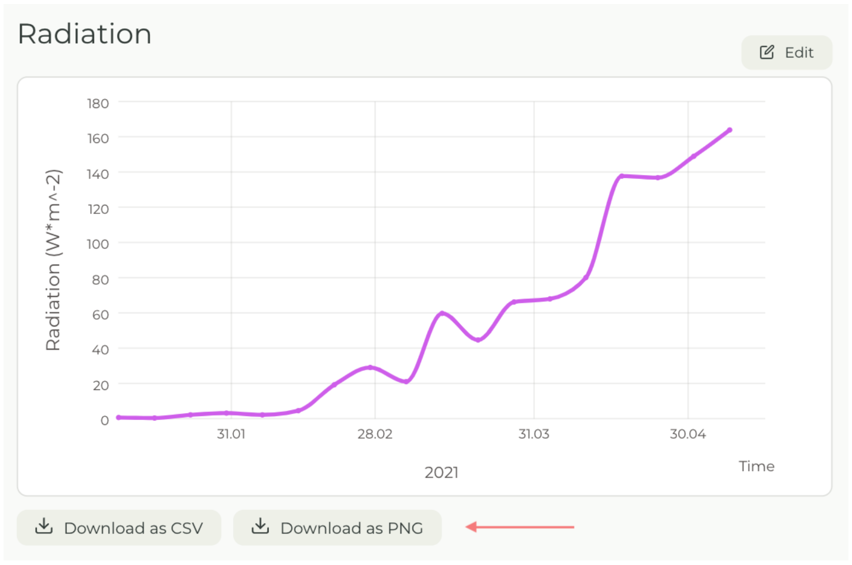


Figure 16.6: Nykken: Download Buttons

16.1.6 Not Including Custom Charts

During the prototype testing, most users found the custom charts useful. Despite this, due to time restrictions, the custom charts were not included in the final concept. A full overview of which user stories were implemented can be found in Appendix E.

Chapter 17

Validation of Requirements

In this chapter, the validation of the requirements from Chapter 11 will be discussed.

17.1 Definitions of Completion

When deciding if a requirement was validated or not, the definitions of completion had to be defined. These were divided into three possible options: Complete, Partially Complete, and Not Complete. This information can be found in Table 17.1.

Completion	**Elaboration**	**Definition**
C	Complete	Requirement fulfilled
PC	Partially Complete	Requirement mostly fulfilled, but with some restrictions
NC	Not Complete	Requirement not fulfilled

Table 17.1: Definitions of Completion

17.2 Functional Requirements

In Table 17.2, each functional requirement was validated as one of the three definitions of completion, with additional comments, to discuss why the requirement was or was not fulfilled.

ID	Description	Completion	Comments
F1	The application shall have a dashboard	C	A dashboard is present in Nykken, presenting most of the functionality.
F2	The application shall visualize sensor data	C	Nykken visualizes all sensor data in charts and sensor groups.
F3	The application shall provide data download	PC	Nykken provides data download, but it is limited based on the granularity options for the selected time stamps.
F4	The application shall allow users to combine multiple sensor data in a visualization	PC	It is possible to combine sensor data in sensor groups, but it is not possible to combine multiple sensor data in one chart,
F5	The application shall allow users to choose which sensor data they want to see visualized	C	The user decides which sensor charts should be visible on the dashboard.
F6	The application shall allow users to select which time frame they want in the visualizations	C	This is made possible through the *Edit dashboard* functionality.
F7	The application shall allow users to download image versions of sensor data visualizations	PC	Sensor charts can be downloaded as PNG images, but sensor groups do not have this functionality.
F8	The application shall give users information about existing sensors	C	The user gets an overview of all available sensors when adding/ removing sensors from the dashboard. When the user selects a sensor to visualize, information about the unit visible in the chart.
F9	The application shall allow users to remove added visualizations from the dashboard	C	It is easy for the user to add/remove whichever sensors they want to and from the dashboard.
F10	The application shall provide the latest data available	PC	Currently, this latest date for retrieving data has to be hardcoded by a developer.
F11	The application shall provide the date for when the data was last updated	PC	Of similar reasons to F10, this has to be done by a developer, but it is possible through minor changes in the code.

Table 17.2: Validation of Functional Requirements

Since the non-functional requirements are validated through user testing, they will be discussed in Section 20.2, after the results have been presented.

Part V

Results

This part will present a detailed description of the results gathered from the different data generation methods used during this project. This includes a description of the participants and how they participated throughout the project, as well as how they responded to the prototypes created in this project.

Chapter 18

Results

Firstly, we will describe the different participants in each stage of the process. We will then give a detailed description of the results gathered in the same stages.

18.1 Participants

In total, nine people participated in our project. Out of these, three of them contributed in all five stages. All of these were from the WWSE-group. We also contacted bachelor's students studying hydrology to participate in the user tests, but all but one declined due to final exams and project work deadlines. We did not invite the experts from NVE to participate in the user test, as we chose to only test on potential users. A complete overview of the participants and their contributions can be found in Table 18.1. Because of the limited amount of available participants, we chose to invite our stakeholder to participate in both observations and questionnaires, even though he was involved in the project through meeting attendance and regular contact. The participants will be further discussed in Section 19.1.

Code	**Interview**	**O1**	**Q1**	**O2**	**Q2**
Prof1	X	X	X	X	X
Prof2	X	X		X	X
PhD1	X	X	X	X	X
PhD2	X	X	X	X	X
PhD3		X	X	X	X
Tech	X				
NVE1	X				
NVE2	X				
BS				X	X

Table 18.1: Participation Overview

18.2 Interview

Seven people were interviewed as a part of the first stage of the design and development process. The PhD Candidates and professors from the WWSE-group were considered our main user group. Another potential user was the technician, also from the WWSE-group. The experts from NVE mainly talked about their experience with data visualization and shared useful insight into how NVE visualized their data. The interview results will therefore be divided into two categories; the user group and NVE.

18.2.1 WWSE-Group

The results from the interviews with our potential user groups shared many similarities, with only small variations. We will now present an overview of the answers from these interviews, mentioning both similarities and outliers from the answers collected.

Current Situation

For the participants using the data for research, their main task when using the sensor data is *machine learning* (ML) modeling. The data from Risvollan are used to train their machines to predict the runoff in the sewage and drainage pipes. One of the professors also uses the data when lecturing. The technician's main task is to make sure that the sensors are working correctly and that the data is registered as it should.

When describing their current situation and daily workflow, most participants answered that they first had to get access to the data. For some, that simply meant downloading it from a database. However, for others, that meant finding someone that had access to the data and ask them to download and forward it. This resulted in an inconsistency in file format and file type. Most of the participants therefore spent time cleaning and trimming the data, and stitching files together to make it usable for their tasks. The technician had easier access to the data, but he also struggled with large data files that needed clean-ups and trimming before use.

In general, weather and runoff data were listed as the sensor data they mostly interacted with, when doing research. This is the data that is required in the ML models they are developing. One professor stated that they looked at events, with big rainfalls being specifically mentioned. Since the technician's main task is to monitor the sensors, all sensors are equally relevant to them.

According to all participants, they used different time frames and granularities, depending on the data they were using and the work they were doing. One of the technician's tasks was to change faulty sensors, so they usually checked the incoming sensor data from recently replaced sensors to make sure it was working properly and that the sensor data was registered in the right way.

Existing Solutions

In this interview section, all of the participants focused on the solutions they were using when working with the data from Risvollan. Once they had gotten access to the sensor data, all participants said that they did some form of control to see if it was good to use. This included importing the sensor data into a suitable application and cleaning it, either manually or by using scripts, before most of them stated that they plotted it into a chart to get a quick overview of it.

The most common application used to handle the sensor data was R. Excel, Python, and MatLab were also mentioned as suitable options. The participants said that they mostly used the application they had already used to clean and trim the data, to visualize it. Both R and Excel offer some customization options, such as changing the color legends and renaming the axes, but since most of the participants mainly used the visualizations to make sure that the data was correct, they did not need many customization options.

Features and Functionality

The main focus of this interview section was which data the users wanted to see, and how they wanted to see it. Here, some participants mentioned that they sometimes liked to combine different sensor data together in one chart, such as temperature, flow data, and rainfall, with rainfall as a bar chart upside down on top, to look for correlations. Other interesting comparisons mentioned, were flow in similar time periods, and one also mentioned that they would like to be able to upload their own data to compare their predictions with the actual data.

Motivation and Expectations

In the last section, we discussed the users' expectations for a new system and their motivations for potentially using it. The most wanted feature and area of improvement mentioned here was accessibility. As discussed, the sensor data was not easily downloadable, and it required multiple steps to get it in a usable format. Another important feature mentioned, was the ability to only download the sensor data required for a specific task. This included both selecting just one sensor to do download data from, as well as only downloading the sensor data in a given time frame.

Many also mentioned that it would be nice to get to see the sensor data visualized before they downloaded it because that could help indicate if the sensor data was faulty or not. There were, however, disagreements about what was needed to detect faulty data. For some, just looking at the charts would be sufficient, but others preferred it if some sort of notion were given, either using color coding or notifications.

When asked what they considered the most important aspect in a data visualization system, all but one answered that it should be easy and comfortable to use. Some further elaborated and said that it should not include too much information and that the charts should be easily understood by first and second-year students.

18.2.2 NVE

We interviewed two experts from NVE, one being a hydrologist and one being a science engineer, turned system developer at NVE. The hydrologist let us know that they had not worked with any data from RMS and that they did not have much insight into data visualization other than the occasional usage of the systems developed at NVE. They therefore suggested that we talked to someone with a technical background and then provided us with the system developer's contact information.

The system developer had worked with visualizing hydrological data since the ’80s and had developed several visualization systems during his time at NVE. The last one launched being Sildre, the application mentioned in Section 6.1. We discussed topics related to both development and user satisfaction, with the most important ones being described below.

Standards

A central part of our discussion was how different data should be visualized and if any standards exist. According to them, there are no official standards, but there are certain obvious rules that are “self-explanatory”. Examples of this are that sensor data that is accumulated in a given time frame, such as rainfall and melting water, should be visualized as bar charts, and instantaneous measures, such as temperatures, should be line charts. Another standard practice commonly used is to use red and blue to indicate if the temperatures were above or below zero degrees Celsius. They also mentioned that users often appreciate seeing a representation of maximum and minimum standard measurements, to help the users relate the incoming data to what is normal.

Features and Functionality

A desired feature by their users was seeing data from multiple sensors or multiple time series from the same sensor at once. The expert suggested giving the users both the possibility to have the data in the same chart, but also grouping charts together and showing them in the same time frame. Other commonly desired features mentioned, were the option to change the names of the axes, renaming the charts, changing color legends and thickness of the lines, and adding and removing the normality probability band from the graph.

Something they also mentioned was that interactivity, and especially the ability to execute most actions using only the mouse pointer, was a great functionality to offer. This means giving the users the possibility to zoom in by highlighting the area they wanted to focus on using their mouse pointer, instead of having them input it into the time frame picker. This would allow for efficient interactions that quickly give the users an overview of the data they are looking at.

18.3 Observation

Five people participated in the first user test, and six people participated in the second user test. The results from each user test will be presented in the following sections.

18.3.1 1st User Test

As mentioned in Subsection 4.3.2, the participants were given scenarios with tasks to complete during the observation. Because each scenario was created to highlight a feature in the prototype, the following sections will describe how the participants responded to each feature presented, as well as general observations made by us, and comments and feedback from the participants.

General

One thing we noticed, was that the participants responded differently to the limitations in the prototype. While some chose to try and click everything and follow up with explanations of what they would have done had all the available buttons worked properly, others chose to look for other options to solve their tasks. This gave us valuable insight into how the users perceived the different tasks and features presented to them.

Something else we noticed, was that the users forgot which day they were told it was. Due to the prototype's limitations, we told the users to act as though it was the 19th of March 2021 at 11:00, to match the dates and times in the prototype. This caused some confusion when the users were asked to look at the last dates or hours, as the prototype only showed the sensor data in a pre-defined time frame.

In general, the participants seemed to like the prototype. One commented that they “quite liked the workflow” and that it was “quite good”. The participants also seemed content with the features presented in the prototype. Even though many participants had stated in the initial user interviews that they were mostly interested in the raw data for their analysis tasks, they seemed to enjoy seeing the data in a chart before they downloaded it. It seemed to be a consensus that even though they had to spend some time exploring the different features, they still felt it was intuitive and easy to use, with a clean design.

Looking for Values

When asked to find the air temperature on the given dates, the participants quickly identified the correct chart and started interacting with it. Most users discovered the hovering effect through this interaction and understood that they could find the desired values this way. However, when asked to look for other values, all users opted for the ”custom” time frame picker instead of the ”latest ...” options available. While some users considered opting for the predefined options, others went straight to ”custom” without hesitation. One user did not seem satisfied with reading the values off of the chart and wanted to download the raw data to find the values there instead. Overall, the hover effect did not seem to be the first solution that came to mind when asked to look for certain values, but once it was discovered, most users decided to read the values from hovering over the specific dates and times. All participants preferred customizing the time frames themselves rather than choosing the predefined options.

Chart Sections

The biggest confusion connected to the chart sections was the name of the button. The participants did not understand that a chart section could be used to group charts together to control them simultaneously. Due to this confusion, most participants either chose to select the charts they wanted to see and add them to the dashboard and change the time frames individually for each chart. Others chose to click the ”Create custom chart”-button and add the desired sensors into the same chart. Only one participant actually clicked the ”Create Section”-button to see what happened. When asked why they chose not to click the ”Create Section” button, the participants answered that it did not seem intuitive and that it confused them, since they did not understand what a section was. When the participants were shown the chart section after the test, they found the feature to be interesting, with one stating it was “really cool”. They all suggested calling it *group* instead of *section.*

Custom Chart

The custom chart was something that several participants mentioned in the user interview that they had interacted with and looked at before. Many commented that they really liked the rainfall as an upside-down bar chart in the prototype. One user exclaimed “That’s how you do it!” when they saw the newly created custom chart. Another participant asked which program we had used to generate the chart, and was saddened to learn that it was a drawing created by Sunniva. Some users commented that it was hard to see which y-axis belonged to which graph, and suggested using color-coding to make it clearer. Another user also preferred the color legends to not only match the color of the graph, but also the shape. This means that bar charts would have a box-shaped color legend and that lines would have a line with dots at each end as color legends.

The option to either group the charts together to see them separately or to plot multiple sensor data in the same chart, seemed to please the participants.

Download Options

The most stated pain point we uncovered during the interviews, was the accessibility of the sensor data, or rather, lack thereof. This was further confirmed after the first observation. One participant stated about our application that “The most important thing to remember is that it should be a tool to get data, not plotting data in an unlimited amount of ways”. They added that as long the data is available and downloadable, they can choose to either use our pre-made charts, or plot the data into another visualization program and customize it exactly how they want it. Any way, this new and simple access to the data would help them in their everyday work. Another user said that they were really looking forward to testing the application once it had real data, indicating that they wanted to start downloading CSV files right away.

18.3.2 2nd User Test

As in the first user test, the participants were given five scenarios with tasks to complete during the observation. We will now present the results from the 2nd user test.

General

One thing we quickly noticed, was that the page slowed down considerably if the users requested a large amount of data. This led to some loading time if the users chose a bigger time frame than the tasks asked for, and one user also had to restart their browser because it crashed. To avoid this in the following user tests, we decided to limit the amount of data a user could request, by removing the option to get the data *”As measured”* if the time frame was too big. Once this option was removed, we did not have the same problems anymore.

Due to the confusion about what time it was in the first user test, we started our second user test by asking the participants to find out when Nykken was last updated. This way, we could both observe how they liked this new addition to the web page, but also give them notice that this was the latest data they could request.

There were also some design issues that we were aware of in the first user tests, that we did not have time to fix because of some unforeseen technical issues that occurred in the last couple of days of development. This included some inconsistencies in the button placements, as well as some drop-down menus with only one available option. This also led to some confusion in the first three user tests, but they were fixed before the last three.

However, the overall impression was that the participants were happy with Nykken. One participant stated that Nykken ".. is extremely cool and I cannot wait to start using it", and later told us "You can deliver your thesis knowing that you have made a contribution that we can use both in research and in education". The other participants also commented that they liked the clean design without too much clutter and that they liked what they saw.

Edit Dashboard

When asked to add sensors to the dashboard, most users found the "Add/remove senors" button and added the sensors to the dashboard. Some users, however, either also created a chart group and added it to the dashboard or went straight for the chart group option. When asked why they chose to add a chart group instead of adding the sensor separately, most users stated that it was the biggest and most visible button on the web page and that it seemed like a good place to start.

Edit Time Frame

As previously mentioned, there was some confusion regarding the dropdown menus with just one option. This was very apparent in the time frame picker. When the users opened the time frame picker to select a new date, they saw that "custom" was selected and wanted to change it. But when they clicked it, they did not get any new options and did not realize they had to click the "custom" option for the time frame picker to appear. However, all users eventually tried clicking it and managed to edit the time frame. In the last three user tests, none of the users had any trouble changing the time frame.

There were, however, some issues connected to the design of the time frame picker. Nykken uses the time frame picker generated by the browser, so the design varied greatly between the users. One participant was using a friend's computer that had an American time frame picker, and they therefore struggled to choose the correct times because of the AM and PM format. Another participant had to manually type in all the values, which was not as fast a using just the mouse pointer.

The participants seemed to like the new feature that allowed them to change the granularity. Most of the users chose a granularity without it being a part of the tasks when choosing a time frame. When choosing the granularity, most of the participants seemed to consider which granularity would be the most logical, for instance, choosing daily average for time frames that spanned multiple days and hourly when choosing one day.

Download CSV

The participants seemed excited to learn that they could actually download real measurements from Risvollan. When opening the CSV file, most participants commented that they liked how clean it looked. One participant stated “that is quite rare”. Something that was noted was that the measurements came before the time stamp and that the header for the measurements did not state which unit they were in.

Download PNG

The option to download the charts as PNG did not seem that interesting to the participants. Many stated that as long as they had the raw data, they had no trouble plotting it into a chart in R or Python. Something we also learned doing the user testing was that there are size requirements for figures included in scientific papers. As of now, the PNGs are the chart component exported as an image, meaning that it does not uphold to those requirements. One participant also commented that they would have preferred to download it as PDF instead, as they usually used PDFs in their papers.

In general, the participants seemed to think that the charts shown on Nykken were sufficient to get an overview of the data they wanted to download, but would like more customization options if they needed a chart for a paper.

Add Chart Group

Since all but one participant had tested the prototype, most participants seemed familiar with how the grouping worked. However, the concept of grouping charts together did not seem like something they had done before this project. The bachelor’s student commented that they thought the charts would have been layered to create a chart with multiple charts in it, similar to the custom chart feature in the prototype when creating a group. One participant stated that it was “actually a cool feature” and that it could sometimes be more useful than a custom chart, because it was “not as clumped”. Some participants also commented that the wide style of the group charts was nice, as they filled more of the screen than the single charts.

18.4 Questionnaire

All the participants from the observations were asked to fill out a short Google survey after the observation was complete. The results from these questionnaires will be presented in the following sections. The SUS scores from both questionnaires are presented in Table 18.2.

The SUS score is calculated by subtracting 1 from each odd statement and by subtracting the users’ responses from 5 for each even statement. This means that for each statement, the top score is 4. In Table 18.2, the scores from each participants’ responses have been added together, meaning that the top score for each statement is 16 for the first questionnaire, and 24 for the second. The total sum is all the statements’ scores summed up. The final SUS score is found by dividing the sum by the number of participants and multiplying it by 2.5, to get a score from 1 to 100.

ID	Statement	Score 1	Score 2
1	I think that I would like to use this system frequently.	13	18
2	I found the system unnecessarily complex.	15	23
3	I thought the system was easy to use.	15	22
4	I think that I would need the support of a technical person to be able to use this system.	16	19
5	I found the various functions in this system were well integrated.	13	16 (one reply missing)
6	I thought there was too much inconsistency in this system.	13	21
7	I would imagine that most people would learn to use this system very quickly.	11 (one reply missing)	23
8	I found the system very cumbersome to use.	16	24
9	I felt very confident using the system.	15	20
10	I needed to learn a lot of things before I could get going with this system.	16	24
Total Sum		143	210
SUS-score ((Sum/Number of participants) * 2.5)		89.375	87.5

Table 18.2: Results from the System Usability Scale

18.4.1 1st Questionnaire

Three PhD Candidates and one professor answered the questionnaire following the first user test. One participant from the user test did not respond. The main goal of the questionnaire was to examine if our ideas and features matched the users' expectations.

To help us prioritize which features to focus on when developing, we asked the participants to rate the features presented in the prototype on a scale from 1 to 5, with 1 being *"Not relevant"* and 5 being *"Highly relevant"*. The distribution of the ratings can be found in Table 18.3. From this table, we can see that all participants rated downloading the raw data as highly relevant. Changing the time frames and creating sections of charts to control simultaneously were also rated relevant. The option to create and download a new chart as PNG differed in rating. This can be because two different features were presented as one in the questionnaire, which did not allow for the users to rate them separately. During the user tests, most participants seemed to appreciate the opportunity to create a new custom chart, but the interest in downloading it as PNGs did not seem equally high.

Feature	1	2	3	4	5
Hovering over the graph for more information	0	1	1	2	0
Changing the time frames	0	0	0	1	3
Creating a section of charts	0	0	0	1	3
Creating and downloading a new chart as PNG	0	1	1	1	1
Downloading the raw data	0	0	0	0	4

Table 18.3: Rating of Features According to Relevancy

We also wanted to see if our workflow and design were usable. As shown in Table 18.2, the users seemed to think that our prototype was above average in usability. Even with one score missing for one of the statements, our prototype almost got a score of 90 out of 100, leaving it in the top 10% of SUS scores. This was a good indication that our prototype was easy to use and understandable.

One participant also left a comment in the optional text box stating: “I think the prototype is very intuitive to use, it gives basic functionalities. I think I just miss a small mention that gives time and hour of the last successful update of the data or something like ”last update: 19/03/20 14:00””.

18.4.2 2nd Questionnaire

The second questionnaire had six respondents, three PhD Candidates, two professors, and one bachelor’s student. The main goal of the second questionnaire was to measure the usability of the final solution. By using the SUS again, we could compare the score from the prototype with the score from the final concept. As shown in Table 18.2, there was a small decrease in the SUS-score from the prototype and final solution, with the new score being 87.5, down from 89.375. However, the score is still in the top 10%, so we can assume that the participants found it easy to use.

Due to few respondents, it was not possible to analyze differences in SUS scores between gender, age, or occupation.

Part VI

Discussion

This part will present a discussion of the research methods used in this project. An assessment of the application's visualization design is also included. Lastly, a more detailed evaluation of the entire project will be given, including answers to the research questions and project goal.

Chapter 19

Research Methodology Evaluation

This chapter will present a discussion and evaluation of the research methodology used in this project.

19.1 Participants

There were nine participants in this project. Three of these participated in all five stages, whereas two participated in four stages each. In an article from 2018, Mothershaw states that you should consider reusing your research participants when “your research is targeted towards a restricted target audience”, which was the case in this project. This project was also limited in terms of time and resources. The decision to reuse the participants was therefore made, because we wanted to make an application for the WWSE-group studying the hydrological data from the RMS, not for a general audience. As stated in Section 18.1, we asked bachelor’s students to participate to expand our test population, but only one accepted the invitation. The reusing of participants may have affected the outcome of the second user test, as most participants already were familiar with the concepts in the application.

The decision to include the main stakeholder in the user tests was done partly because he also was a part of our user group and partly because we needed a bigger test population. Several studies have shown that most issues are uncovered with only five testers, and with the stakeholder participating in the first user test, we had five participants [39, 23, 24]. His participation may have affected the results, both positively and negatively. While he might have had some additional familiarity with the project compared to the other participants, he also knew us better, and therefore might have been more honest and direct in his feedback.

19.2 Methods

Several methods were used during this project. We will now review the performance of each of these methods in the following sections.

19.2.1 Semi-Structured Interview

As mentioned in Subsection 4.3.1, interviews are suitable for generating detailed information. It is also a vital part of user-centric design. The interviews gave us a great amount of information about the user groups, their everyday work lives, and their expectations about our project. The semi-structured interviews let the participants speak freely and the discussion moved naturally from topic to topic. While this resulted in us uncovering some unexpected pain points, it also steered the discussions away from the topic of visualization. This might have affected the outcome of this project, as we chose to focus more on giving the user groups access to the data, rather than the visualization of it.

With the people from NVE, we did not know what to expect, so the decision to make it a semi-structured interview seemed appropriate. This resulted in the interview being a combination of an interview and a presentation of different visualization tools used in NVE. In this case, it ended up giving us both information and inspiration.

The interviews were the most time-consuming in regards to planning, conducting, transcribing, and analyzing, but the results from the interviews served as the foundation for the rest of the process and helped us uncover our users' needs.

19.2.2 Systematic Observation

As with interviews, systemic observations are a great way to collect qualitative data. By systematically observing the participants during the user tests, we could discover how the participants interacted with our prototype and final concept. There are, however, downsides to using systematic observations. As mentioned in Section 4.5, the Hawthorn Effect can affect the outcome of the behavior of the participants. Also, observations may not provide or explain intentions, meanings and reasons [43]. However, the participants were good at describing what they saw and their intentions and thought processes behind their decisions when trying to complete their tasks.

Since each user test only lasted around 15 minutes, it does not correctly illustrate how the participants would use the application in real life but rather demonstrates the acceptance of the functionally and concept of the application. To get a more correct impression of the usage of the application in more realistic scenarios, a longer testing period should be conducted.

19.2.3 Questionnaire

A questionnaire is a quick and efficient way to gather quantitative data. With only four and six respondents in the two questionnaires, the results can not be generalized to a larger group, such as other researchers working with similar data. However, since we were working with a specific user group and there was little variation between the responses, the questionnaires still provided value in the project.

Chapter 20

Application Evaluation

This chapter will evaluate the application design and the validity of the non-functional requirements of the application and final solution, Nykken.

20.1 Design Evaluation

In this section, the design of Nykken will be evaluated based on the theory from Chapter 5. It will more specifically discuss improvements that could be implemented in Nykken.

20.1.1 Chart Types

The decisions on chart types for each of the sensor data were based on standards that were observed in existing solutions, especially Yr and Sildre, which are written about in Chapter 6. The decisions were also based on Hardin, Hom, Perez, and Williams' recommendations for when to use certain types of charts [21].

Line Charts

The majority of the sensor data were visualized using line charts. This type of visualization covers all sensors besides the rainfall data. Line charts are used to show trends in data over time [21]. They are also used as a simple way to connect data points and visualize them as a sequence of values. Data measurements like temperature, humidity, radiation, and wind speed are all values that continuously change over time, and a line chart is therefore a good way to visualize these types of measurements.

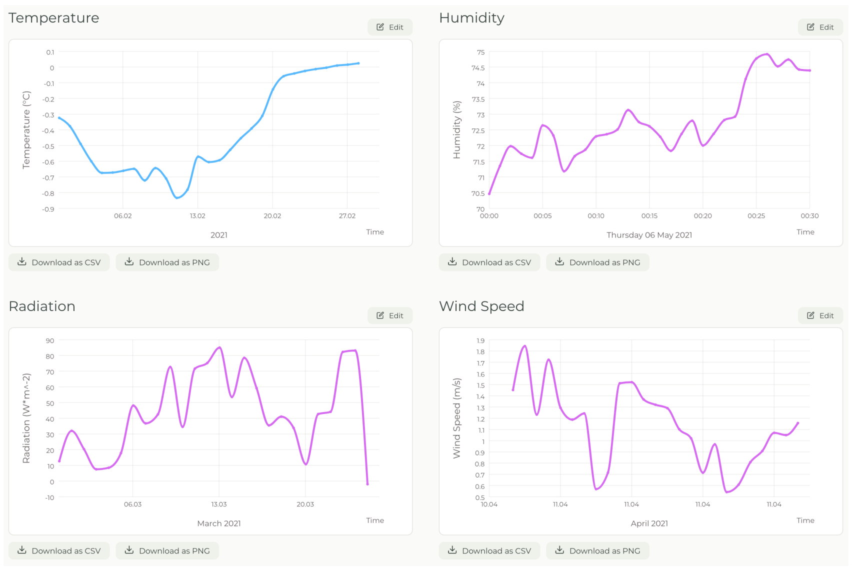
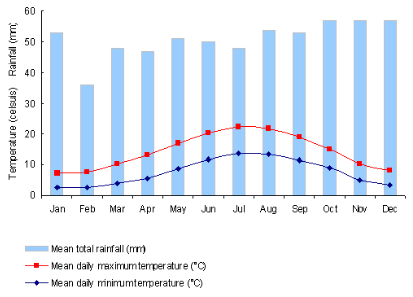


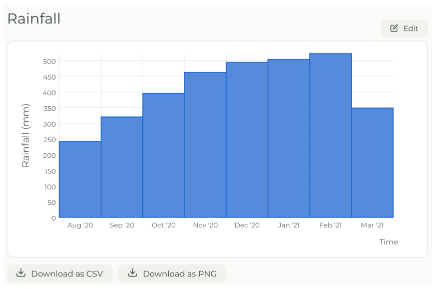
Figure 20.1: Different Sensor Data Visualized as Line Charts

Bar Charts

Bar charts are used to compare data and are especially effective when the data can be split nicely into categories that can be compared [21]. Rainfall data are usually always visualized in bar charts since it is an effective way to show how much it has rained during specific time periods.



(a) Climograph [75]



(b) Nykken: Bar Chart

Figure 20.2: Rainfall Data Visualized as Bar Charts

Climographs are used to represent rainfall and temperature data in one combine chart to quickly be able to describe the climate in one location [6]. The standard way of doing this is to visualize the temperature data as a line, while the rainfall data is visualized as bars. An example of this can be seen in Figure 20.2a, Nykken has used this standard when visualizing its rainfall data as bar charts.

Area Charts

When visualizing water level data in Nykken, it was first visualized using a regular line chart. However, after discussions with the main stakeholder, the area beneath the line was filled in with a semi-transparent color. This was done to separate this data from the rest of the data which were visualized using line charts, and also to better show that the data was representing the actual water level in the pipes.

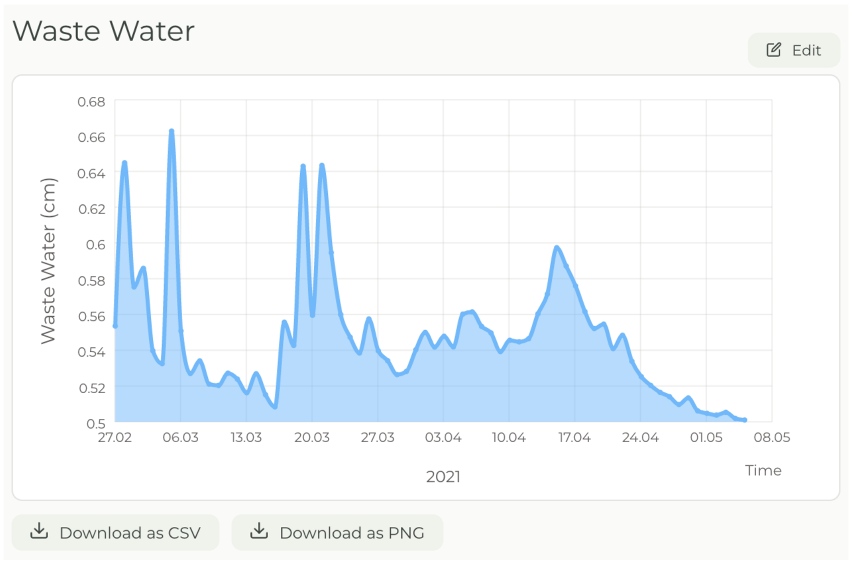


Figure 20.3: Waste Water as Area Chart

20.1.2 Chart Colors

The colors used in the Nykken charts were chosen with what sensor data the colors would represent in mind. When the concepts that were visualized had an obvious semantically resonant color available, it was chosen for this sensor. When the concepts did not have such a color available, other factors were taken into consideration.

Line Chart Colors

The line charts in Nykken are divided into two color groups: blue and purple. The temperature sensor measurements are visualized as blue line charts, while the rest of the sensors have their data visualized with purple lines. To accommodate users with potential color vision deficiencies, the purple color was set to have a lower color value, which makes it appears darker than the blue color. This was done to make the contrast between the colors visible whether the user could perceive color or not.

Originally, the plan for the temperature lines was to color them red and blue to represent positive and negative temperatures respectively, which is further discussed in Section 20.4.2. The basic premise being that red and blue colors are usually used to represent warmer and cooler temperatures since people tend to relate them together.

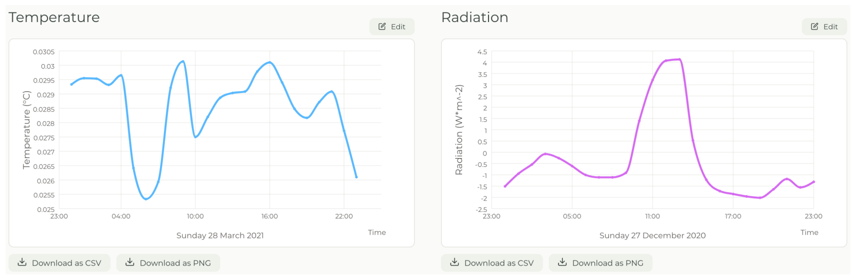


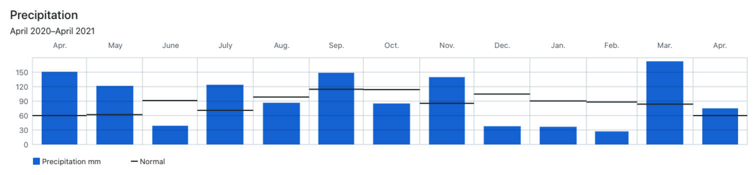
Figure 20.4: Line Chart Colors

The purple color for the rest of the sensors, that would use line charts, was not picked because of the specific meaning of the color. It was picked as a general color that did not have any particular connotations to it, which made it not more nor less suitable for any specific sensor. It was also a conscious decision to pick a color that would harmonize with the colors already picked for the application.

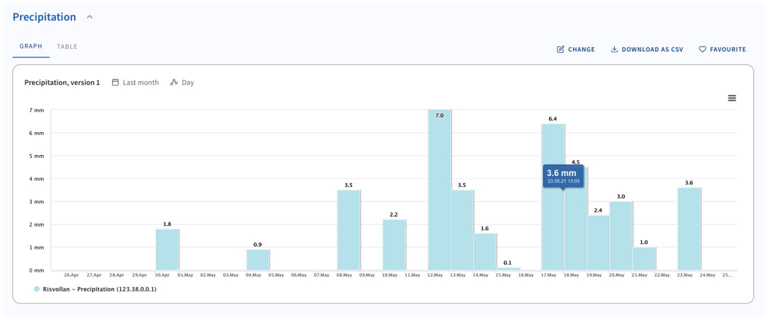
As previously mentioned, custom charts were originally supposed to be a feature in Nykken but ended up not being a part of the final solution. In the custom charts, it would be possible to combine multiple line charts together. In that case, it would be helpful to the user if each sensor had its own color, to easily be able to distinguish them from each other. We will later discuss the potential option for users to select their own colors in Section 20.3, but there should be a default value for all sensors that make sense for that particular sensor data.

Rainfall as Blue Bars

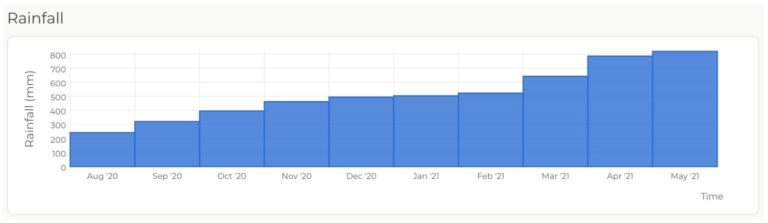
Since the bar charts were supposed to represent rainfall, it was a natural choice to use blue for these charts, since blue is a semantically resonant color for water [34]. This is also what Yr and Sildre have opted for, as seen in Figure 20.5. When choosing the hue of the color, it was opted for a true hue of blue, so that it would stand out from the background and be different enough from the more teal blue color of the temperature charts.



(a) Yr: Bar Chart



(b) Sildre: Bar Chart



(c) Nykken: Bar Chart

Figure 20.5: Bar Chart Colors

Red to Imply Faulty Data

Nykken does, not imply faulty data in its charts. This is something that the user has to uncover themselves when studying the charts. One exception is in the bar charts that represent rainfall data. It is obvious that rainfall data should not be negative, since the lowest amount of rainfall possible is none, or zero.

It was decided to show negative rainfall as red bars that go beneath the zero line of the chart, which can be seen in Figure 20.6. Red is a color that is often used to display error messages since the color itself attracts attention and is often used as a stop signal in traffic [34]. Because of this, the red bars could help signal to the user that there is something wrong with the data, and that it should be looked into.

In Figure 20.6, the rainfall value for each data point is nearly identical to the other data points, apart from the negative value. The reason for this is that the rainfall sensor only measures cumulative rainfall data. This means that the rainfall is measured by how much water there is in the rainfall tank at a certain point in time, and not how much it has changed over time.

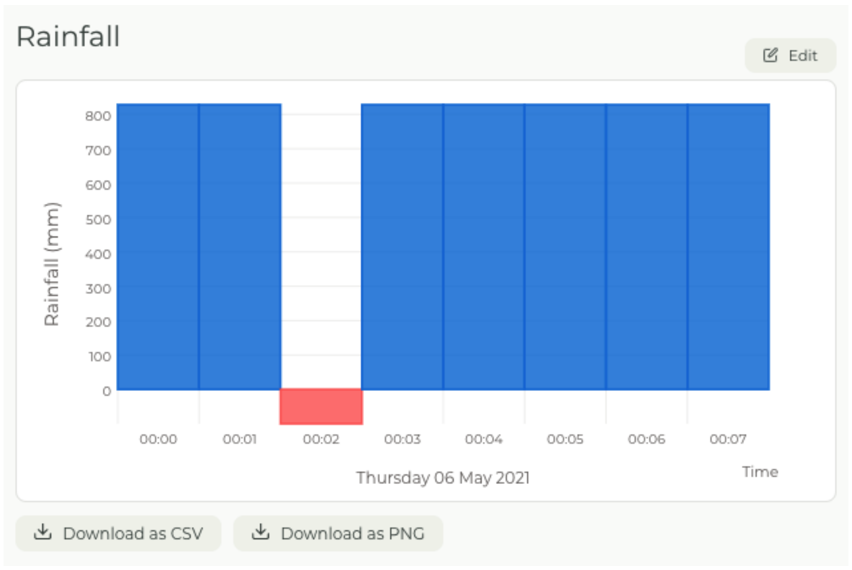


Figure 20.6: Negative Rainfall as Red Bars

20.2 Validation of Non-Functional Requirements

The validation of the non-functional requirements is presented in Table 20.1 with the same definitions of completion as the functional requirements, which are described in Section 17.2. The completion values were *Complete* (C), *Partly Complete* (PC), and *Not Complete* (NC).

ID	Description	Completion	Comments
N1	The application shall function and be available on modern browsers	PC	Nykken has been developed and tested in Chrome, Firefox, and Safari with some minor flaws to the design in Safari.
N2	The application shall be easy and intuitive to use, with minimal uncertainty	C	Through the user tests, the users found Nykken easy and intuitive to use.
N3	The application's main functionality shall be easily accessible, with minimal distractions or clutter	C	The users did not have major issues finding the functionality in the application.
N4	The application users shall find the application secure, without being unsure about personal information leaking without permission	C	There are no personal information that the user has to share with the application.
N5	The application shall have short response time when retrieving data for visualizations	PC	When retrieving data for visualizations, the loading time is short due to granularity restrictions.
N6	The application shall allow for extension of the API, for example adding more sensors	C	The application allows for extensions of the API as it should be easy to implement additional functionality.
N7	The application shall be available and not have much down time	PC	When the API is functioning as expected, the application is up and running without problems. It does however not have a good way of dealing with the API being down.

Table 20.1: Validation of Non-Functional Requirements

20.3 Future Development

In this chapter, the future development of the Nykken application will be discussed. This includes potential changes that could be added to improve the application. These changes are based upon feedback from the last user tests, as well as features that were planned to be implemented, but that fell through at the end of development. In addition to discussing the future work of the application, other factors which influence the way Nykken works will also be discussed. This includes work that is not necessarily the application itself, but work that could improve or expand the functionality that it provides.

20.4 Nykken

This section will discuss improvements that could be done in the actual application, Nykken. These improvements either deal with design and usability or with functionality.

20.4.1 Design and Usability

Responsive Charts

Nykken is designed to display multiple charts on the users' dashboard. Having responsive charts on the dashboard would be a great improvement to the application because it would mean that the charts would look great on a wider range of screen sizes.

Since the charts are built with SVG elements, they are not straightforward to make responsive. This is because the width and the height of the SVG elements have to be set in the HTML-document, and cannot be changed with simple CSS. During development, the focus was on implementing the core functionalities, which meant that the responsiveness of the charts was not prioritized. In a future version of Nykken, responsive charts would be a good improvement.

Improved Data Loading Time

When a user wants to display a lot of data measurements in a chart, the data could potentially take some time to load. This issue was partially solved with the introduction of granularity when selecting time frames for charts. The granularity options would change depending on how big the time frame was. This means that only "As measured" is shown if the time frame is within the same day. In an ideal version of Nykken, it would be possible to choose smaller granularities even when the time frame is bigger than one day.

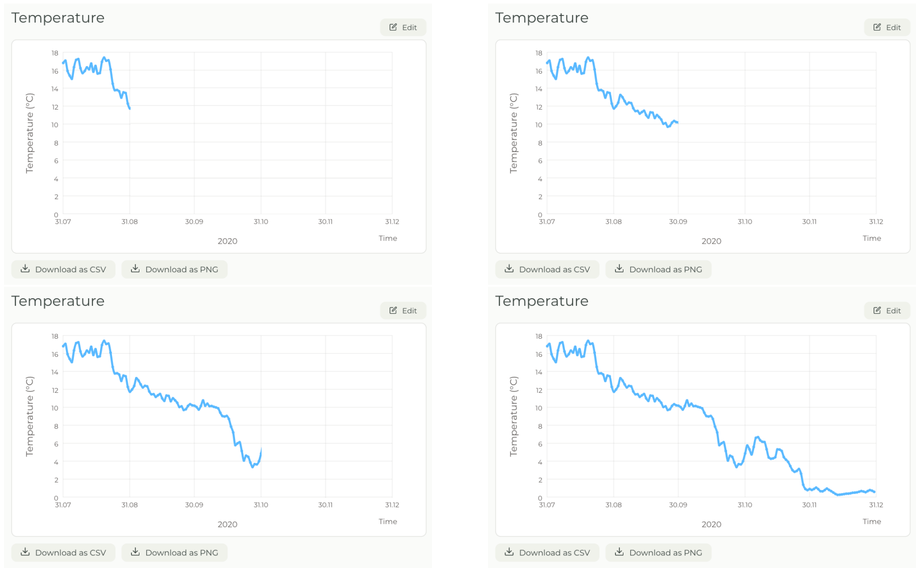


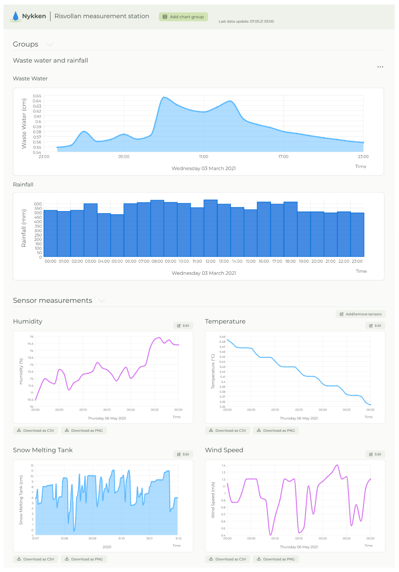
Figure 20.7: Potential Solution: Loading Chunks of Data

A potential improved solution to this problem would be to load chunks of data when sending a request to the API. This would mean that if the user selected a time frame of one year, chunks of one month, or one week, would be retrieved at a time. The data could then be displayed in the chart as it was being fetched. An example of how this could look can be seen in Figure 20.7.

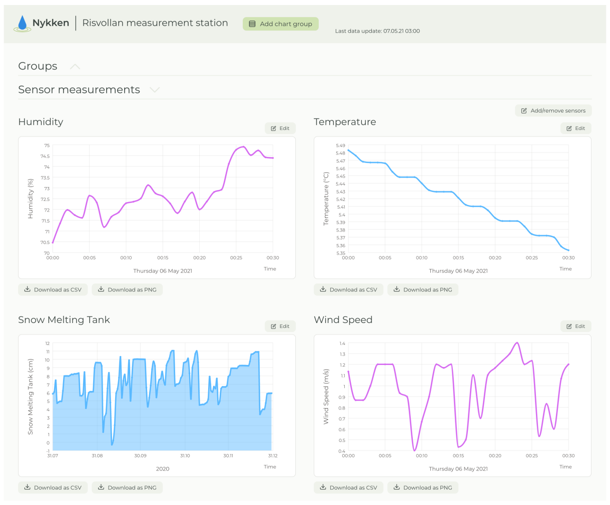
Collapsible Sections

When defining the non-functional requirements of the application, which are located in Section 11.2, one of the requirements read "The application's main functionality shall be easily accessible, with minimal distractions or clutter".

Since the users have no limitations of how many groups or custom charts they can add, the dashboard could easily become crowded and not as easy to navigate. One possible solution to this problem could be to make the sections *Sensor measurements*, *Groups*, and *Custom charts* collapsible, which would mean that the user could hide and show the charts in the sections to free up space on the dashboard. An example of how this could look can be seen in Figure 20.8.



(a) Before Collapsing Groups

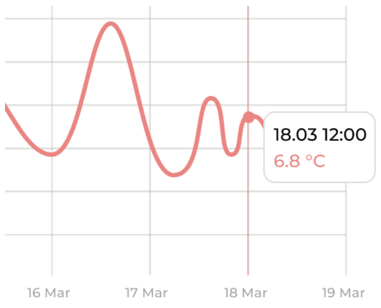


(b) After Collapsing Groups

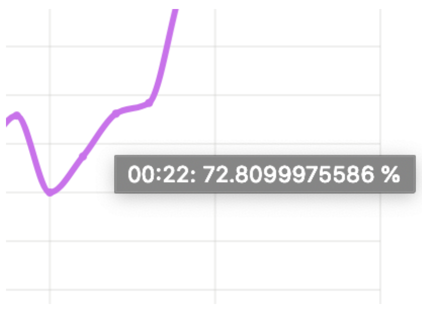
Figure 20.8: Collapsible Sections

Improved Hover Details

Hover details was a feature that was included in all the sketches as well as in the prototypes. Hover details were implemented in Nykken but were not as detailed and user-friendly as in the prototype. Creating hover details similar to the prototype's version would be an improvement to the usability of this feature. The comparison between the prototype's hover details and the actually implemented hover details can be seen in Figure 20.9.



(a) Prototype: Hover Details



(b) Nykken: Hover Details

Figure 20.9: Hover Details Comparison

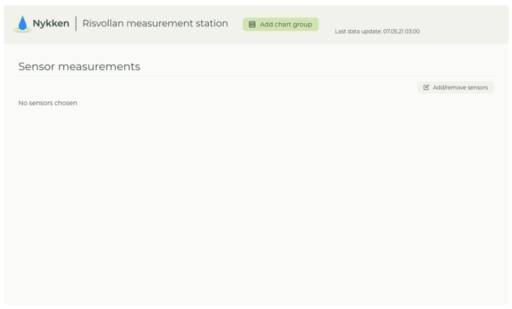
General Responsiveness

While the application has basic responsiveness implemented, such as a responsive chart grid and a responsive header, improvements can be made. One specific part of the web page that should be more responsive, is the font sizes. When the users open the web page on a bigger screen, the text tends to get too small. This should be improved in a future version of Nykken.

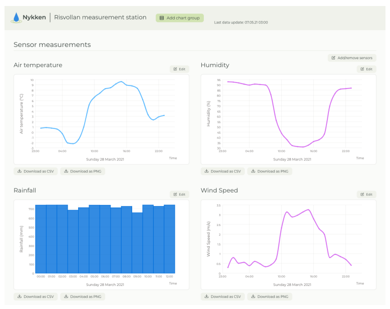
20.4.2 Features

Default Charts

When users visit Nykken for the first time, the original plan was to have a set of default charts that would display the most current data for some selected sensors. This was a feature that was not implemented during development but is a feature that could improve the user experience. A good idea would be to curate this set of charts based on feedback from the users about which sensor data they were most interested in. An example of how this could look can be seen in Figure 20.10.



(a) Current Default Dashboard



(b) Default Dashboard with Charts

Figure 20.10: Default Dashboards

Last Data Update

In the header of the web page, the "last data update"-timestamp is updated manually based on information from the person responsible for the API. Since it is not effective to keep manually updating this date every time there is new data in the API, this should be done automatically by checking the API for data and adjusting the "last data update"-timestamp.

Expanded Granularity Options

If the user wants to change the time frame of the data in a chart, they also have to select a granularity option, which decides how many measurements will be shown in the chart. To prevent site crashes, the granularity options are limited depending on the dates chosen.

These are the set granularity options available:

Time stamps	**Granularity options**
Same hour	As measured
Same day, different hour	As measured, hourly
Same week, different day	Hourly, daily
Same month, different week	Daily, weekly
Same year, different month	Daily, weekly, monthly
Different year	Monthly, yearly

Table 20.2: Currently Available Granularity Options for each Time Frame

In some cases, the granularity options are too limited, for example when two days are within the same week but are in different months, the *hourly* option will not be available. This is because the application checks for the "largest" entity that is different. An ideal solution would be to have as many granularity options available as possible for the time frame chosen. For example, data with timestamps from different years should be able to show weekly and daily measurements up to a certain number of years.

Handling No Data Available

There are currently no time restrictions for how far back in time the user can go when selecting start and end timestamps for the data measurements. This means that if the users go back far enough, there will be no data available. When Nykken retrieves no data from the API when trying to display data in a chart, the chart will go into an infinite loading, or the site will crash. This is a crucial flaw that has to be fixed. One way of solving this is to set a time restriction, preventing the users from choosing data outside the time frame the API has available data. Ideally, the website will let the user know that there are no data available for the time frame chosen.

Improved PNGs

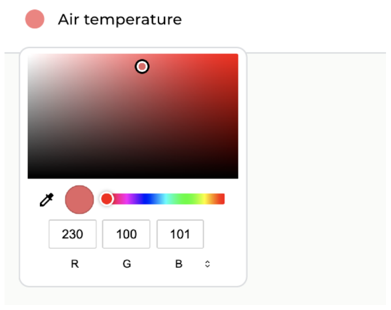
If the users at some point want to download their charts as a PNG image to put into their presentations or papers, they can click the "Download as PNG" button which is attached to each chart. This functionality is based upon exporting React components. Since the charts are defined as a React component with SVG elements, this is what will be exported when the user clicks this button.

The resolution of the image is determined by the width and the height of the component in pixels. This means that if the chart component itself is small, the image will be equally as small as the component, which in Nykken results in a PNG image with lower quality than desirable. During the second user test, which is described in Subsection 18.3.2, the users commented how the images had to have a certain pixel density, more accurately 300 PPI (Pixel per inch) if they were to use it in a paper. Since the images exported by Nykken do not fulfill this requirement, it is something that should be improved in a future version of the application.

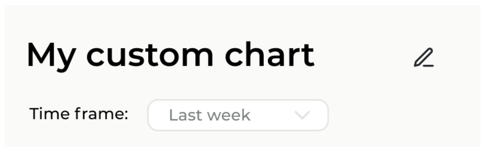
Some users also commented that it would be nice to be able to download a PDF version of the chart as well, so this is something that could be considered.

Customizable Charts

A feature that was requested by the users was the option of customizing charts. Especially for downloading or adding custom charts to the dashboard, this is a feature that could be very useful. The customization could include customizing the axis' titles, graph colors, and adding custom chart names. Examples of potential customization can be seen in Figure 20.11.



(a) Custom Graph Color

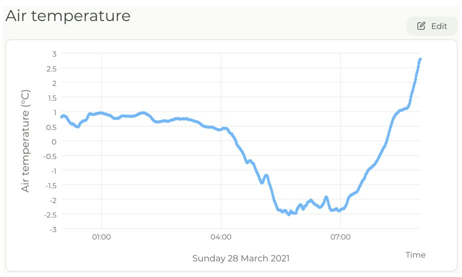


(b) Custom Chart Name

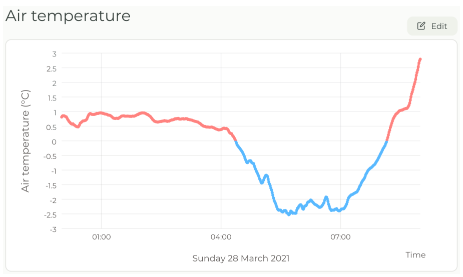
Figure 20.11: Customization

Colored Temperature Lines

One type of sensor data that Nykken visualizes, is temperature data. When examining other applications that visualized this type of data, the common practice was to visualize negative temperature values with the color blue, while positive temperature values were visualized with the color red. Initially, this was how Nykken was supposed to visualize these types of values as well, but it was not a feature that was prioritized to implement. The color of the temperature charts ended up being colored blue in the final version of the application. Since this is not common practice, this is something that should be added in a future update. This could be done as shown in Figure 20.12b.



(a) Current Temperature Chart



(b) Colored Temperature Chart

Figure 20.12: Temperature Charts

Custom Chart

Creating custom charts was a feature that was originally planned to be released in this version of Nykken. However, this feature was not finished by the end of development. Since this was a highly requested feature by the users, it should eventually be a part of the application. An example of how the feature could look in the future can be seen in Figure 20.13

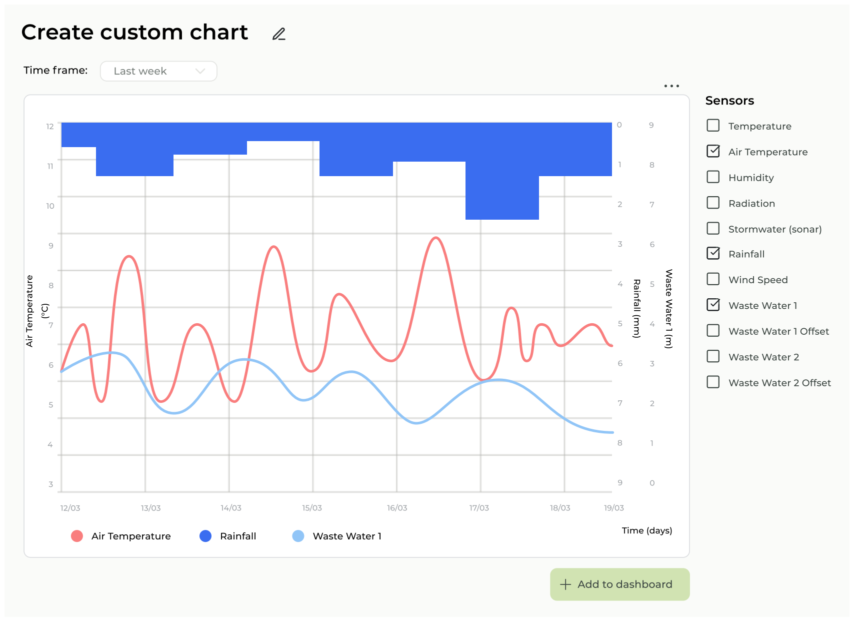


Figure 20.13: Custom Chart Functionality

Color Legends

As seen in Figure 20.13, the custom chart functionality was supposed to come with color legends. In the mock-ups, all the color legends are similar, except for their color. When testing the prototype, it was commented that the color legends could instead reflect what type of chart they represented, to make it clearer. Data visualized with line charts could have lines as their color legends, while data visualized with bar charts could have a square as a color legend. Yr provides a great example of different color legends in their combined weather chart, which an excerpt of can be seen in Figure 20.14.

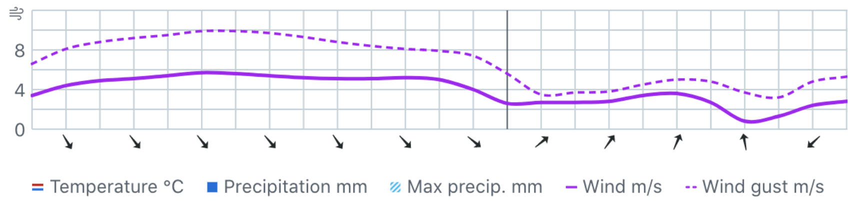


Figure 20.14: Yr: Different Color Legends

Compare Same Sensor Data

A feature that was mentioned by users during the interviews, was the ability to compare data from the same sensor. Nykken currently provides comparisons only between data from separate sensors. Ideally, it would be possible to compare for instance rainfall measurements from separate time frames like months or years. This could potentially be solved by allowing the same sensor to be added in a custom chart or sensor group multiple times with different time frames. Since this is not currently possible, it is something that could be considered in the future.

20.5 Related Work

This section will discuss improvements that could be done in the API and as well as other additions that would require insight into the hydrological field.

20.5.1 The API

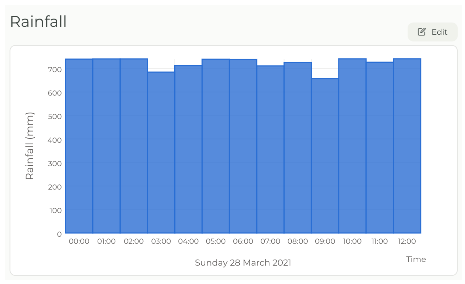
Since Nykken is based upon the functionality of the API, some extensions could be added to the API, which in return could expand the functionality of Nykken.

Real Time Data

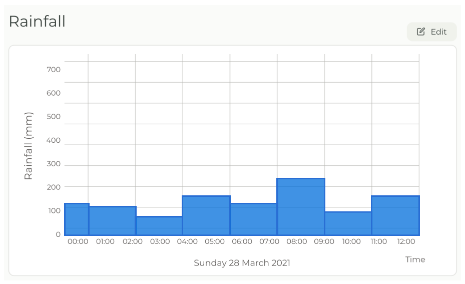
The API does not currently provide real-time data, which is a feature that could be useful for multiple users. At the moment of writing, there is no standard as to how often the data is updated, but this could change in the future.

Improved Rainfall Visualization

Currently, the rainfall measurements are cumulative in the API. This means that the actual rainfall does not appear how the user might expect it to in the chart. Instead of showing rainfall as precipitation from a specific time frame, it measures the accumulated rainfall in the rainfall gauge. To better visualize the rainfall, the sensor data should only measure the increase. Ideally, it should be possible to show both the cumulative data and the subtractive data in the application, as shown in Figure 20.15.



(a) Cumulative Rainfall Data



(b) Subtractive Rainfall Data

Figure 20.15: Rainfall Visualizations

Updated Sensor Titles

When the user is selecting sensors to display on the dashboard or combine them in groups, the sensors are visualized as checkboxes with names. These names are retrieved from the API as sensor titles. Initially, the sensors only had a description, but titles were introduced as a way of presenting the sensors clearly and concisely.

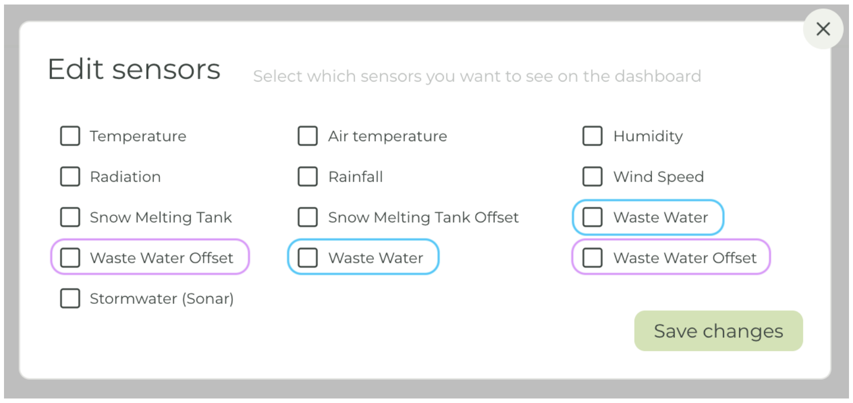


Figure 20.16: Current Sensor Titles

When the titles were added, multiple sensors got the same title, even though the sensor was different. The sensors in question were the sensors with the descriptions *Water level 2, waste water* and *Water level 3, waste water*, which got the same title *Waste Water*, and *Water level 2, offset* and *Water level 3, offset*, which got the same title *Waste Water Offset*. To the user, these look identical to each other in the UI, as can be seen in Figure 20.16. A solution to this would either be to always include the description as well as the title in the UI, or update the titles in the API so that they are different.

Include Older Sensor Data

When the API was first deployed, there was a set date where the sensor data was starting to feed into the API. Even though this date was the first date where the API received data, it is not the first date to have sensor measurements from Risvollan Measurement Station. There are sensor data from back to the '80s, which should be included in the API.

20.5.2 Expert Insights

The following improvements would rely on the insights of a hydrology expert since it includes features that could not be implemented without knowledge about hydrological data.

Error Detection

During the interviews, multiple users mentioned that it would be helpful to have a way to tell if there was something wrong with the data. A well-rounded error detection feature would require insight into hydrological data, which we did not have. Although some basic error detection could be done without insight, such as negative numbers for rainfall, it was not a feature that was prioritized. Another student at NTNU has been working on a bachelor's thesis about error detection of hydrological data parallel to this project, which could potentially be merged into Nykken in the future.

Specific Events

It was also mentioned in one of the interviews that it would be interesting to be able to capture specific weather events in the application, such as a rain event. To be able to visualize these specific events, what would be considered an *event* would have to be defined. Since a task like this would again require insight that we did not have, this was not a feature that was prioritized. It is something that could be implemented in the future, with an amount of research into this specific subject.

Reference Points

Reference points can bring useful context to charts, which can help the user interpret the charts more effectively. The introduction of reference points could be done using insights about what "normal" or "average" values for each sensor should be. If these references were ultimately a part of the application, it is a feature that could give the user additional value, since it would help put the sensor data in a context.

Chapter 21

Evaluation of the Project

21.1 Fulfillment of the Research Questions

This chapter will present the fulfillment of the two research questions, as well as the project goal.

21.1.1 Research Question 1

In what ways can interactive visualizations improve/aid the analysis of hydrological data?

The results showed that one of the pain points the WWSE-group had, was gaining access to the sensor data, as well as a lack of consistency in the data in regards to file format and type. With Nykken, they now have easy access to the data, and it is downloadable as a CSV file, all in the same format.

The participants also stated that they usually visualized the data they wanted to use, to get an overview and insight into the data before they started working with it. Nykken provides a simple chart of the data they want to download. Providing this visualization to the users before they download it, can save the users time because this lets them know if the data is available and usable before they download it.

By using visual aids, such as colors, warnings, and reference points, the visualization can help the users process the data faster and more confidently. This is especially helpful to users who are unfamiliar with hydrological data, as they require more sensory information to interpret the charts [36].

The visualization offers interactivity through the time frame and granularity picker. This lets the users choose if they want to see long data series with average values or shorter time frames with more detailed measurements.

21.1.2 Research Question 2

Which kinds of interactive visualizations of such hydrological data are better suited for the different sensor measurements? Through our preliminary study, described in Section 5.2, we found that there are different standards and motivations as to how data should be visualized [32]. By interviewing our user group, experts, and looking at other visualizations of hydrological data, we found that hydrological data in Norway is visualized using conventional charts. The rainfall data, which is accumulated data in a given time frame, is shown in a bar chart. Instantaneous measurements, such as rainfall and humidity, are shown as line charts. The flow data is shown using area charts, with a shaded area between the line graph and the horizontal line, to give the illusion of water levels.

One visualization that appears to be a standard in the WWSE-group, is the upside-down rainfall bars in combined charts. When comparing rainfall data to other data, the participants from the user group stated that they preferred having the rainfall bars on top of the chart, facing downwards. This is in contrast to how Yr presents it [14].

To facilitate comparisons and correlation detection between different sensor data, a combined chart with the sensor data in question is a common approach. It is also possible to stack charts on top of each other, with the same time frame and intervals on the x-axis, for a less cluttered design.

If the participants wanted to use visualizations in a paper or presentation, most participants stated that they would prefer creating their own charts, using a tool they already knew, such as R, Python, or Excel. As of now, Nykken does not provide many customization options regarding chart design. There are no available options to change the graph colors or axis labels, something the other visualization tools offer. If Nykken were to add such customization options, a reiteration of user testing should be conducted, as well as a follow-up study to see if more users started using Nykken as their preferred visualization tool.

21.2 Fulfillment of the Project Goal

The goal was divided into the research questions which were discussed above. The final testing gave data and results that suggest that the users appreciate having easy access to the sensor measurements from RMS.

Even though research shows that visualization can help with data analysis, and our research has found effective ways to display data, a more detailed examination should be performed in the future, to see how the users interact with and use the available visualizations at Nykken.

Nykken fulfills the project goal. However, as discussed in Chapter 20, additional work and features could improve the quality of the solution.

Part VII

Conclusion and Future Work

In this final part, the conclusion of the project will be presented. Lastly, the future work chapter will discuss recommendations for further work.

Chapter 22

Conclusion

In this thesis, we aimed at creating a new application that would provide the WWSE-group access to the RMS data as well as examining if and how data visualization could be of use to them. To do this, visualization theory was studied. Different approaches to data visualization were explored, along with common practices within the hydrological field. Interviews with both users and experts also contributed to this research.

A prototype of a new visualization application has been created and tested. The results from the user tests have been analyzed, giving an indication of the application's usability and its features' relevancy.

The new visualization application, Nykken, gives the users access to the sensor data from RMS. It lets the user choose which sensor data they want to interact with and gives them a simple overview of the sensor data in a given time frame, set by the user. This allows them to check the quality and status of the data before they decide if they want to download it or not.

Research question 1 asked how interactive visualizations can improve data analysis. The results gathered from the data generation methods illustrated that visualizing data is an effective way to get an overview of the data you want to analyze.

Research question 2 asked which kinds of interactive visualizations were better suited to display the station measurements. The results found that there are standards for how each sensor data should be visualized, based on both general standards as well as disciplinary ones. Combined or stacked charts can be used to compare and look for correlations in the data.

The goal of this project was to develop and test an application that gave the WWSE-group access to the RMS data, which was accomplished by creating Nykken. To confidently identify and measure improvements in the WWSE-group's workflow, additional work and further research will be needed. However, the preliminary results indicate that Nykken is of value to the WWSE-group.

Chapter 23

Future Work

This chapter will present the potential future work of the project. This includes what could be done in the future to continue the research and improve the final solution even further.

23.1 Implementation

As discussed in Section 20.3, improvements can be added to the charts currently presented in Nykken. A second development phase should therefore be considered, to implement the suggested improvements.

An expert on hydrological data should also be included to help with the implementation of reference points and to define what constitutes a specific event.

23.2 Testing

Since our user tests mainly tested the general usability of Nykken, rather than the visualization design, new tests should be executed to better examine how the users perceived the data visualization.

With further iterations, it would be helpful to introduce additional test subjects. The pool of test subjects was limited during this project, and it would therefore be especially helpful to expand the participation group to gain more insight.

One motivation for using Nykken, stated by professors, was to use it in educational circumstances, such as in lectures and assignments. They wanted to use Nykken to allow for more advanced analysis tasks related to hydrological data. An interactive chart could also be used to give each student their own tasks, to prevent students from copying each others' work. According to the technician, there are reportedly also multiple students using the measurement data in their papers and assignments. More students should therefore be included as test subjects in future user tests.

The last user group that should be included in future user tests, is the technicians in the WWSE-group. Since they were unable to attend any of the user tests, it is not possible to draw any conclusions as to how Nykken accommodates this user group's needs.

A follow-up study should also be considered, to explore how the users interact with Nykken in everyday life would be especially interesting to examine if the use of Nykken would improve the WWSE-group's workflow and if they interact with the data in any new ways.

Bibliography

[1] Alper Aydin. *Interactive Visualization with Dash and Plotly.* Accessed: 27 May 2021. 2019. URL: https://towardsdatascience.com/interactive-visualization-with-dash-and-plotly-29eaccc90104/.

[2] Gerald Benoît. *Introduction to Information Visualization: Transforming Data into Meaningful Information.* Rowman Littlefield, 2019.

[3] David Butler and John W. Davies. *Urban Drainage 3rd Edition.* Spon Press, 2011.

[4] Chart.js. *Chart.js API.* Accessed: 02 October 2020. 2020. URL: https://www.chartjs.org/docs/latest/developers/api.html?.

[5] Chart.js. *Installation.* Accessed: 02 October 2020. 2020. URL: https://www.chartjs.org/docs/latest/getting-started/installation.html.

[6] Artem Cheprasov. *Climograph: Definition & Uses.* Accessed: 24 May 2021. 2021. URL: https://study.com/academy/lesson/climograph-definition-uses.html.

[7] Climate.gov. *Data Snapshots: Easy access to climate data, products, and services.* Accessed: 29 May 2021. URL: https://www.climate.gov/maps-data/data-snapshots/averagetemp-monthly-cmb-2021-03-00.

[8] Sue Coles and Jenny Rowley. "Creating Effective Graphs and Charts". In: (1997).

[9] D3.js. *Data-Driven Documents.* Accessed: 15 December 2020. 2020. URL: https://d3js.org/.

[10] *Department of Civil and Transport Engineering Water and wastewater systems engineering.* Accessed: 30 May 2021. URL: https://www.ntnu.edu/ibm/water-and-wastewater-engineering.

[11] Cambridge Dictionary. *Chart.* Accessed: 30 May 2021. URL: https://dictionary.cambridge.org/dictionary/english/chart.

[12] Andrew Disney. *Choosing colors for your data visualization.* Accessed: 24 May 2021. 2020. URL: https://cambridge-intelligence.com/choosing-colors-for-your-data-visualization/.

[13] Petter L. H. Eide. *Quantification and Traceability of Requirements.* Faculty of Information Technology, Mathematics, Electrical Engineering Department of Computer, and Information Science, 2005.

[14] Thor Gjermund Eriksen. *Om Yr.* Accessed: 03 May 2021. 2021. URL: https://hjelp.yr.no/hc/no/articles/206550539-Fakta-om-Yr.

[15] Steven Few. "The Encyclopedia of Human-Computer Interaction, 2nd Ed.," in: ().

[16] Figma. *Figma: the collaborative interface design tool.* Accessed: 13 May 2021. 2021. URL: https://www.figma.com/.

[17] Interaction Design Foundation. *What are User Scenarios?* Accessed: 3 March 2021. 2021. URL: https://www.interaction-design.org/literature/topics/user-scenarios.

[18] *Good enough to great A quick guide for better data visualizations*. Accessed: 24 May 2021. 2006. URL: https://www.tableau.com/good-to-great.

[19] John R. Goodall. "Visualization is better! A comparative evaluation". In: (2009).

[20] Guru99. *What is a Functional Requirement in Software Engineering? Specification, Types, Examples*. Accessed: 8 March 2021. URL: https://www.guru99.com/functional-requirement-specification-example.html.

[21] Maila Hardin et al. *Which chart or graph is right for you? Tell impactful stories with data*. Accessed: 24 May 2021. 2012. URL: https://theathenaforum.org/sites/default/files/WHich%5C%20chart%5C%20is%5C%20right%7B%7D%5C%20for%5C%20you.pdf.

[22] Robert L. Harris. *Information Graphics: A Comprehensive Illustrated Reference*. Management Graphics, 1996.

[23] *How many participants do you need for usability testing?* Accessed: 01 March 2021. URL: https://www.bellomy.com/blog/how-many-participants-do-you-need-usability-testing.

[24] *How to Determine the Right Number of Participants for Usability Studies*. Accessed: 01 March 2021. URL: https://www.uxmatters.com/mt/archives/2016/01/how-to-determine-the-right-number-of-participants-for-usability-studies.php.

[25] *hydrologi*. Accessed: 30 May 2021. URL: https://snl.no/hydrologi.

[26] GitLab Inc. *About GitLab*. Accessed: 28 April 2021. URL: https://about.gitlab.com/company/.

[27] Netlify Inc. *About Netlify*. Accessed: 1 May 2021. URL: https://www.netlify.com/about/.

[28] *Instrumentation at RUHS*. Accessed: 30 May 2021. URL: http://www.ivt.ntnu.no/ivm/risvollan/Instruments.html.

[29] Diane J.Janvrina, Robyn L.Raschkeb, and William N.Dillaa. "Making sense of complex data using interactive data visualization". In: (2014).

[30] Curran Kelleher. *Datavis 2020*. Accessed: 22 December 2020. 2020. URL: https://datavis.tech/datavis-2020/.

[31] Henrik Kniberg. *Scrum and XP from the Trenches - 2nd Edition*. C4Media, publisher of InfoQ.com, 2015.

[32] Charles Kostelnick. "Conflicting standards for designing data displays: Following, flouting, and reconciling them". In: (2012).

[33] Athony Ladson. "Visualising Hydrologic Data". In: (2018).

[34] Sharon Lin and Jeffrey Heer. *The Right Colors Make Data Easier To Read*. Accessed: 25 May 2021. 2014. URL: https://hbr.org/2014/04/the-right-colors-make-data-easier-to-read.

[35] LogRocket. *Data visualization in React using React D3*. Accessed: 15 December 2020. 2020. URL: https://www.youtube.com/watch?v=YKDIsXA4OAc.

[36] Saul A. McLeod. "Information processing. Simply Psychology". In: (2008).

[37] Vibhu O. Mittal. "Visual Prompts and Graphical Design: A Framework for Exploring the Design Space of 2-D Charts and Graphs". In: (1997).

[38] Robert Mundigl. *An Underrated Chart Type: The Band Chart*. Accessed: 24 May 2021. 2011. URL:

https://www.clearlyandsimply.com/clearly_and_simply/2011/04/an-underrated-chart-type-the-band-chart.html.

[39] Jacob Nielsen. *How Many Test Users in a Usability Study?* Accessed: 01 March 2021. URL: https://www.nngroup.com/articles/how-many-test-users/.

[40] Nivo. *About*. Accessed: 14 October 2020. 2020. URL: https://nivo.rocks/about.

[41] NVE. *NVEs historie*. Accessed: 8 June 2021. 2021. URL: https://www.nve.no/om-nve/vassdrags-og-energihistorie/nves-historie/.

[42] NVE. *Om Sildre*. Accessed: 22 February 2021. 2021. URL: https://beta-sildre.nve.no/help.

[43] Briony J. Oates. *Researching Information Systems and Computing*. SAGE, 2006.

[44] Open Source Initiative. *Apache License, Version 2.0*. Accessed: 30 May 2021. 2004. URL: https://opensource.org/licenses/Apache-2.0.

[45] Open Source Initiative. *The 3-Clause BSD License*. Accessed: 30 May 2021. 2021. URL: https://opensource.org/licenses/BSD-3-Clause.

[46] Open Source Initiative. *The MIT License*. Accessed: 30 May 2021. 2021. URL: https://opensource.org/licenses/MIT.

[47] Color Universal Design Organization. *Color Universal Design Handbook*. Accessed: 26 May 2021. EIZO NANAO CORPORATION, 2006.

[48] Arif Perdana, Alastair Robb, and Fiona Rohde. “Does Visualization Matter? The Role of Interactive Data Visualization to Make Sense of Information”. In: (2018).

[49] Arif Perdana, Alastair Robb, and Fiona Rohde. “Does Visualization Matter? The Role of Interactive Data Visualization to Make Sense of Information”. In: (2018).

[50] Generated Photos. *Unique real-time face generator*. Accessed: 3 March 2021. 2021. URL: https://generated.photos/face-generator/.

[51] React. *Create a New React App*. Accessed: 24 May 2021. 2021. URL: https://reactjs.org/docs/create-a-new-react-app.html#create-react-app.

[52] React. *React - A JavaScript library for building user interfaces*. Accessed: 22 February 2021. 2021. URL: https://reactjs.org/.

[53] React-csv. *react-csv*. Accessed: 1 April 2021. URL: https://github.com/react-csv/react-csv.

[54] Max Rehkopf. *User Stories with Examples and Template*. Accessed: 9 March 2021. URL: https://www.atlassian.com/agile/project-management/user-stories.

[55] Mark Richards. *Software Architecture Patterns*. O’Reilly Media, Inc., 2015.

[56] Teo Yu Siang Rikke Friid Dam. *Personas - A Simple Introduction*. Accessed: 2 March 2021. 2020. URL: https://www.interaction-design.org/literature/article/personas-why-and-how-you-should-use-them.

[57] Tobias Rolfes, Jürgen Roth, and Wolfgang Schnotz. “Effects of Tables, Bar Charts, and Graphs on Solving Function Tasks”. In: (2017).

[58] Im-salman. *react-component-export-image*. Accessed: 1 April 2021. URL: https://github.com/im-salman/react-component-export-image#readme.

[59] Ward E. Sanford and David L. Selnick. “Estimation of Evapotranspiration Across the Conterminous United States Using a Regression with Climate and Land-Cover Data”. In: (2012).

[60] Jeff Sauro. *Measuring Usability With the System Usability Scale (SUS)*. Accessed: 08 March 2021. 2011. URL: https://measuringu.com/sus/.

[61] Ken Schwaber and Jeff Sutherland. “The Scrum Guide”. In: (2020).

[62] Samuel Setsoafia. *Nivo - a GREAT alternative to d3 in react*. Accessed: 14 October 2020. 2018. URL: `https://medium.com/@samuelsetsoafia/nivo-a-great-alternative-to-d3-in-react-6cb18d907d2`.

[63] Sisense. *Interactive Data Visualization*. Accessed: 27 May 2021. 2021. URL: `https://www.sisense.com/glossary/interactive-visualization/`.

[64] Dag Svanæs. *Brukertesting*. Slides from Blackboard, IT3402 NTNU. 2019.

[65] Adobe Systems. *Gorgeous graphics. Absolutely anywhere.* Accessed: 1 April 2021. URL: `https://www.adobe.com/products/illustrator.html`.

[66] Dejan Todorovic. *Gestalt Principles*. 2008. URL: `http://www.scholarpedia.org/article/Gestalt_principles?__hstc=77520074.36a0ddae8e24bce7`.

[67] Toptal. *Colorblind Web Page Filter*. Accessed: 26 May 2021. 2021. URL: `https://www.toptal.com/designers/colorfilter`.

[68] The Coding Train. *1.3: Graphing with Chart.js - Working With Data APIs in JavaScript*. Accessed: 02 October 2020. 2019. URL: `https://www.youtube.com/watch?v=5-ptp9tRApM`.

[69] Edward R. Tufte. *the visual display of quantitative information*. Graphic Press, 1983.

[70] Unsplash. *The internet's source of freely-usable images.* Accessed: 3 March 2021. 2021. URL: `https://unsplash.com/`.

[71] *User Centered Design*. Accessed: 09 March 2021. URL: `https://www.interaction-design.org/literature/topics/user-centered-design`.

[72] VG. *Coronaviruset*. Accessed: 22 February 2021. 2021. URL: `https://www.vg.no/spesial/corona/?utm_source=coronav-new-front`.

[73] Victory. *Getting Started with Victory*. Accessed: 11 November 2020. 2020. URL: `https://formidable.com/open-source/victory/docs/`.

[74] Jed Watson. *React Select*. Accessed: 1 April 2021. URL: `https://react-select.com/home`.

[75] Wikimedia Commons. *File: London-climate.png*. Accessed: 24 May 2021. 2021. URL: `https://commons.wikimedia.org/wiki/File:London-climate.png`.

[76] Yr. *Getting Started*. Accessed: 03 May 2021. 2021. URL: `https://developer.yr.no/doc/GettingStarted/`.

Appendix A

Interview Guide

Current Situation

- How would you describe the current workflow when working with the sensor data?
- What are your main tasks?
- Which measurements/what sensor data do you interact with?
- Which time frames do you work with?
- Do you look for historical overviews, or more detailed measures?

Existing Solutions

- What do you do in order to make your data/measurements usable for your tasks?
- Which visualization tools have you previously used?
- Which features did these visualization tools have?
- Which functionalities did you use and why did you use them?

Features and Functionalities

- Which aspects of the data are you interested in?
- How do you want the system to visualize the different types of data?
- What types of data could be interesting to compare in the system?
- Which measurements would you want to see in the same graph/simultaneously?
- Which correlations are relevant to your work?

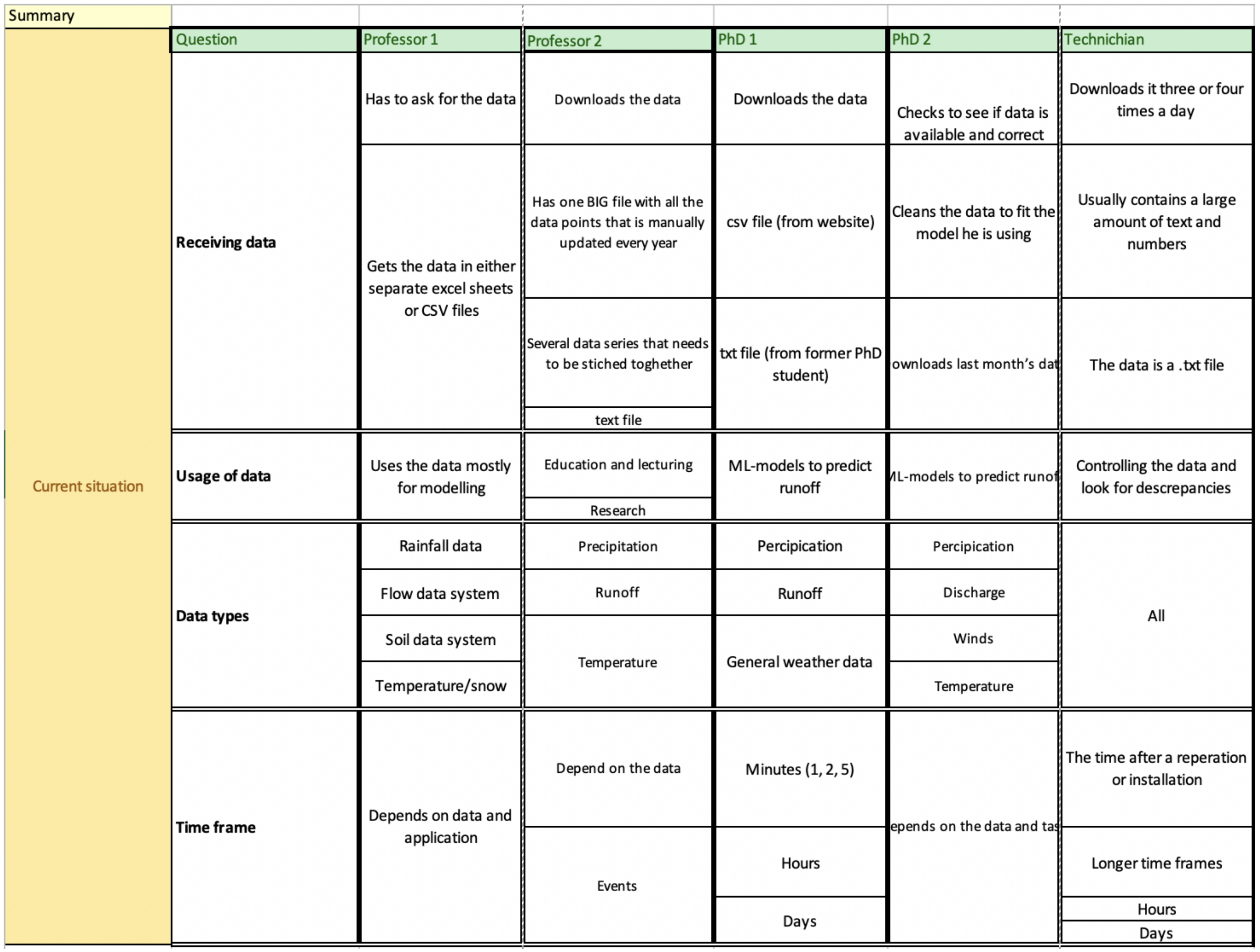
Motivation and Expectations

- What aspects do you consider important in a data visualization system?
- Which functionality is relevant to you when working with the data?
- What is the main functionality you want the system to include?
- In what ways do you want the system to improve your workflow?

Appendix B

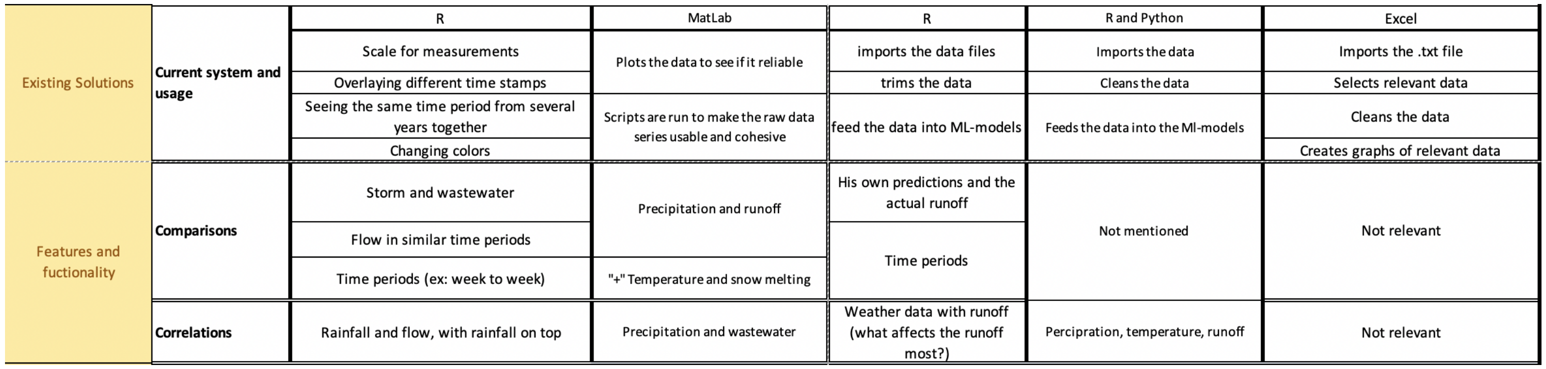
Interview Summary Matrix

The interview summary matrix can be found on the next page.



Summary	Question	Professor 1	Professor 2	PhD 1	PhD 2	Technichian
Current situation	Receiving data	Has to ask for the data	Downloads the data	Downloads the data	Checks to see if data is available and correct	Downloads it three or four times a day
		Gets the data in either separate excel sheets or CSV files	Has one BIG file with all the data points that is manually updated every year	csv file (from website)	Cleans the data to fit the model he is using	Usually contains a large amount of text and numbers
			Several data series that needs to be stiched toghether	txt file (from former PhD student)	ownloads last month's dat	The data is a .txt file
			text file			
	Usage of data	Uses the data mostly for modelling	Education and lecturing	ML-models to predict runoff	L-models to predict runof	Controlling the data and look for descrepancies
			Research			
	Data types	Rainfall data	Precipitation	Percipication	Percipication	All
		Flow data system	Runoff	Runoff	Discharge	
		Soil data system	Temperature	General weather data	Winds	
		Temperature/snow			Temperature	
	Time frame	Depends on data and application	Depend on the data	Minutes (1, 2, 5)	epends on the data and tas	The time after a reperation or installation
			Events	Hours		Longer time frames
				Days		Hours
						Days

Figure B.1: Interview Results Matrix: Current Situation

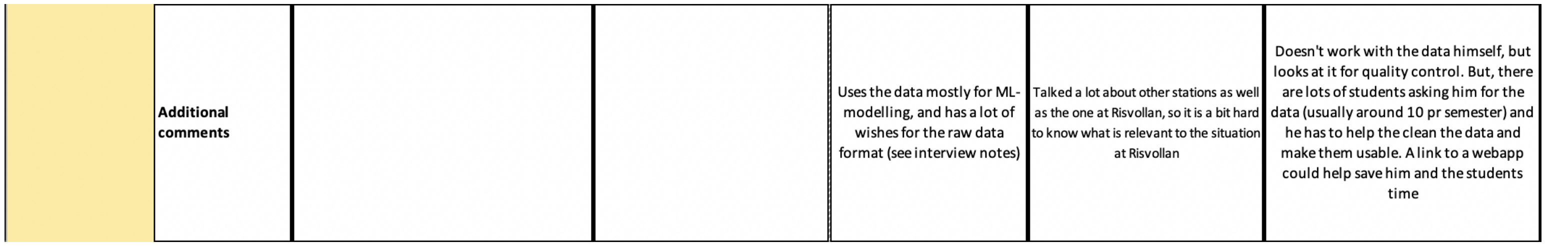


		R	MatLab	R	R and Python	Excel
Existing Solutions	**Current system and usage**	Scale for measurements	Plots the data to see if it reliable	imports the data files	Imports the data	Imports the .txt file
		Overlaying different time stamps		trims the data	Cleans the data	Selects relevant data
		Seeing the same time period from several years together	Scripts are run to make the raw data series usable and cohesive	feed the data into ML-models	Feeds the data into the Ml-models	Cleans the data
		Changing colors				Creates graphs of relevant data
Features and fuctionality	**Comparisons**	Storm and wastewater	Precipitation and runoff	His own predictions and the actual runoff	Not mentioned	Not relevant
		Flow in similar time periods		Time periods		
		Time periods (ex: week to week)	"+" Temperature and snow melting			
	Correlations	Rainfall and flow, with rainfall on top	Precipitation and wastewater	Weather data with runoff (what affects the runoff most?)	Percipration, temperature, runoff	Not relevant

Figure B.2: Interview Results Matrix: Existing Solutions and Features and Functionality

Motivation and expectations	Would like	Dashboard where you can see the different data	Warning if the data is not behaving as expected	Warning when the data is failty	Data status (when was data last imported?)	A notification when there are descrepancies in the data
		An ongoing realtime data serie		Graphs to use in his papers	See the pictures taken in the selected time frames	Ability to add a remark that a sensor has been changed or repaired
		Easy access to download	Downloadable data series	Downloadable raw files in selected format	Color coded sensor status (green for good, orange for suspected faultiness and red for not working)	See just the relevant data
		Change of resolution (time series)	Change the resolution	Change of resolution	Show the units of measurement on the axis	Select the time frames
			Choose events	Change the axis		
				Bilingual (Norwegian and English)	See the sensor information	
		Select the data you want to see	Interactivity	See multiple sensors at once	Select multiple sensor to see in the same graph/simultaniously	On behalf of the people he has helped: See correlations
				Upload and add your own data on top		
		(Export to other applications real time)	Accumultated precipitation	Average values in selected time frames	plot it as a cumulated, distributed function ("what is the discharge occurring the last year in one percent of the case")	Easier access to control the data (not having to go through Excel)
				Clear variable names		
			Simple and cohesive data format	change the colors		
	Important aspects	Easy to use	Understandable visualization for first and second grade students	Comfortable	Easy to use	Notification when descrepancies occurs
					Reliable	
		Clear, not too much information, not too crowded	Easy access	Simple views	Consistancy in data format	Easy to spot descrepancies in the data
	Main functionality	See what the data looks like	Plotting of climate data along with runoff data	Customizable downloads	Status check on incomming data	Quality control
		Quality control		Warning options		
		Error detection	Expandable solution for future further development	Compare and look for correlations		
		Changing the graph				
	Main area of improvement	Accesability	Usage of the data in lectures and student assignments	Downloadable	Status check on incomming data	Time
		Downloadable	Easy access to new incomming data series	Time		

Figure B.3: Interview Results Matrix: Motivation and Expectations

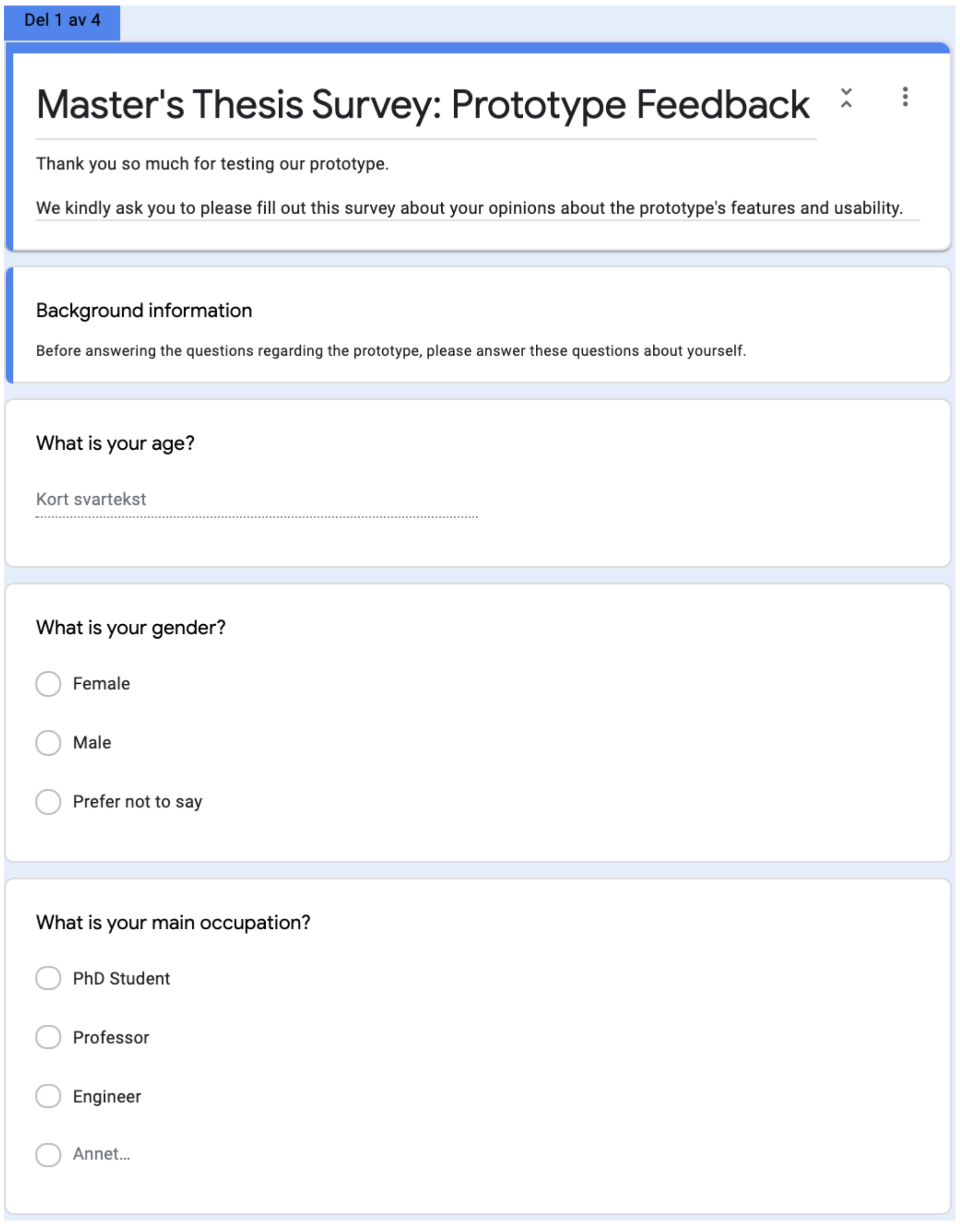


	Additional comments			Uses the data mostly for ML-modelling, and has a lot of wishes for the raw data format (see interview notes)	Talked a lot about other stations as well as the one at Risvollan, so it is a bit hard to know what is relevant to the situation at Risvollan	Doesn't work with the data himself, but looks at it for quality control. But, there are lots of students asking him for the data (usually around 10 pr semester) and he has to help the clean the data and make them usable. A link to a webapp could help save him and the students time

Figure B.4: Interview Results Matrix: Additional Comments

Appendix C

Questionnaire



Del 1 av 4

Master's Thesis Survey: Prototype Feedback

Thank you so much for testing our prototype.

We kindly ask you to please fill out this survey about your opinions about the prototype's features and usability.

Background information

Before answering the questions regarding the prototype, please answer these questions about yourself.

What is your age?

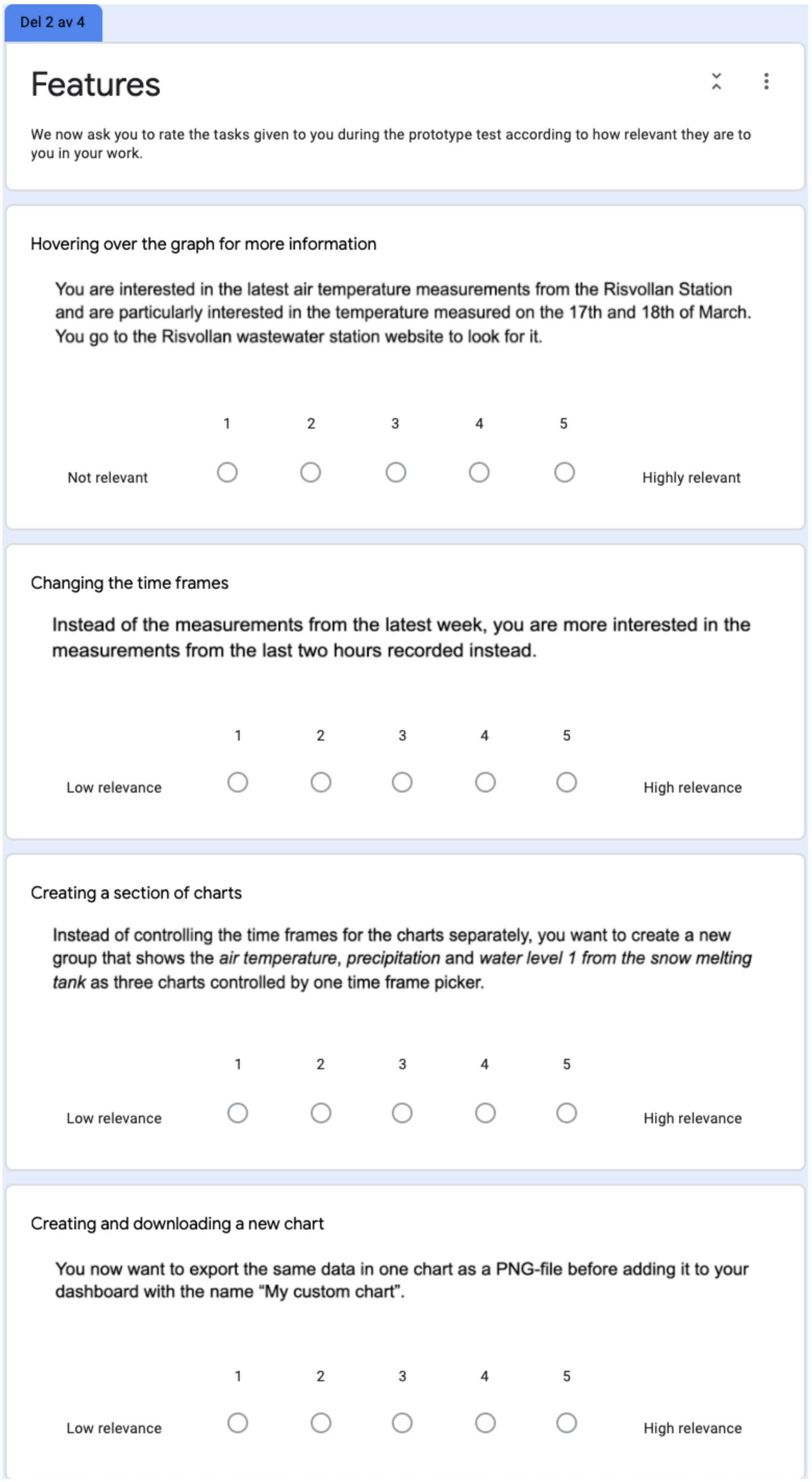
Kort svartekst

What is your gender?

- Female
- Male
- Prefer not to say

What is your main occupation?

- PhD Student
- Professor
- Engineer
- Annet...



Del 2 av 4

Features

We now ask you to rate the tasks given to you during the prototype test according to how relevant they are to you in your work.

Hovering over the graph for more information

You are interested in the latest air temperature measurements from the Risvollan Station and are particularly interested in the temperature measured on the 17th and 18th of March. You go to the Risvollan wastewater station website to look for it.

	1	2	3	4	5	
Not relevant	○	○	○	○	○	Highly relevant

Changing the time frames

Instead of the measurements from the latest week, you are more interested in the measurements from the last two hours recorded instead.

	1	2	3	4	5	
Low relevance	○	○	○	○	○	High relevance

Creating a section of charts

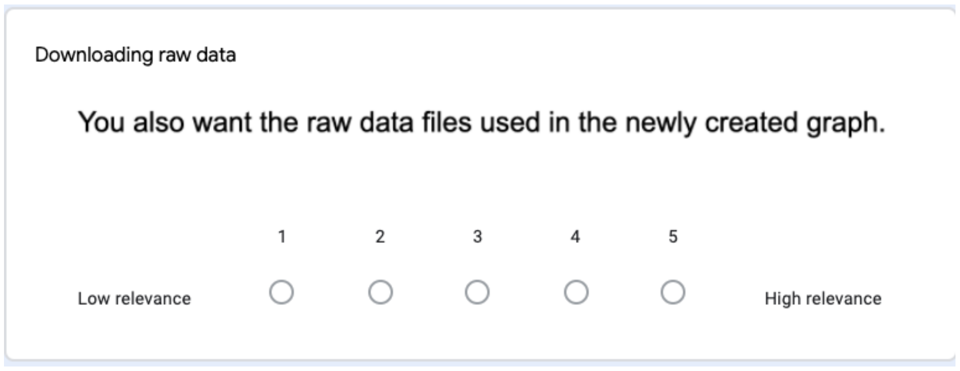
Instead of controlling the time frames for the charts separately, you want to create a new group that shows the *air temperature*, *precipitation* and *water level 1 from the snow melting tank* as three charts controlled by one time frame picker.

	1	2	3	4	5	
Low relevance	○	○	○	○	○	High relevance

Creating and downloading a new chart

You now want to export the same data in one chart as a PNG-file before adding it to your dashboard with the name “My custom chart”.

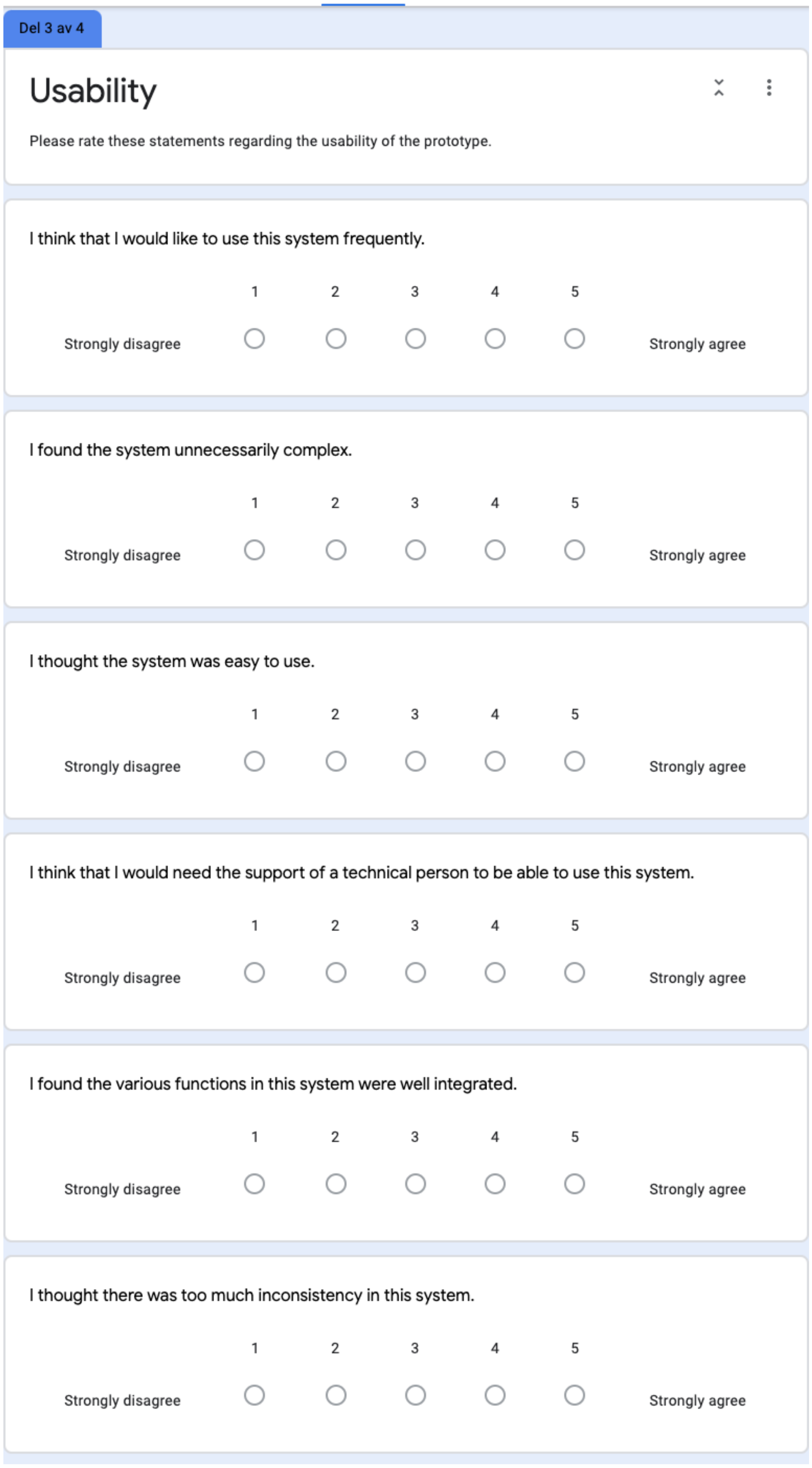
	1	2	3	4	5	
Low relevance	○	○	○	○	○	High relevance



Downloading raw data

You also want the raw data files used in the newly created graph.

	1	2	3	4	5	
Low relevance	○	○	○	○	○	High relevance



Del 3 av 4

Usability

Please rate these statements regarding the usability of the prototype.

I think that I would like to use this system frequently.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

I found the system unnecessarily complex.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

I thought the system was easy to use.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

I think that I would need the support of a technical person to be able to use this system.

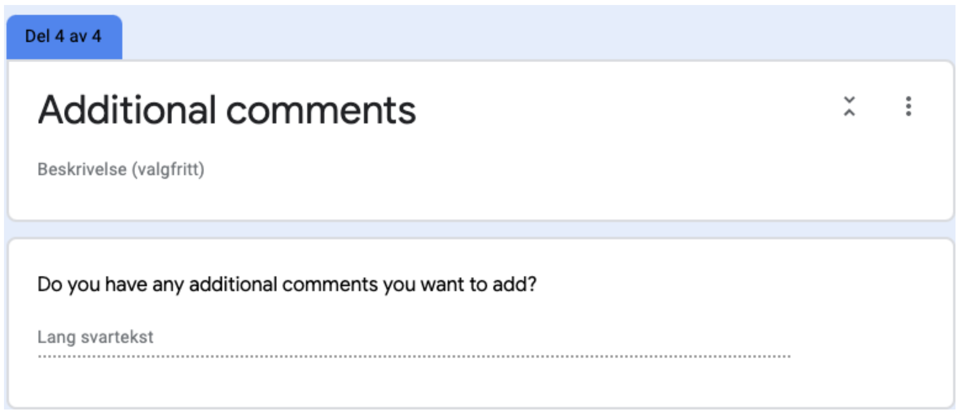
	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

I found the various functions in this system were well integrated.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree

I thought there was too much inconsistency in this system.

	1	2	3	4	5	
Strongly disagree	○	○	○	○	○	Strongly agree


Del 4 av 4

Additional comments

Beskrivelse (valgfritt)

Do you have any additional comments you want to add?

Lang svartekst

Selecting Libraries to Test

This part of the appendix explains the process of selecting potential libraries for testing.

Finding Potential Libraries

There are a lot of visualization tools out there. Lecturer at Berkely School of Information, Benoît, classifies visualization tools into three groups [2]:

1. Stand-alone visualization applications
2. Tools for creating interactive visualizations (usually web-based)
3. Tools that visualize statistical data, either as a built-in part of the program, or as a plug-in itself

Since the goal was to create a visualization application, the first group was not relevant other than for inspiration. Since it was clear from the beginning that the visualizations should be interactive, the third group was also not as relevant to this project. As for the second group, this is where you will find a lot of the visualization tools that could be relevant when developing the final solution. Benoît mentions D3.js as a powerful and approachable do-it-yourself visualization library [2].

Nine potential visualization libraries were found in our research. These are presented in Table C.1, with some additional information about each of them.

Library	**Technology**	**License**	**Relevancy**	
			Last updated	**GitHub stars**
Chart.js	JavaScript	MIT [46]	Sep 2020	52.1k
D3.js	JavaScript	BSD 3-Clause License [45]	Sep 2020	95.7k
Dash	Python	MIT	Sep 2020	13.9k
Google Charts	JavaScript	Apache 2.0 License [44]	Jul 2020	—
Nivo	React, D3	MIT	Jul 2019	8.3k
React-vis	React, D3	MIT	Apr 2019	7.7k
Recharts	React	MIT	Sep 2020	15.8k
Toast UI	JavaScript	MIT	Apr 2020	4.9k

Library	Technology	License	Relevancy	
			Last updated	**GitHub stars**
Victory	React	MIT	Sep 2020	8.7k

Table C.1: Library Candidates

C.0.1 Defining Library Criteria

The next step in the process was to define criteria for evaluating the libraries. The goal was to test a select few of the candidates and decide on the most suitable one. The criteria were discussed and defined with input from the supervisor and stakeholder.

Browser Compatibility

It was important to consider browser compatibility, since the application needed to be accessible to the users, in other words, work on as many of the widely used browsers as possible.

Ability to Add Multiple Y-Axes

Since the main goal of the application was to visualize sensor data, it was agreed that the solution would focus on presenting charts. As the chart types were not yet decided, the possibility of needing to display multiple y-axes in the charts, could not be ruled out. This criteria was noted as especially important since not having this feature would be limiting during the implementation process.

Documentation Quality

Using a library requires good *documentation quality* (DQ), so that the developer understands how to utilize its functionality. To evaluate the quality of the documentations, a way of calculating a score was created, which was based on the documentation's *availability*, *organization*, and *simplicity*. The possible scores were *Low* (0-1), *Medium* (2), and *High* (3).

Availability score + Organization score + Simplicity score = Total DQ score

The documentation quality score definitions can be seen in Table C.2, while an example of giving a DQ score to the React-vis library, can be seen in Figure C.3.

	Availability	Organization	Simplicity
1	Good	Good	Simple
0	Poor	Poor	Complicated

Table C.2: Documentation Quality Score Definitions

React-vis	Score
Availability	1
Organization	1
Simplicity	0

React-vis	**Score**
Total =	2

Table C.3: Example of Giving DQ Score to a Library

Supporting Material Availability

In addition to having good documentation quality, it was considered helpful if the library had a reasonable amount of online supporting material. This signaled that the library was established and favoured by other developers.

To rate the supporting material availability of the libraries, each one was categorized as either having a *Limited* or a *Wide* availability.

Customizability

Since the purpose of the solution was to visualize data in multiple ways, few limitations were favoured. Additionally, the library should support various types of data sets, be reasonably customizable when it came to fonts and colors, and ideally also customizable when it came to margins and padding, placements of texts etc.

The libraries' customizabilities were rated as either *Low* (Limited customization), *Medium* (Some customization available), or *High* (A lot of customization available).

License

Since the application was intended to be deployed for external use, the library would ideally need to be open source. In simple terms, open source means that there are no restrictions on how and where people can use the code from the library. This means that there also would be no risk of legal action when using these libraries in the project.

Relevancy

Relevancy of each library was considered. In this case, a relevant library is a framework that is in touch with the ever-evolving world of web development, which means that it is frequently updated and has a substantial amount of users. To define how relevant a library was, the date of the last library update was considered, as well as how often is was updated, and the amount of GitHub stars.

Overall Impression

After inspecting the libraries, we wanted to document the overall impression of each library. Influenced by the results from all of the previous criteria, the overall impression would work as a way of selecting which frameworks were going to be tested.

When defining the overall impression of the libraries, the libraries were categorized as either having a *Good* (At least 3/4 criteria passed), *Mediocre* (At least 2/4 criteria passed), or *Poor* (Less than 2/4 criteria passed) overall impression. If the library got an overall good impression, it would go on to being tested. The browser compatibility criteria was not included in the total, since so many of the results were unclear.

C.0.2 Evaluating Potential Libraries

Table C.4 shows how each of the nine potential libraries were evaluated based on the criteria.

Browser Support Definitions

- Legacy: IE8+, Chrome, Firefox, Edge, Safari
- Modern: IE9 (or other if specified), Chrome, Firefox, Edge, Safari
- Unclear: Not mentioned, but most likely modern

Library	Browser compatibility	Multiple y-axes	Documentation quality	Supporting material availability	Customizability	Overall impression
Chart.js	Modern (IE11)	Yes	High (1+1+1=3)	Wide	Medium	Good
D3.js	Modern	Yes	Medium (1+1+0=2)	Wide	High	Good
Dash	Unclear	Unclear	Low (1+0+0=1)	Wide	High	Mediocre
Google Charts	Legacy	Yes	Low (1+0+0=1)	Wide	Medium	Mediocre
Nivo	Unclear	Yes	High (1+1+1=3)	Limited	Medium	Good
React-vis	Unclear	Unclear	Medium (1+1+0=2)	Limited	Medium	Poor
Recharts	Unclear	Yes	High (1+1+1=3)	Limited	Low	Mediocre
Toast UI	Legacy	Unclear	Medium (1+1+0=2)	Limited	Medium	Poor
Victory	Unclear	Yes	High (1+1+1=3)	Limited	Medium	Good

Table C.4: Potential Libraries and Criteria

C.0.3 Libraries Selected for Testing

The libraries selected for testing were Chart.js, D3.js, Nivo, and Victory. These libraries were ranked, in order to choose which library was going to be tested first. The ranking was based on the criteria that were considered most important: *Multiple y-axes*, *customizability*, *license* and *relevancy*.

If the library had recently updated and had a substantial amount of stars, it would be *highly relevant*. A substantial amount would be at least ten thousand stars above the average number of stars between the selected libraries. Recently updated meant that it was updated at least within the last six months.

Top Rated Libraries

1. **D3.js -** Multiple y-axes, high customizability, BSD 3-Clause license, highly relevant
2. **Chart.js -** Multiple y-axes, medium customizability, MIT license, highly relevant
3. **Victory -** Multiple y-axes, medium customizability, MIT license, medium relevant
4. **Nivo -** Multiple y-axes, medium customizability, MIT license, medium relevant

Appendix D

Technologies

This table shows the technologies that were used for designing, prototyping, and developing Nykken. It also describes the type of technology, as well as the main use of each technology.

Technology	**Type**	**Use**
Adobe Illustrator [65]	Vector based design program	Designing Nykken logo and other icons
D3.js [9]	JavaScript library	Visualizing sensor data in charts
Figma [16]	Web-based application	Creating and testing prototypes
GitLab [26]	DevOps lifecycle tool	Source code management and issue tracking
Netlify [27]	Hosting service	Hosting Nykken application
React [52]	JavaScript library	Developing web application
React-component-export-image [58]	JavaScript library	Downloading PNG image of chart
React-csv [53]	JavaScript library	Downloading sensor data as a CSV file
React-select [74]	JavaScript library	Creating customizable select drop-downs

Table D.1: Technologies Used to Design and Develop Nykken

Appendix E

Status of User Stories

This appendix presents the status of the user stories after the development process.

ID	Description	Status	Comments
1	As a user I want to be able to see sensor data in a chart.	C	Users can see sensor data in charts and sensor groups.
2	As a user I want to be able to choose which sensor charts I want to see on my dashboard.	C	This is made possible through the Add/remove sensors functionality.
3	As a user I want to be able to select custom time frames for each chart.	C	Possible through the Edit timeframe functionality.
4	As a user I want to be able to zoom in and out of the chart.	NC	Zooming in and out of chart was not implemented in the final solution.
5	As a user I want to be able to hover to see more detailed information about data points.	PC	It is possible to hover for more information, but the functionality needs improvement.
6	As a user I want to be able to download the data in a CSV-file.	C	Made possible through clicking the Download as CSV-button.
7	As a user I want to be able to download the chart as a PNG-file.	C	Made possible through clicking the Download as PNG-button.

ID	Description	Status	Comments
8	As a user I want to be able to see multiple sensor data in one custom chart.	NC	The custom chart functionality was not implemented in the final solution.
9	As a user I want to be able to give my custom chart a name.	NC	Since custom charts were not included in the final solution, it was also not possible to give a custom chart a name.
10	As a user I want to be able to customize the colors of the graphs inside of my custom chart.	NC	Similar reasons to why ID 9 was not completed, it was not possible to customize colors of a chart that could not be created.
11	As a user I want to be able to receive updates on selected charts.	NC	This functionality was not implemented in the final solution.
12	As a user I want to be able to group charts together and set a time series for the whole group.	C	It is possible to group sensor charts together by clicking the Add sensor group-button.
13	As a user I want to be able to see when the API was updated last.	PC	The header includes a text which tells the user when the API was last updated, but it is only accurate if the hardcoded date matches the actual date of the last data update in the API.
14	As a user I want to be able to access the sensor information.	C	When visualizing sensor data i charts, the sensor information is visible as the title of the sensor and also the unit on the y-axis.
15	As a user I want to be able to see the status of the data (faulty or good).	PC	The status of the data can be interpreted by looking at the charts. E.g. a negative value for rainfall would imply that the data was faulty. Nykken does however not specifically tell the user if there is something faulty about the data.
16	As a user I want to be notified when discrepancies occur.	NC	This functionality was not implemented in the final solution.

Table E.1: Status of User Stories

Sunniva M. Runde & Kaja L. Solberg

Nykken: Visualizing Hydrological Data

NTNU
Norwegian University of
Science and Technology