Lowering household CO_2 emissions from energy consumption through an advice driven web application

Frederik Bode Thorbensen, Hans Erik Heje, Jakob Saadbye,

Markus Hye-Knudsen, Ming Hui Sun og Sture Svensson

June 18, 2021

AALBORG UNIVERSITY

DENMARK

AALBORG UNIVERSITY

STUDENT REPORT

Title:
Lowering household CO_2 emissions from energy consumption through an advice driven web application

Theme:
Smart homes and smart cities

Project Period:
2. SEMESTER

Project Group:
sw-k2204

Participant(s):
Frederik Bode Thorbensen
Hans Erik Heje
Markus Emil Hye-Knudsen
Ming Hui Sun
Jakob Saadbye
Sture Svensson

Supervisor:
Sokol Kosta

Copies: 1

Page Numbers: 114

Date of Completion:
June 18, 2021

Abstract:

The amount of personal and in-home electronic devices has been steadily increasing in recent times, and an average Dane uses an estimated 1.600 kWh pr. year. This development has happened alongside an increasingly global crisis and with more than 194 countries committing to the Paris Agreement, trying to limit the carbon emissions level increase to 2% pre-industrial levels. To lower carbon emissions, society needs to utilize sustainable energy sources more effectively to minimize the use of non-sustainable sources. After surveying Danish individuals, we found that 77.7% would be willing to postpone energy-consuming activities to reduce their carbon footprint. An analysis of the survey results led to the following problem statement: "*How can we build a web-based application to inform Danish smart-home owners when to use their appliances based on the availability of sustainable energy*". After researching the current state of the art and similar applications, we extracted requirements to develop a web application that informs and advises the users about current energy trends. Using the latest web technologies we were able to build a web-application where users get informed about their energy utilization. Moreover, the application uses a data-driven approach to advise the users how to make more sustainable choices and help them in the planning of their daily energy-consuming activities through CO2 emissions forecasts. Automatic end-to-end tests confirmed the reliability of the application while the user tests shows the application could be improved with smart connectivity. Even without this feature, it was deemed highly usable and informative by the test subjects, and thus it can be concluded that the project was a success.

Preface

The code base for our projects web-application can be found at https://github.com/bodeby/Project-skarp. The web application can be accessed by visiting http://sustaininator.eu

We would like to thank Sokol Kosta for his inspirational supervising and the friends and family that participated in the preliminary survey and final user testing.

Contents

1 Introduction

Most people living in modern society, including citizens in Denmark, require electricity to some degree. Today's electricity is an essential resource for many technologies we have become dependant upon in our daily lives. The technologies would not even function, whether refrigerators that improve our food's longevity, laundry machines that keep our clothes clean, or light bulbs that illuminate our homes. The list of home appliances requiring power has steadily increased over the years [1][2], increasing power production to meet the demand. In 2020 the average Dane used roughly 1.600 kWh a year, and the average family uses an estimated 4.500 kWh [3]. According to a report from the Danish Energy Agency, every kWh energy produced creates 226 g CO_2 [4], which translates to about 1017 kg CO_2 emitted by an average danish household per year.

It is no secret that some appliances use vastly more power than others. The diagram below illustrates the energy usage of a typical family living in a house.

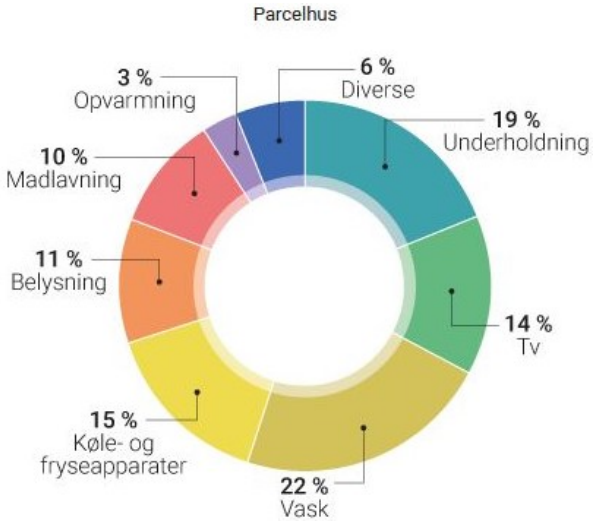


Figure 1: Electrical consumption in an average family house in Denmark [5]

.

Home appliances such as refrigerators, laundry machines, and freezers make up the most significant part of residential energy consumption. In contrast, entertainment such as TVs, computers, and gaming consoles make up almost a third. Some home appliances have relatively constant energy usage and are therefore not suitable for time-controlled operation. Refrigerators and freezers need to be on, and as such, their energy usage cannot be made smart. However, the inefficiency of older device (see appendix A for our definition of a device) models can be replaced with more modern and energy-efficient versions.

To keep track of the energy consumption, one may look at their energy bill to gain insights into their energy spending. They might find that they consume more or less energy than the average household and wonder what steps they can take to lower their energy consumption. The installed power meter will only give the total energy consumption and cannot determine the consumption of individual devices. Knowing the power consumption of every device in a home would make it possible to spot energy guzzlers and reduce their impact based on the knowledge obtained by monitoring them.

1.1 The idea

Our main idea is to build a web application that can inform users of their energy consumption and allows the user to align their appliances use-times to times of higher sustainable (see Appendix A for our definition of sustainable) energy production and lower carbon-emission rates. The information will analyze the current production and CO_2 emissions data provided by energinet.dk. The information shown will be easier to understand the advice cards with information such as: "*The Energy Production is at a high point, now would be a good time to use more energy*" or feedback like "*You're doing great, keep up the good work*."
The user has to register a profile that will be representing their household. Users can place sensors on appliances by adding devices to their profile, which will give users the ability to monitor energy consumption and support the recommendation system. The end goal of a solution like this is for users to contribute to the shifting and flattening of the energy demand curve, reducing the need to import other types of energy that ultimately increase CO_2 emissions.

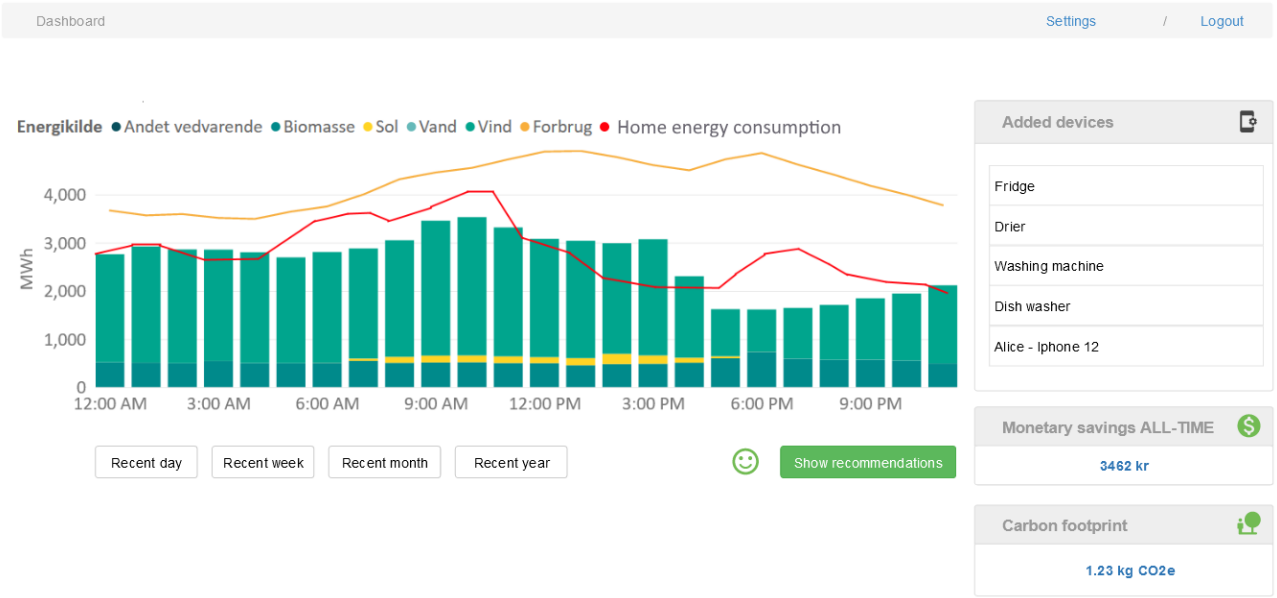


Figure 2: Early envisioned user interface

2 Motivation

Like many other countries, Denmark has agreed upon the Paris Agreement, which aims to keep the global temperature increase below 2°C from pre-industrial levels and "*to limit the increase to 1.5°C, since this would significantly reduce risks and the impacts of climate change*" [6]. Even though 194 states and the EU have committed to this agreement, the world's global CO_2 emissions are steadily rising [7]. It is a commonly known fact that the world needs to move in the opposite direction. We believe that all steps taken in this regard are a positive development towards making the planet sustainable.

For the planet to be more sustainable and to halt the increase in temperature, factories, vehicles, and households must alter their energy consumption in some way in order to decrease their CO_2 emissions. The danish households account for 10% of the total CO_2 emissions in Denmark [8], for households, the emission can be attributed to two main factors: electricity usage and heating. Even minor alterations in electricity usage of Danish households will make a difference in preventing the temperature increase.

Meeting the energy consumption demand of Danish citizens is done by energy production, which in Denmark comes more and more from renewable energy sources like wind and solar. These renewable energy sources help reduce our CO_2 emissions and have seen a steady climb during the past 30 years. The renewable energy sources account for around 40% of the total energy produced [9] and are predicted to rise towards 2030 with Denmark's goal of reducing CO_2 emissions by 70%. Still, the wind and solar energy produced, on windy and bright days, make up the energy demand of the Danes, where the excess energy is exported to neighboring countries. On these high energy production days, it is crucial that countries keep using renewable energy and that appliances can react to a spike in sustainable energy production to flatten the energy demand curve. As Signe Horn Rosted (Vice President, Business and Markets - *Energinet [10]*) puts it:

> It is crucial for an effective green transition, that we use green energy so that electric cars can charge when the wind is blowing and electrical heat pumps in both district heating and private homes react to when there is an abundance of cheap and green energy available in the energy system [11].

To make appliances react to the peak production of sustainable energy, they need to be able to communicate with the outside world. This need for appliances to react to incoming information leads to the idea of smart appliances, more commonly known as smart devices. To better understand what these smart devices are, we need to look at what a smart home is.

2.1 Smart Homes: A brief explanation

To understand what a smart home is, we examine the definition of a smart home. It is described as *a home equipped with lighting, heating, and electronic devices that can be controlled remotely by smartphone or computer* [12]. A smart home comprises different 'smart devices' from light bulbs that can be remote-controlled, fridges with cameras to make grocery lists, or doors that can be locked or unlocked by a smartphone. Smart devices have in common that they can be monitored, controlled, or accessed remotely. These smart devices usually connect via a central hub that communicates and sends commands to the individual devices to turn them on or off. The central hubs are often connected to the house's Wi-Fi connection, allowing it to be controlled from the user's smartphone or tablet. In a true fashioned smart home, all the devices that make up the smart home are scheduled and controlled to work in harmony with the house resident.

Controlling smart appliances is done through a central smart hub that connects the different protocols found in smart home products within a home. It is usually accessible through a centralized app that controls connected devices. Smart lights, pet bowls, smart speakers, and motion sensors are accessible through the hub. Instead of walking around flipping light switches and toggling thermostat settings, a smart hub "digitalizes" the manual steps involved in operating these devices. A resident can schedule, command, and control everything from one central interface using a smart hub.
For example, the hub can do cooking reminders at a fixed time if the person is busy with work at home. If a person has trouble waking up in the morning, then a smart hub can help set the alarm by speaking to voice-controlled devices, like Amazon Echo or Google Home [13]. Smart hubs are connected through an internet connection, making them accessible anywhere.

2.2 Energy efficient smart homes

Type of smart home
1 Smart homes for security
2 Smart homes for eldercare
3 Smart homes for healthcare
4 Smart homes for childcare
5 Smart homes for energy efficiency
6 Smart homes for entertainment

Table 1: Different types of smart homes [14].

In general, smart homes can be split into six different types based on the applications used in smart homes. We will mention two of them briefly, and then we will narrow our focus down to smart homes for energy efficiency. One aspect of smart homes is surveillance and home security. This smart home consists of devices to protect against burglary and vandalism. Some of the devices in a smart home for security include remote-controlled door locks, motion sensors, and intelligent cameras that monitor activity. The security system is then based on the data collected by the sensor and camera to give users feedback and alerts on possible intruders. Homes with alarm systems that make calls to an alarm center are also considered smart homes for security.
Another aspect where smart homes have found useful is in the eldercare sector with the latest technology of fall detection used in nursing homes [15]. The technology consists of one or more devices that detect when an elderly may have fallen and alerted the nurses caring for them. The technology has been implemented in Danish nursing homes and assists nurses in responding to possible injuries.

The more important application for our project is smart homes for energy efficiency. There is a growing number of solutions on the market to automatically monitor energy usage and propose a way to lower it. Devices found in a smart home for energy efficiency often include smart thermostats, smart plugs, and smart light bulbs, a few of these solutions will be expanded on in Section 5 regarding the state of the art.

2.3 Buyers of smart home technology

There can be many reasons as to why one may invest in a smart home. Building up a smart home that is intuitive takes, as all other practices, time. It usually begins with some annoyance, be it forgetting to turn off the lights at night or maybe you have placed your switch in a place where you can’t easily get to a TV or something

else. It can also be that you are at a friend's house, stuffed with smart home technology, and it's cool to see an automated home. Both are good reasons to start getting into the field of smart homes. The smart devices incorporated in your house can be made to control everything from lights to heaters or garage doors with a tap on your smartphone, and the technology is there to automate just about everything you would like, so why isn't everyone starting to rebuild their house?

Smart home technology can help with automating a lot of stuff, and the problem is not that there are not enough smart devices out there. The problem is in adapting to an automated lifestyle with technology around you. Adapting to a different way of living takes time, and especially for the elderly not born with technology at hand, which is also shown in Figure 3. The tendency to have smart home devices drastically drops in the age group 55-64. The younger half of the population generally have an easier time adapting to smart home devices and are therefore more willing and committed to investing.

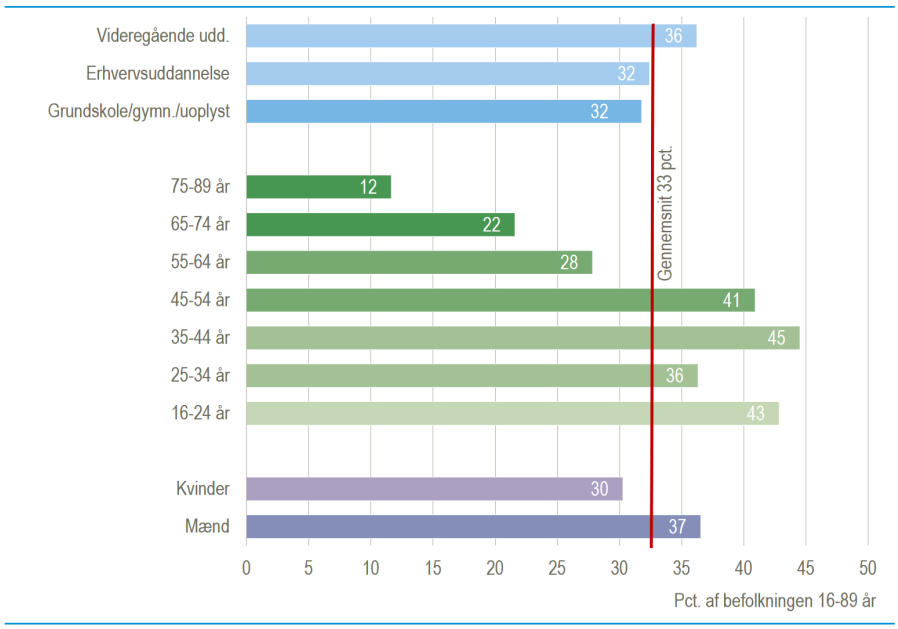


Figure 3: Who ses 'smart home' - devices (IoT) in Denmark [16].

The smart home market by revenue is forecasted to double between 2020-2024(S. 9, [17]), suggesting that more and more homes are becoming smart. As more people will invest in smart devices, more and more competitors will show up with their solutions for a smart home product. Not only have small companies begun to launch new smart home devices for a fair price, but also the bigger companies like Nest with their thermostats and IKEA with light bulbs [18]. In 2017, a smart thermometer by Nest had gone from $249 originally down to $169. It is still just as good as the previous one according to the verge - Micah Singleton, smart locks have gone down to $149 instead of $229, lastly, IKEA's light bulbs have an affordable price that starts at $12 [18]. It is more likely that people are interested in finding a product that they can control and visualize the energy consumption of, which is better value because it is affordable and efficient.

2.4 Denmark's Energy Production: Past, Present, and Future

Denmark has many other countries that have agreed to the Paris agreement and tried to cut back on energy production from sources with high CO_2 density. Since the 1990s, there has been a rapid development in the energy production sector. As we can see from the table in Figure 4 regarding energy production by source,

the total amount of energy produced peaked around 2005, with a total of 1.311.683 TJ (terajoule), where about 61% of the total energy produced came from raw fossil fuels. Since then, Denmark produces less total energy, primarily because the quantities of fossil fuel that have been extracted is only about 27% of the amount extracted in 2005, which means only about 40,5% of the energy comes from fossil fuel. So, where does the remaining energy come from? Since the 1990s, Denmark's sustainable energy production has seen a massive surge at about 288%, which translates to about 34,4% of Denmark's total energy production in 2019, versus the 10,7% in 1990 [4].

									Ændring
Direkte energiindhold [TJ]	**1990**	**2000**	**2005**	**2010**	**2015**	**2017**	**2018**	**2019**	**'90-'19**
Produktion i alt	**424 361**	**1 164 525**	**1 311 683**	**978 612**	**679 252**	**655 390**	**580 817**	**523 118**	**23,3%**
Råolie	255 959	764 526	796 224	522 733	330 662	289 690	243 629	215 741	-15,7%
Naturgas	115 967	310 307	392 868	307 425	173 510	182 142	155 071	115 740	-0,2%
Vedvarende energi	45 461	76 016	105 585	131 306	159 313	167 606	166 998	176 376	288%
Affald, ikke-bionedbrydeligt	6 975	13 676	17 006	17 148	15 767	15 951	15 119	15 260	119%

Figure 4: Primary Energy Production sources - Denmark 1990-2019 [4].

We can see this development reflected in Figure 5 below, which depicts the size of each of the four main groups in energy production. That means that the sustainable energy production sector is growing, and the transition to sustainable energy is gaining traction. But there is still a long way to become a nation that relies solely on sustainable energy sources.

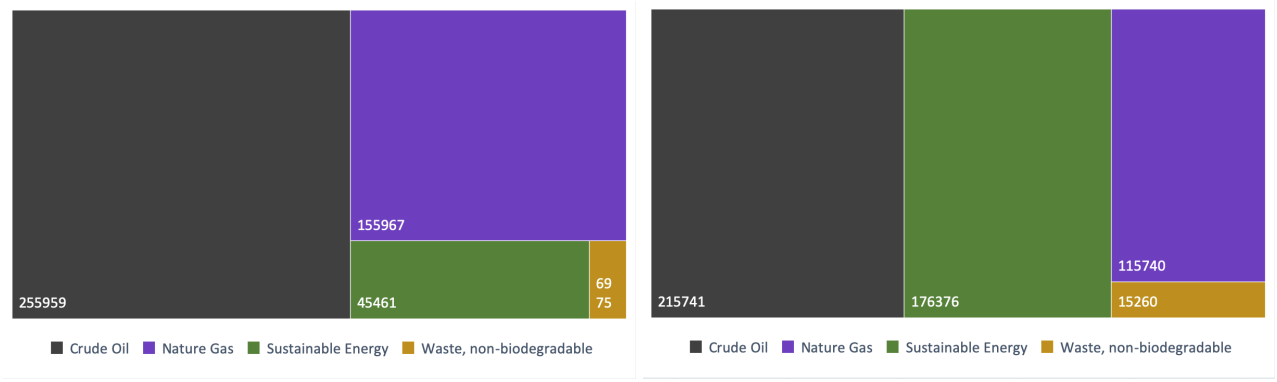


Figure 5: Distribution of energy production sources measured in terajoule (left: 1990, right: 2019) [4].

A projection report from the Danish Energy Agency provides the projection of the development in Denmark's energy production in 2018 towards 2030 [19]. The use of coal is expected to phase out of regular production and is only kept as a reserve option in periods of high demand, while oil and gas will be roughly halved. During this time, the energy production from wind and solar will more than double. Whether Denmark achieves these goals remains to be seen, but at the very least, an incentive to shift residential energy usage from brown to green exists until 2030.

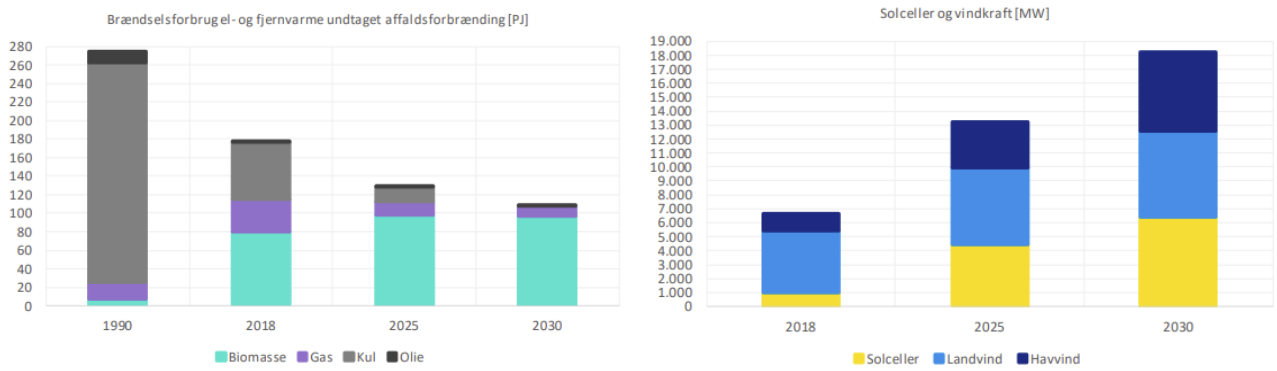


Figure 6: Left: Fuel use in energy production - Right: Renewable energy production

Hypothesis:

- The general public does not know the current renewable energy production status and fluctuations of energy prices and therefore can not react to a momentary surplus of renewable energy.
- Current home energy monitoring solutions focus more on energy savings as a means of monetary savings, rather than a focus towards lowering the CO_2 footprint of homeowners.

2.5 Survey

When considering a solution to reduce energy consumption and improve the amount of sustainable energy used for electricity in homes, we decided to do an anonymous survey to gain knowledge of how much people thought about using electricity and their electricity usage. When developing software for a consumer, it is vital to have an understanding of who the consumer is and how the software can impact them. For the survey to grant us the needed insight, we set up some criteria for what we needed to know. These criteria can be divided into three minor groups: Demography, environmental awareness, and energy consumption habits.

- Demography:
  During the survey, participants are asked their age group, current housing, and the size of the city they live in. These questions provide us with a better understanding of who our target audience is. We are using these questions to discern the span of age and regional differences in the end consumer of the product. This will allow us to provide a solution that is tailor-made to our specific target audience.
- Environmental awareness:
  This section of the survey serves to provide a deeper understanding of the mindset of our consumers concerning sustainability. We are hoping to learn whether the target audience is interested in bettering the environment or if their economy is a larger motivator.
- Energy consumption habits:
  Diving into the habits of our target audience when it comes to energy consumption, we first and foremost expect to determine what their current habits are and secondly whether or not they would be willing to alter those habits.

2.5.1 Survey results

The survey was made publicly available through several Facebook survey groups. It ran for one week. The following analysis is based on more than 50 responses we received during this week. Analyzing the responses

to the survey, we find that the vast majority of respondents, 89.2%, are between ages 18 and 39. Furthermore, 58.5% of them live in larger cities, and 68.5% live in apartments. This result is a clear indicator that people are interested in a survey regarding sustainable electricity consumption in homes have specific demography. The following diagrams are a representation of the data we collected during our survey.

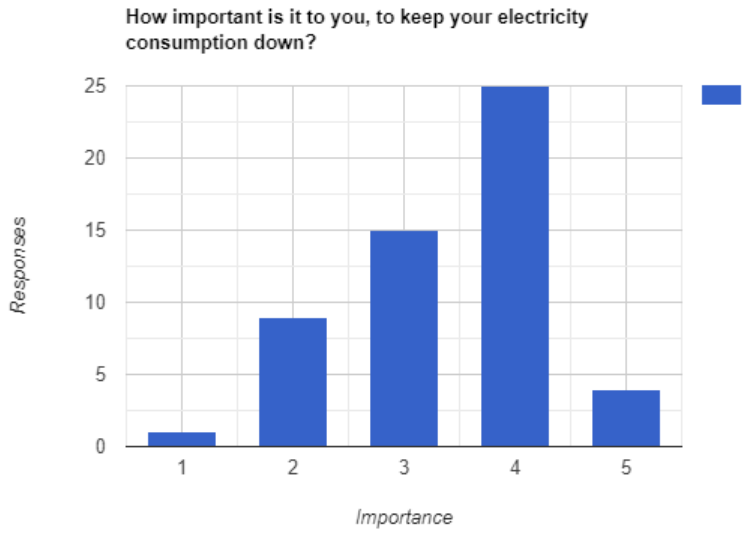


Figure 7: Electricity consumption importance

For the question regarding the importance of keeping down the electricity consumption, as seen in Figure 7, more than 81% of the respondents express that it is somewhat important or very important for them. This shows that the vast majority of the respondents are attentive to the amount of electricity they use at home.
When creating the survey, we sought to investigate exactly which measurements the respondents took to minimize their electricity bill's size or their impact on the environment from energy consumption. The next diagram, Figure 8, correlate to this aspiration.

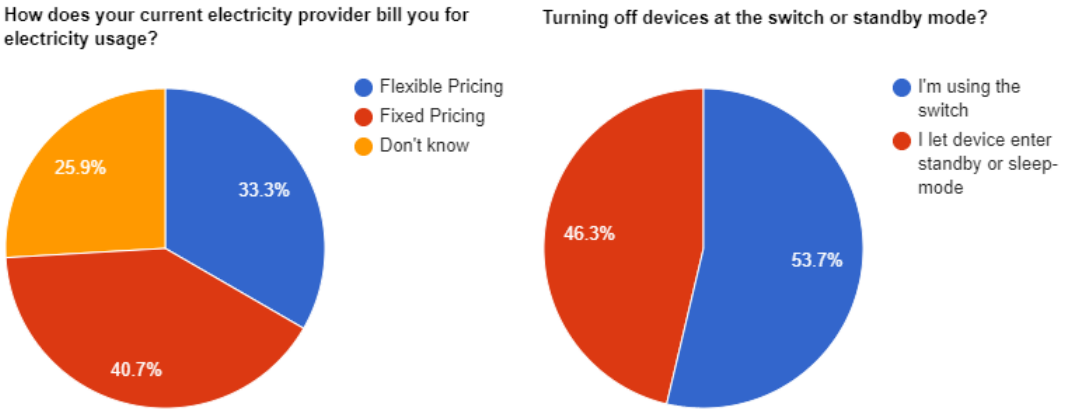


Figure 8: Left: Current billing type. Right: Flip the switch or standby

When looking at the 33.3% of respondents who have the bill with flexible pricing from their current electricity provider, it is necessary to keep in mind that 53.7% of the respondents also felt it was important or very important for them to keep their electricity consumption down. The one major component consumers have when cutting down their electricity bill, aside from not using electricity at all, is flexible pricing.

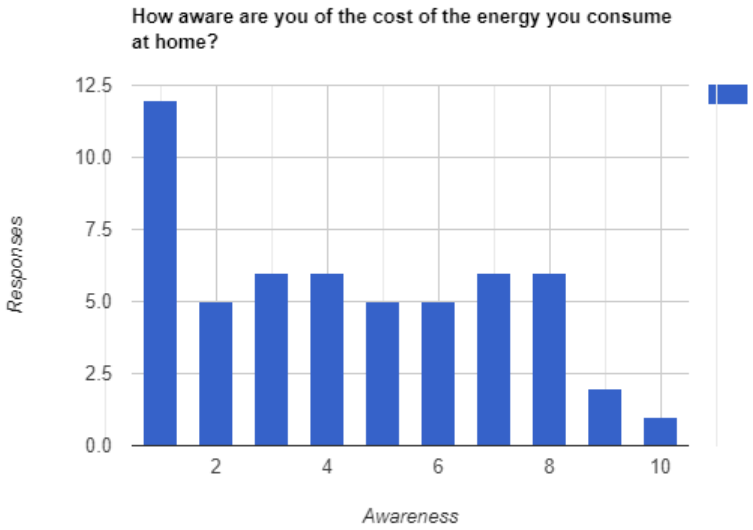


Figure 9: Energy price awareness

To expand our insight into the energy consumption habits of our respondents, we asked how aware they are of the price of the electricity they consume. For this question, as seen in Figure 9, we learned that a large segment of the respondents (22%) are completely unaware of the price. This, combined with the number of respondents who are billed flexibly from Figure 8, indicates that monetary enticement is not necessarily the most substantial influence when it comes to impacting the consumer's energy consumption habit.

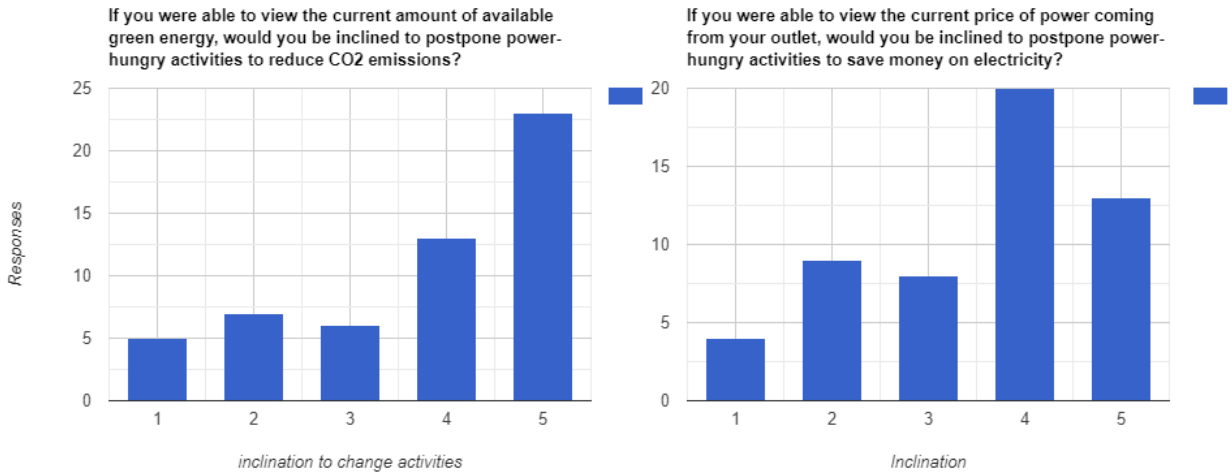


Figure 10: Left: Inclination to reduce CO_2 emissions. Right: Inclination to save money on electricity

The idea behind the questions in Figure 10 was that most people would be willing to move some of their power-hungry activities to a later time to save either money or reduce their CO_2 emissions if information regarding the current price and cleanliness of the electricity was readily available. All of the respondents were asked both questions, and we observe that the vast majority of them would be inclined to alter their consumption habits. The environmental benefits from moving power-hungry activities weigh the most, with more than 66% of our respondents expressing a high or very high inclination to move their activities.

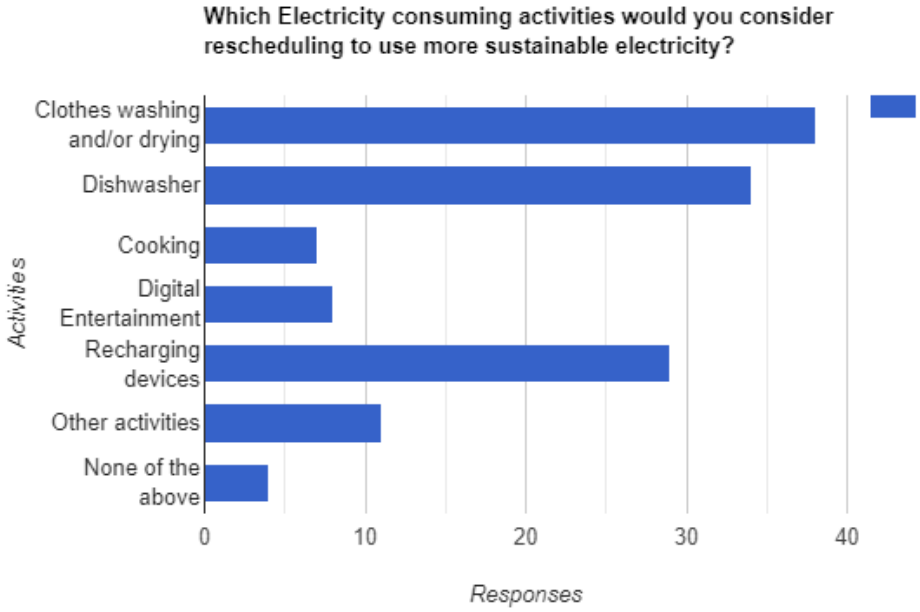


Figure 11: Activities respondents are willing to move

We assumed that people would be willing to move some of their power-hungry activities if it allowed them to save either money or help the environment. This lead to the question in Figure 11 regarding which activities our respondents would consider rescheduling. Most notably, this shows that 70% of the respondents would consider postponing the washing and/or drying of clothes, while more than 62% would postpone their dishwasher and 53% would postpone the recharging of devices. While not many people show interest in postponing digital entertainment and cooking, some respondents reflect the possibilities of affecting a multitude of different people.

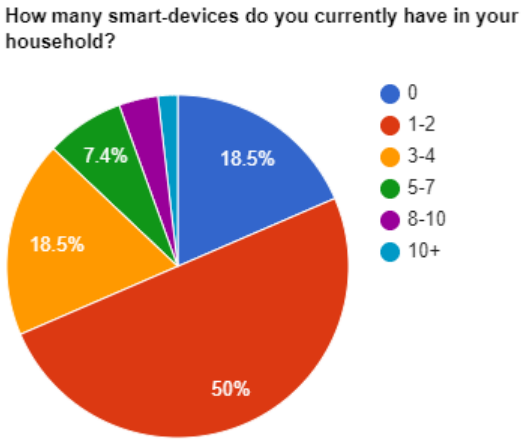


Figure 12: Amount of smart-devices in respondents households

Provided the overall subject for this project being Smart Homes and Smart Cities, we required an understanding of how smart the average consumers' home is. This leads to the question in Figure 12 which presents the case that 68.5% of the respondents' households contain somewhere between 0 and 2 smart devices. Furthermore, only 13% have more than 5 smart devices, which in turn indicates that a solution only accommodating smart devices would have very few possible end-consumers.

2.5.2 Survey conclusion

The results from the survey have granted important insight regarding our target audience, their environmental awareness, and their energy consumption habits. Based on the responses to the survey, our average end-user is between the ages of 18 and 39. They live in a larger city, in an apartment. Furthermore, only 33% of the respondents are billed flexibly by their electricity provider. At the same time, close to 75% express an interest in postponing their power-hungry activities to save money on their electricity bill. From the respondents who wish to save money on electricity, only the respondents with flexible pricing would benefit from a solution that provided the user with the optimal time to consume electricity.

The respondents who express an interest in postponing power-hungry activities to reduce their carbon footprint are more passionate. More than 42% of the respondents express a **high** inclination to modify their energy consumption habits, and in total 77.7% have at least some inclination. The respondents' disposition to alter their energy consumption habits for monetary gain or the reduction of CO_2 emissions is a clear indicator that the target audience comprises two groups. The focus of our solution must lie where we can impact the most people. Consequently, our main concern is with the group of users who wish to save CO_2. We choose to focus on this segment, as monetary savings is somewhat correlated with how CO_2 neutral the produced energy is, this correlation is explained in more detail in section 2.7.3.

A solution targeting owners of smart devices alone would require the target audience to have a substantial amount of smart devices in their homes in order to have a significant impact on either their economy or CO_2 emissions. The survey shows that more than 68% of the respondents' households have between 0 and 2 smart devices. These households would experience a minimal impact from the use of such a solution. Hence, to accommodate as many users as possible, our solution must support non-smart devices.

2.6 Problem statement

How can we build a web-based application to inform Danish smart-home owners when to use their appliances based on the availability of sustainable energy.

- *How do we collect the data necessary to help users monitor electricity prices and carbon footprint?*
- *How do we present the data in an organized way?*
- *How do we collect data on non-smart devices to cooperate with our web app?*

2.7 Analysis and Environmental Motivation

In accordance with the United Nations resolution 13: "Climate Action" from the Sustainable Development Goals, taking action against or minimizing climate change is of critical importance to all areas of society. This is best encapsulated in their quote concerning the subject:

> "Climate change presents the single biggest threat to development, and its widespread, unprecedented impacts disproportionately burden the poorest and most vulnerable. Urgent action to combat climate change and minimize its disruptions is integral to successfully implementing the Sustainable Development Goals." [20].

This action does not have to come solely from new technology but can also come from optimizing old technology or changing our culture and consumer patterns. This is definitely what our solution hopes to accomplish.

Instead of creating more waste to generate more energy instead, the current amount of sustainable energy should be utilized to a more significant effect, such that it does not go to waste.

The question quickly becomes: How much can a solution for electricity consumption in households theoretically save the environment?

2.7.1 Environmental Benefit

In Denmark, the total corrected (adjusted for import and export) amount of emissions in 2018 was equivalent to 51.6 million tons of CO_2. Out of these 51.6 million tons, 60.8% came from energy usage ([4], p. 42). Homes make up 30.1% of the total Danish energy usage. Which is:

$$51.6 \cdot 10^6 tons\, CO_2 \cdot 60.8\% \cdot 30.1\% = 9.44 \cdot 10^6 tons\, CO_2$$

$$or$$

$$\frac{9.44 \cdot 10^6 tons\, CO_2}{51.6 \cdot 10^6 tons\, CO_2} = 18.3\%$$

$$of\ the\ total\ Danish\ emissions$$

As seen on Figure 13 below, out of these 18.3%, 25.7% is already sustainable energy, and 19.6% is currently non-sustainable electricity usage.

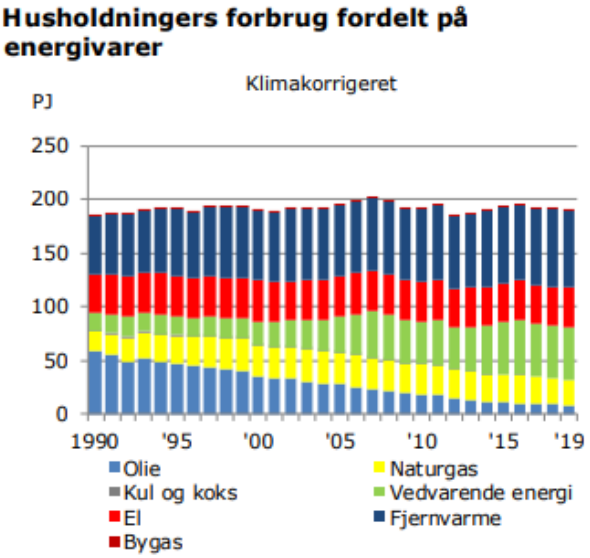


Figure 13: Energy Usage in the home ([4], p. 35).
Sustainable energy, Electricity consumption.

This means that a solution which eliminates non-sustainable electricity usage in Denmark, given the entire population uses the solution perfectly efficiently, can save the environment for:

$$18.3\% \cdot 19.6\% = 3.6\%\ of\ the\ total\ Danish\ emission$$

$$or$$

$$\frac{51.6 \cdot 10^6 tons\, CO_2}{100} \cdot 3.6 = 1.86 \cdot 10^6 tons\, CO_2$$

$$1.86\ million\ tons\ of\ CO_2$$

It i s not realistic that everyone would use this product, nor is it confirmed that the solution removes all non-sustainable electricity usage. If one were lucky, maybe 50% of the population would use it, and the answer may remove half of all non-sustainable electricity usage. If one were to take a wild guess, it might remove 25% of the non-sustainable electricity usage in Danish homes. This may not seem like much, but any step forward is a step in the right direction.

2.7.2 Economic Incentive

Even if the product does not directly have much economic incentive overlapping with the sustainable plan as discussed in Section 2.7.1, hindering climate change alone can be considered a financial incentive. This consideration stems from the fact that global warming directly contributes to the total cases of dangerous and costly diseases, which in turn increases societies' healthcare expenses [21]. If these things did not happen, then may it be so that our community not only saves money but potentially converts sick people into taxpayers who contribute to society and its economic growth.
Now the question becomes, does the sustainable agenda overlap with the direct economic incentive, and when could one optimize both factors the most.

2.7.3 Correlation Between Price and Sustainability

For the purpose of testing whether a sweet spot for when energy is both cheap and renewable exists, a lot of data is needed. Two electricity service providers had a live feed on their electricity prices before taxation [22] [23]. This hourly rate is only applicable to those who choose flexible prices. Their prices were written down, the average between them was taken and plotted in an hourly graph, which also represents the average price between 17/1/2021-17/2/2021:

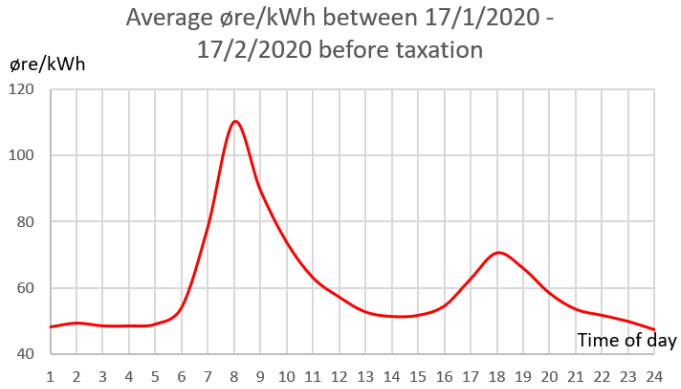


Figure 14: Average Price of Electricity Throughout the Day

From this graph, it can be seen that electricity is the cheapest between 20-6 and again between 11-17. An excellent reverse correlation with the times people are usually at home from work, and either has to cook food, turn on all the lights, or do other energy-intensive tasks.

Next the total amount of CO_2 emissions per kWh in the same period was plotted (data from [24]):

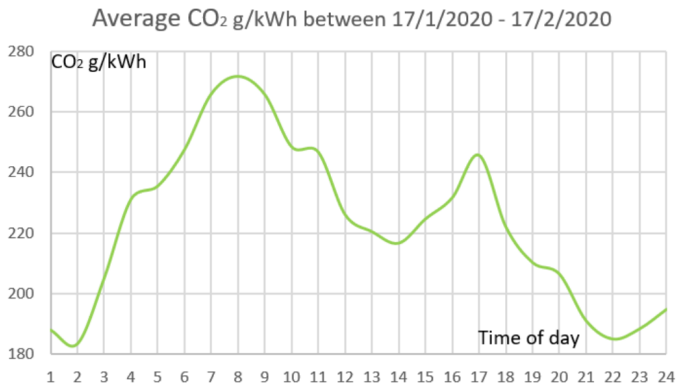


Figure 15: Average CO_2 emissions for each kWh produced

Just looking at both graphs, there seems to be some correlation, where the lower the price, the more environmentally friendly the energy is. A qualified guess is that the renewable energy sources are always turned on, while the none sustainable methods are only turned on when demand exceeds the production of renewable energy.

To test this hypothesis, the graphs were combined such that the price is graphed on the x-axis, and the CO_2 emissions are graphed on the y-axis.

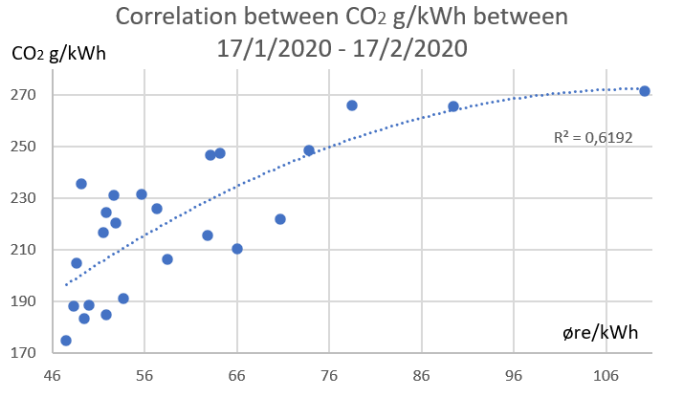


Figure 16: Correlation between price and CO_2 emissions

There does seem to be a trend, evidently not a very strong one with an R-value of 0.6192. The cheaper the price is on the average-case scenario, the more sustainable the energy production most likely is.
One thing that is important to note is that the prices and how environmentally friendly the energy is, do not follow each other daily. The market is highly chaotic since people might use more electricity during the weekend and less on weekdays. Sometimes it is sunny outside, and sometimes it is raining. Sometimes people are out traveling, and other times people may be jobless and constantly at home. This is before even addressing the elephant in the room: Different seasons of the year invoke very different consumer patterns. The greatest takeaway from this is that renewable energy and cheap energy throughout a long period seems to be somewhat correlated, such that the lower the price for electricity, the more likely it's also renewable, but assuming this on a day to day basis is unreliable.

Quick recap

The amount of variables to calculate is immense. Therefore a dynamic system that reacts to daily differences in consumer and weather patterns is necessary. No average graph will be a good guideline for very long. When that is said, getting both sustainable and cheap energy is plausible in a lot of circumstances. If one was to optimize solely for sustainable energy, in the best-case scenario, this could reduce Denmark's total CO_2 emissions by 3.6% while possibly saving money.

3 Limitations

This section outlines the limitations that have been set for this project:

Because this project is a second-semester report, and due to the Covid-19 lockdown restrictions we are confined by, developing an integrated hardware solution is out of the scope of this project. Therefore the hardware side shall be simulated.
Due to the complexity of the GDPR law and other privacy protection acts, they will be ignored in the development of this project.
In order to comply with the learning objectives for the project course, the solution has to be a web solution at its core. This report will therefore focus mainly on the web development, implementation, and databases side of a solution.

4 Methodology

In the **early phase of our project**, we explore the fields of smart homes and environmental sustainability by looking for articles on the internet to identify an issue that we can attempt to provide a solution to. Based on what we find, we do group-based brainstorming sessions until we have narrowed down the initial problem field into a more clearly defined problem that we can develop a solution to. Throughout the project, our **information gathering** consists mainly of searching the internet using any of the search engines "Google", "Bing" or "DuckDuckGo".

To broaden our knowledge on the **state of the art**, each group member performs internet searches for existing solutions pertaining to the issues we have identified. We then examine the features of each solution to establish an idea of which features our application should have, after which we combine the details of our findings in the section State of the Art, which describes the whole process in more detail.

In addition to this, we develop a **survey** to identify usage patterns among electricity consumers, including owners of smart devices. The survey is designed with questions to verify if our motivation for the project is sound, i.e. is our working idea a good one? Additionally, the survey contains questions that can provide us with guidance in establishing and ranking requirements for our application. The survey is sent to a part of our fellow students at AAU and posted to several survey groups on Facebook.

Lastly, we chose to make a **user story** to put our idea into perspective by giving a "real world" example of a fictive user called Alice. A user story naturally describes software features coming from an end-user and highlights what they want and why they need a specific feature. This helps to get a simplified description for one or multiple requirements [25].

Our **workflow** was centered around 4 - 5 **group meetings** a week in which we decided which upcoming tasks needed to be undertaken. This was done partly based on our understanding of which direction we wanted to take the project in, but also based on the **feedback** on previously completed tasks we received in weekly meetings with our **supervisor**. Once we identified and agreed upon tasks to be completed, we put descriptions of each task on the site called trello.com, and once enough tasks existed that every group member could find something to do, it was up to each member to decide which task they wished to work on. If some tasks were too large for one person to tackle alone, an appropriate number of people were assigned to work together on the task.

4.1 Waterfall model software development

The original Waterfall model has existed since 1956 [26] and has received modifications over the years. Originally, it consisted of several well-defined phases in a software development process, that had to be completed before one could proceed into the subsequent phases. Over the years, modifications to the initial version allows for looping back into previous phases if problems arise along the way (See Figure 17).

4.1.1 Phases of the Waterfall Model

The following is a brief description of the model's various phases in a software development project as it is described on tutorialspoint.com [27].
Requirements of the proposed product must be found and documented.
Design of the overall system structure is formed to incorporate the requirements.
Implementation of the various functions that are part of the system structure. This is where the actual coding of each part of the system takes place.
Verification of how each part of the system works and how they interact. Normally, this happens in tandem with users whose feedback can give rise to the re-implementation of specific parts of the system. In our project, we will perform the testing ourselves.
Maintenance is the life of the "finished" product. It is now in the hands of the users and subject to continual updates due to user feedback, the discovery of bugs, or a desire of the developers to introduce updated functionality.

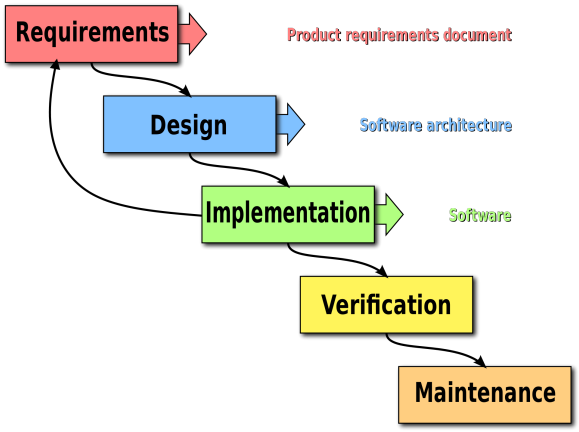


Figure 17: The Waterfall model structure (modified)[26].

Our intent is to follow the original model to a high degree, such that we aim to complete a phase before moving on to the next while keeping the option open to loop back and make corrections in earlier phases only if we deem it necessary. By doing this we expect to limit the number of changes we will have to make in the later phases of our project. Since the problem, to which we develop a solution, and the learning goals of our semester are both well defined, we choose the Waterfall model since it applies well to clearly defined projects, whereas other more agile models may be better suited for projects open to change or containing a less defined end goal. Two other models we possibly could have used, and the reasons for why we do not, are briefly described below.

4.1.2 V-model

The V-model, also called the verification and validation model, is a type of SDLC (Systems Development Life Cycle) is a model with many aspects in common with the waterfall model. The biggest difference is the focus on testing and verifying each phase's correctness in the development process. Each phase of the development process has its own test, that may be revisited at a later time if the development process so requires it. [28] This has the following advantages and disadvantages:

Advantages:

- V-Model is used for small projects where project requirements are clear.
- Simple and easy to understand and use.
- This model focuses on verification and validation activities early in the life cycle thereby enhancing the probability of building an error-free and good quality product.
- It enables project management to track progress accurately [28].

Disadvantages:

- Not suitable for complex and object-oriented projects.
- When requirements are not clear project can always change in time if there is problems with the project.
- This model does not support iteration of phases [28].

We choose not to use the V-model since verification and validation has to be done often. Trying to do countless tests can delay our deadline for completing our project, especially since the development period consists of a couple of months, then this model was seen as less suitable for our use case compared with the standard water fall model, which is more straight-forward in its progress.

4.1.3 Agile model

The agile model uses iterations to develop software. It works with the same structure like the waterfall model, other than that it iterates through the waterfall model numerous times in a short time frame [29]. Planning is typically the first step in the agile model, then requirements are analysed followed by designing the software, coding the essential pieces and testing to see if the developed application is working as intended. Then it repeats until a complete project has been made.
According to javatpoint, these are the advantages and disadvantages of using the agile model:

- Frequent Delivery
- Face-to-Face Communication with clients.
- Efficient design and fulfils the business requirement.
- Changes are always acceptable.
- It reduces total development time [29].
- Due to the shortage of formal documents, it creates confusion and crucial decisions taken throughout various phases can be misinterpreted at any time by different team members.
- Due to the lack of proper documentation, once the project completes and the developers are allotted to another project, maintenance of the finished project can become a difficulty [29].

One of the big difficulties, is that each developer needs to be highly experienced, in order to complete the development cycle without making a mess [29]. Our group was interested in attempting to do the agile model, but due to advice from our supervisor, we decided against it, and followed our supervisors recommendation of using a simpler model. As such we decided to use a modified waterfall model.

4.2 Requirements

By looking at the state of the art, digesting our survey responses, and from our user story, we gather a lot of different functional requirements.

4.2.1 MoSCoW model

To form an overview of the **requirements** of our application we made use of the **MoSCoW** model [30] by which we weighed the different possible **features** of our application **must, should, could**, and **would not** have. The method is to divide requirements into categories of different importance. By doing this our group can make better decisions about which features to implement first, and avoid spending precious development time on non-essential features. Because of the time constraints of our project, we had to consider which features were possible to implement and relevant to the scope of our project. This is why the MoSCoW model is highly suitable for our group, we have a lot of ideas and we need to prioritize them.

4.3 System Design

After we establish a sorted list of requirements, we continue along the Waterfall model and begin the software design phase of our project.

4.3.1 System description

A system description is a description of the different components and features that are based on the requirements and should give a sense of the system as a whole. In our system description we present the different interactions of components in sequence diagrams.

4.3.2 System architecture

The system architecture is the logical distribution of software interfaces made into different components [31]. Components that are closer to each other can be located on the same machine. There are many types of system architectures, but for our project we kept it very simple, just having a high-level representation of our software interfaces showing loosely how everything is connected.

4.3.3 Sequence diagrams

We develop sequence diagrams to think about the logical steps in which data is shared between different parts of the application, with internal and external sources. The sequences shows the order of execution which is really helpful in understanding what happens and when. We present two sequence diagrams, one for gathering and visualising data and another on the operations of a device.

4.3.4 Mockups

To design our applications user-interface without writing actual html and css code we utilise mockups. Mockups is primarily used to model the user-interface of a finished web-application without necessarily having any functionality, and is often used by companies to show clients what the website will look like before any implementation. We saw mockups to be most fitting for designing as they do not require a lot of time and resources to create. We used the mockup tool Moqups which has the functionality to easily drag and drop common html elements into a page [32].

4.4 Implementation

In the **implementation** phase, we use git and GitHub due to the file sharing and version control they provide which allows several group members to work in collaboration on the same piece of code remotely and independently of each other as required.

4.4.1 MVC model

In order to structure our application we employ the highly popular MVC design model/architecture which divides an application into three distinct parts consisting of models, views and controllers. Each part interacts with the others to handle specific functionality of the application. The models represent the data structures the application uses to talk to the database. The controllers serve as a link between the models and the views by using the models to access the database, potentially processing the data handled, before sending it to the views or writing to the database. The views are responsible for the way the data passed by the controller is rendered to the client accessing the server [33].

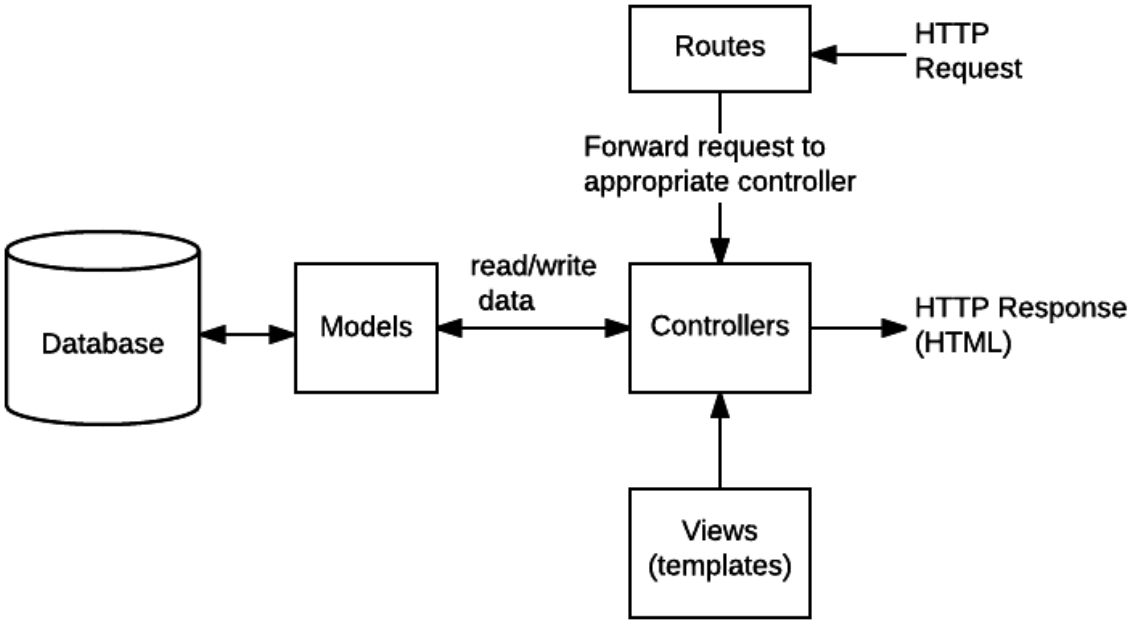


Figure 18: MVC model visualised [34].

We chose to structure our application in this fashion because we feel it helps us maintain a good overview of our code base. The individual parts are described in more detail in the Implementation section.

4.4.2 UML diagrams

Early in the implementation phase we compose UML diagrams describing the data structures we intend to use, which kind of fields they contain, and how they are interconnected. This way we have a better idea of what our code needs to look like before we start coding, and allows each group member to write different code pieces that should integrate more smoothly with each other. An example UML diagram is given in Section 8.3 regarding the database structure.

4.5 Testing

To test how well our implementation fulfills our requirements, we perform two types of tests: an end-to-end test during the implementation phase and a user test after the implementation phase.

4.5.1 End-to-End Test

We use Cypress to simulate a user interacting with our web application. These tests aim to verify whether basic functionality like login and adding devices works the way we intended it.

4.5.2 User Test

We perform a user test to provide us with information about the user experience of using our web application. To do this, we ask a small group of people to be our testers. They are given a guide detailing the steps required to perform the test which runs over two days, after which the testers answer a survey about their experience of using our web application. We use the responses to evaluate on the user experience of our app and reflect on

the design and implementation choices we made during development. The things that could be changed from attaining these insights is mentioned in Section 10.

5 State of the art

As time goes by, we humans develop more and better devices all the time, and more of these newer and smarter devices find their way into consumer households. So when we as consumers integrate these smart solutions into our homes, with smart-lighting or Smart-home Assistants as an example, we gain the ability to control or monitor our in-home devices remotely for a more convenient way of life. This smart concept is behind smart-home technologies with all these smart devices/appliances we use in the household. If we write this down as a question, how does the user effectively manage the desire to lower their CO_2 footprint? The answer is Home Energy Management Systems. In this section, we review some of the critical features of some existing solutions to broaden our knowledge in the field of smart home monitoring. We first describe the search process, then we chose not to look at the list of solutions in detail before describing the solutions we found most attractive along with a description of their properties. Because of our project limitations, we focus on software applications or platforms and briefly describe the hardware aspect. We compare our found solutions to differentiate them and set our base requirements for our solution.

5.1 Selection Criteria

General criteria for selection

We wanted to get an idea of which solutions currently exist in the field of home energy monitoring. The following solutions are selected to represent the current state of the art. In-Home Energy Monitoring Systems (HEMS), based on a series of criteria, we found it important for the solution to accurately represent the area we are trying to build our solution. The solutions have to have some visualization of energy monitoring and receiving data from a monitoring device. Furthermore, we wanted state-of-the-art to represent a multitude of different approaches to solving the same problem. We found many solutions, with some similar, if not the same, approaches or functionality. Therefore we picked the most prominent or developed solutions to represent that type of approach and method to solve the problem.

Search queries used to find solutions

Since we needed to find solutions centered around web-based applications, we primarily used search engines such as Google and DuckDuckGo to find the solutions. To find existing solutions that matched the criteria described above, we used these search queries to find results:

"*Energy Monitoring*", "*Smart Home Energy*", "*Smart home energy management system*", "*Electricity management system*", "*Overview of electricity use in homes*", "*Smart devices electricity usage*", "*Energy home monitoring dashboard*", "*Smart device energy*", "*Energy consumption home monitoring system*", "*Smart device energy consumption*", "*Smart home dashboard*".

We found more than 20 solutions in our research, which matched the criteria for being a state-of-the-art solution to our problem. Out of the solutions we saw, we chose to explore 6 solutions more in-depth.

Discarded Solutions

ZigBee Smart outlet was not included since it had a lot of overlap in features with the other smart plugs already described. Besides the lack of features, it is relatively expensive. To monitor all the devices in a home, you would have to connect every device to a smart plug to monitor energy consumption accurately. With a cost of 45$ pr. Outlet, it would become an expensive solution, and it would not be possible to monitor devices that

are not wall plugged, such as your stove and possibly oven [35].

Loxone is a complete home automation solution letting the user monitor and control just about everything in the house from lighting, heating, and energy consumption to solar panels and swimming pool temperature management. The solution comes with its app, which is similar in functionality to the apps provided by the other solutions. We chose not to include this solution as it did not seem to give any particular functionality that would make it stand out from the rest [36].

The **Yonomi** app seeks to integrate all the different smart home apps a user may have into one place from which they can control such that the user does not need to access every specific app connected to different brands of smart home devices. While this functionality is interesting in its own right, the app itself does not provide the energy management capabilities we were looking for in our problem. Therefore the app did not make it onto our list [37].

Generac provides the HEMS PWRview, which is a phone/tablet application that allows you to monitor energy usage and electrical waste in homes that have a Home Standby Generator installed. We have omitted this solution because it is centering around backup power and general home energy, which is limited to houses with Generac generators installed [38].

Efergy's key features include access via app or web portal, an overview of real-time and historical data, a graphical display of energy demand, and up to 5 individual circuits to monitor. These features are similar to many of our other reviewed solutions, although they were described less clearly, which is why we discarded Efergy [39].

Positive Energy Systems offer solar power installations for home and commercial use. Instead of having their solution, they combine equipment from various external suppliers. As such, they provided no home energy management system of their own and did not warrant further investigation [40].

Swell Energy offer installation of backup battery for the home and integration with existing solar systems. Their solution does not provide management of individual devices in the home and thus did not seem relevant to our project [41].

Solar Analytics offer detailed energy monitoring for home solar installations. They monitor electrical circuits and offer a web- and a phone app that provides information about live and historical energy usage, including peak usage and recommendations for aligning energy use with production. This functionality falls in line with our project compared to the strict focus on solar, whereas we want a solution that works for all homeowners. We could draw inspiration from examining their app, but as a whole, their solution does not provide the required detail about individual devices [42].

CarbonTrack enables the user to monitor both energies from the grid directly and individual appliances/devices through smart plugs. The collected data are communicated to CarbonTrack Cloud, in 15-minute intervals, over secure Telecom networks. We found that this solution was promising, but we discarded it due to the limited amount of information about the solution [43].

Constellation Connect is developed by Constellation, which provides energy and home security to US cus-

tomers. Their HEMS is a shared platform for their security and energy products, where the energy part mainly consists of data from smart plugs and a general overview of energy pricing. Therefore, since it is not primarily a HEMS, we have omitted this solution from selected works [44].

Bosch Energy Manager: This solution does provide an application to monitor in-home energy usage, but the primary focus of this solution is the intelligent redistribution of surplus energy generated from solar energy to family and friend's energy grid rather than the public grid. So the HEM system is more of a byproduct of their solution, and that's why we chose to discard the solution [45].

E.ON Home: The home energy monitoring system E.ON Home was discarded as a solution mainly because it is only accessible by the customers of the German-based energy provider E.ON [46].

EMS3 was not included in the list of selected solutions because it is not a solution that is accessible to private customers. The EMS3-system is a building for large-scale corporate projects or company energy monitoring [47].

Honda Smart-Home US: is that solution that was discarded because it is technically not a home energy monitoring system. Honda Smart Home is a complete smart home solution, which means the entire house is part of the solution. Since the energy monitoring system is not usable in homes other than Honda smart homes, we did not include it [48].

SquareD: we chose to discard the SquareD, primarily because it is not an independent solution and does not have its software application. SquareD's solution is hardware to monitor energy in the home and needs to connect to the Sense dashboard application to display the data [49].

5.2 Selected Solutions

The following section contains a description of the solutions we think have exciting features relative to our project. Each description is divided into parts, giving our reason for including the solution and its introduction, followed by application and hardware subsections where applicable.

5.2.1 Sense

We keep this solution because of its unique way of trying to solve the overall problem. Sense uses machine learning to identify devices/appliances in your home. Combined with smart-home technologies like Google Home, Alexa, etc., it can help the user to monitor home energy consumption at a device level [50].

Sense is a small energy monitoring box that is connected directly to your electrical box. It works by having current sensors detecting patterns in the electrical signals giving devices an electrical signature that uniquely identifies individual devices. It uses machine learning software to see these patterns and will be able to pick up when a house microwave turns on, or bulb lights are being turned off, etc. The algorithm learns these patterns and will, over time, recognize more and more devices in your home. The Sense device can be integrated with google assistant to provide an audio overview of the house energy consumption. Sense devices cannot be bought in Denmark, but they can buy it in America and Canada, where it costs 264 USD [51].

Application: The application consists of various monitoring overviews. The front page shows a list of current devices turned on or off and the total energy consumption of the current day. The application also allows seeing energy trends and can be compared with other Sense owners. One of the more important features is the ability to track energy consumption over time. This gives users a deeper insight into their energy consumption and can also be done on individual devices. Lastly, the application will try and predict the monetary cost of each device, making users able to see their energy bill beforehand.

Hardware: The Sense product comes with the main box that deals with computing and sending information to the application. The main box has to be installed by a qualified electrician and connected to the electrical mains with an antenna poking outside the panel. The box connects to two sensor clamps (See Figure 19). The clamps go over the two electrical mains and are clicked onto the wires, locking them in place. Once they are set up, they will send data on the current and voltage levels into the monitoring box that takes care of computing and sending the information to the application.
The smartphone application that follows, will not advise on actions to take, it will be up to the user to make decisions. The Sense will not be able to turn devices ON/OFF, and according to a blog post of a Sense user [52], the Sense has some difficulties detecting all devices and differentiating devices [50].

Figure 19: Sense energy monitoring device [53].

5.2.2 Smappee

We kept this solution because rather than focusing on specific devices, it monitors the entire house/apartment with its energy management system. The EMS is connected to the building's circuit boards, and the modularity of the hardware allows for customization. Through its web-based application, you can create views to filter and analyze real-time data [54].

The Belgian energy management company, Smappee, is a leading clean-tech company that helps its users save energy and money by reducing energy consumption and improving energy efficiency. Their primary product is the "Smappee Infinity" energy management system, which consists of several different modular monitoring devices. These are connected to your home by embedding the device to the central circuit board and from there

via multiple software solutions give you an accurate overview of home energy usage. According to SmarterLife, in Denmark it costs 2601 DKK (415,99 USD) [55] for a standard energy monitor package. Several other packages can upgrade Smappee, like the big package which costs 7791,95 DKK (1246,19 USD) [56].

Application: Smappee's Infinity system provides the users with a large amount of actionable data. Therefore the data needs to be processed and visualized for the end-user to benefit from it. The company behind Smappee provides three ways to connect. The Smappee system also provides an open API connection to integrate with any Home & building management system, a mobile application, and most importantly, a web-based dashboard application for in-depth analysis. The web-based application provides 7 sections of main functionality in Smappee provided application:

- *Analyze real-time and historical data*, The application provides several views and methods to monitor data from the user's home. In this application, you can monitor several parameters: Always on, Solar, Consumption, Minimum, Maximum, etc.
- *Access sub-metered data*, to monitor specific appliances and their real-time energy consumption.
- *Complete and edit configurations remotely* to manage your connected devices and hardware.
- *Customisation of data visualization* through the application enables you to filter and divide the large amounts of data that has been collected into segmented and dedicated boards. Allowing the user to monitor only what he/she wants to.
- *Monitor multiple sites* through a single application, which is useful if you have more than one home.
- *Export and save energy data* gathered from the Smappee system, and export/share the extracted data in multiple file formats, such as CSV, Excel, etc.
- *Monitor power quality* "With the Smappee Genius and a dedicated license, partners can monitor power quality using the Smappee Dashboard. Analyze minimum and maximum values within the interval of current/voltages with the Electricity usage card. On top of that, partners can monitor life, statistical or historical harmonics, and Total Harmonic Distortion (THD). Simply add the Live harmonics or Harmonics card on the Smappee Dashboard and make informed decisions." - Need to be rewritten.

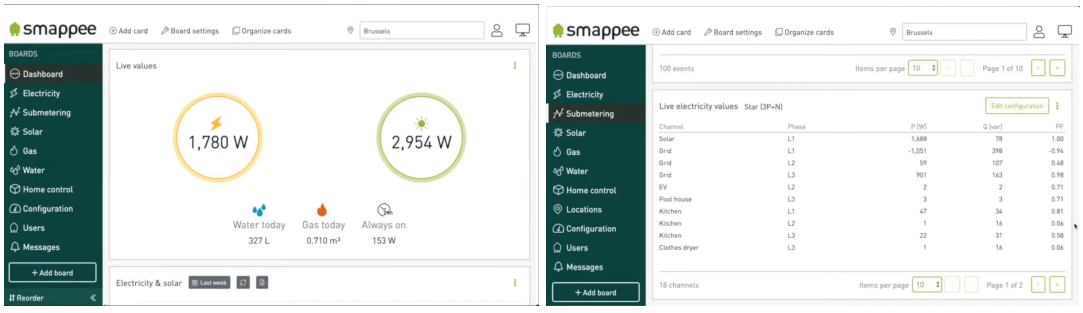


Figure 20: Images of the Smappee Dashboard Application, taken from demo video [57].

Hardware: This section about the Smappee hardware solutions aims to provide a brief understanding of how and where the data is collected and before being sent to API/dashboard applications.

- *Smappee CT Hub* which is the main component in the infinity monitoring system. The CT Hub can connect up to four Current Transformers and measure different currents up to 4,000 A. It allows for accurate

monitoring of a group of appliances. You could also chain connect multiple CT Hubs to measure more installations.

- *Smappee Power Box* Used for connecting the electrical network and other Smappee products. It also provides energy metering functionalities. Measures the line voltage of the connected phases, collects current from the Smappee CT sensors, and calculates power, active energy, reactive energy, and other energy and power data. Data loggers can collect this data.
- *Smappee Genius, Wi-Fi Connect, Connect* The three different modules (Genius, Wi-Fi, Connect) act as the gateway between the monitoring system and the data cloud and provides real-time data from all the components, by either Ethernet, Wi-Fi, or 3G/4G. Genius can use all three internet technologies, where Wi-Fi connects uses only Wi-Fi, and the standard Connect uses only Ethernet.
- *Secondary Modules* Smappee also produces other modules such as Input, Output, Gas & Water module, which provides more access, data, and control over the Smappee Infinity System. Still, they are not crucial to the Ecosystem.

Figure 21: Image of the Smappee Hardware Solution [57].

5.2.3 Eve

This solution is attractive because its approach to solving the overall problem is different from the rest. Eve, unlike the others, is not a circuit board integrated solution but relies on using smart wall plugs for monitoring selected devices/appliances and sending the data via Bluetooth to the users' smartphone application [58].

Eve is a company that offers a series of smart home devices, which elevate your ease of accessibility to the next level. One of the products Eve offers, Eve Energy, is fascinating in regards to energy consumption [58].

Application: Eve's application is a dashboard that provides an overview of the appliances connected to it. It gives the user with detailed information regarding the power consumption of the devices in the home. Furthermore, the Eve app allows the user to power their devices on and off remotely. The app comprises several customization options for the users as well. Users can create "scenes," which allows them to control multiple accessories with a single command. Other functionality includes timers and rules, where timers allow for automatically powering devices on or off. The regulations further increase automation by setting scenes based on

conditions provided by the user.

Hardware: Eve Energy is a smart wall plug, which is part of a wide range of smart home devices offered by Eve. One advantage Eve Energy has, compared to its competitors, is the use of a low-energy Bluetooth connection called BLE for short. In general, a Bluetooth connection is more power-efficient than a Wi-Fi connection, with BLE being even less power-hungry.

Eve's incorporation of BLE combined with the overview of standby power use of every device in your home and the ability to set an on/off schedule for all devices provides their users with a powerful tool to help maximize efficient power consumption in their homes.

Figure 22: Eve Energy wall plug and dashboard overview [58].

5.2.4 ONE Smart Control

We keep this solution because it provides a web-based information dashboard and an app, giving precise control of individual devices and setting up triggers to automate devices' usage through user profiles.

ONE Smart Control [59], from the Swizz-based Aizo Group, is a complete home automation system that allows any switch in the house to be monitored and turned on/off as desired. it is intended for users who want to automate large parts of their house instead of just monitoring or controlling one or more sub-circuits.

Application: The system is controlled with an app that enables a user to control a connected smart house in a variety of ways, including:

- Adding users to the system and assigning them roles granting different levels of control.
- Controlling individual devices or entire groups of devices as part of a scene or scenes grouped as part of a larger scenario.
- Automating devices' usage based on user-defined triggers.

- Monitoring production from solar panels and each connected device's energy consumption.
- Checking who's ringing the doorbell.

In addition to the phone app, there is a web-based dashboard that gives detailed usage statistics of the system (see Figure 23).

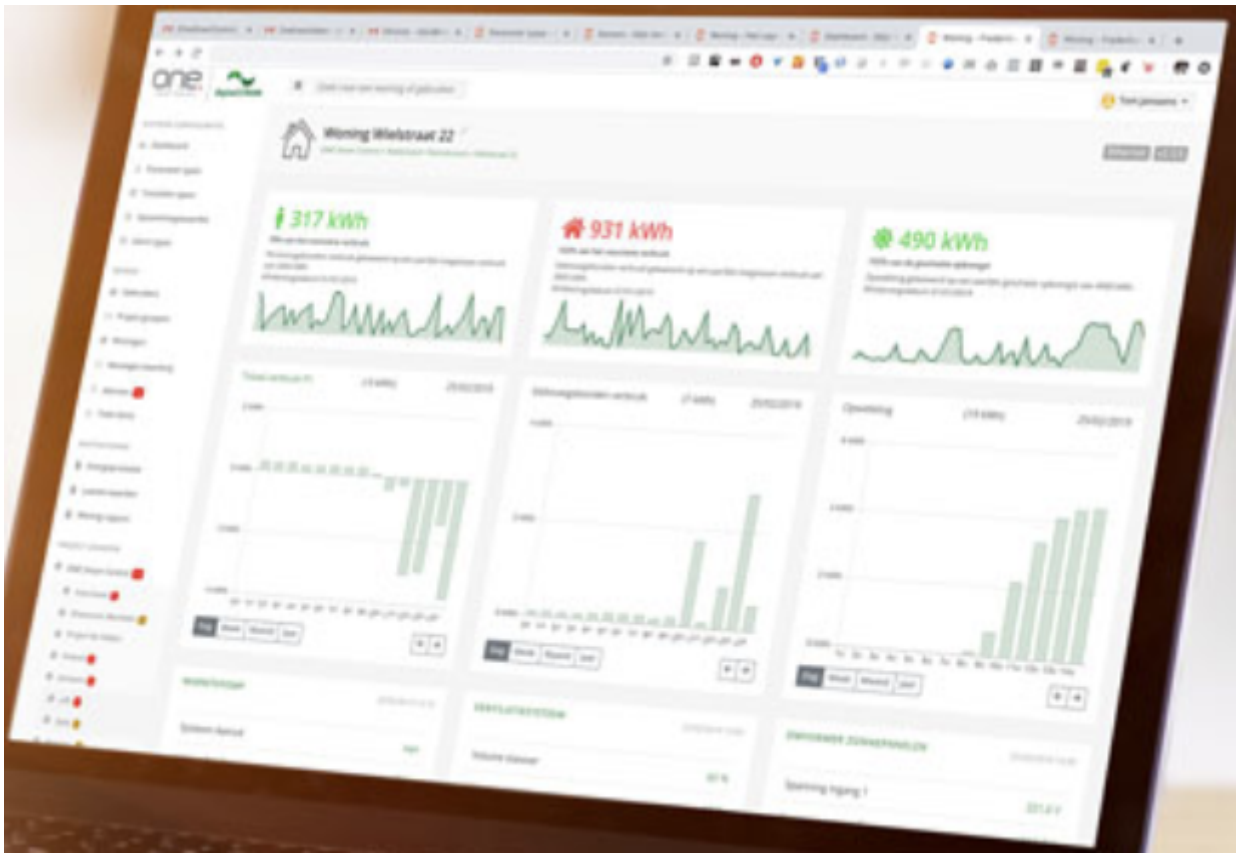


Figure 23: ONE web portal dashboard

Hardware: On the hardware side, ONE consists of several different components working together as part of a network as shown in Figure 24 connecting any electrical device in the entire house as the owner desires.

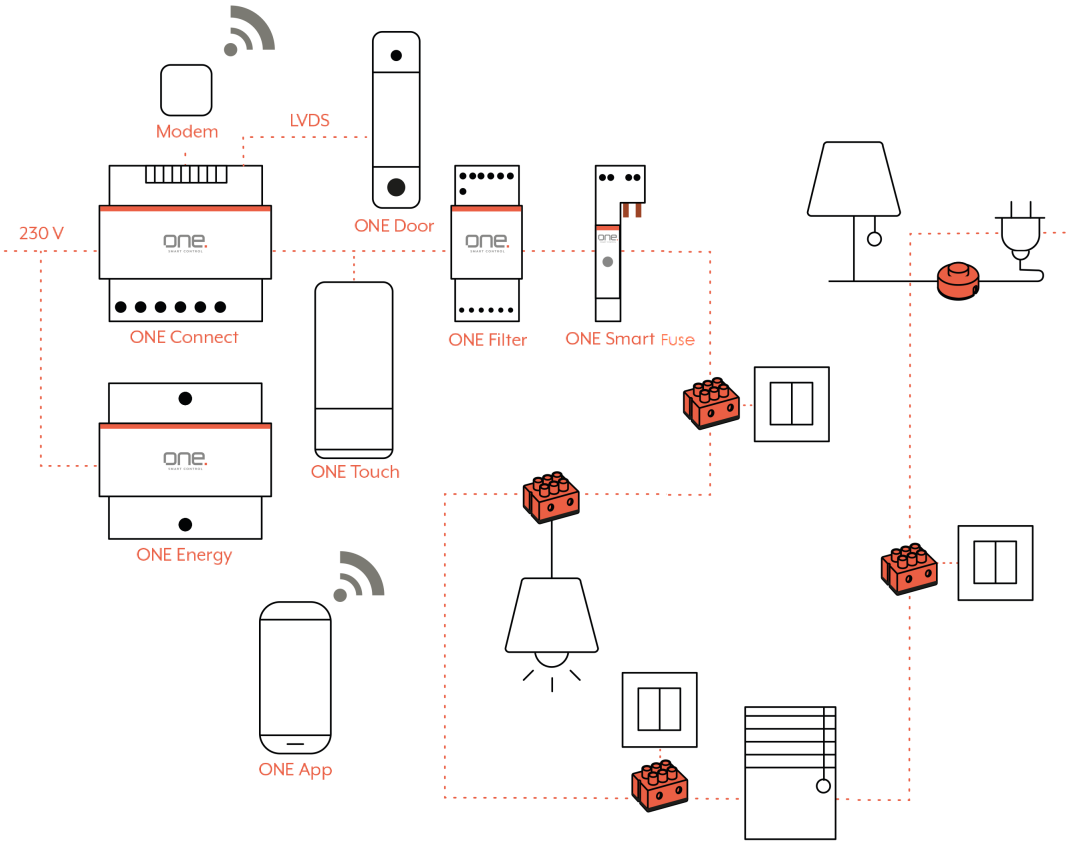


Figure 24: ONE installation overview

- A small **control component** is installed at every switch or device that the homeowner wishes to connect to the system.
- A **smart fuse** needs to be installed for every electrical circuit the owner wishes to make smart. Communication between the smart fuse and each control component happens through the electrical wiring present in the house.
- Between the smart fuses and the central gateway lies a **filter box** that filters out interference from the mains supply lines, which may be due to low-quality solar panels.
- The **ONE Connect** box is the gateway that handles communication between the smart home system and the internet, which lets the owner operate the system from anywhere (with an internet connection).

5.2.5 Homey

The Homey system was picked as part of the SotA because of its ability to track CO_2 emissions, the ability to turn on/off devices remotely, and its overall focus on the user's power consumption. The Homey system is quite different from the other selected solutions. The system relies on one single hardware component that allows the user to connect all his/her devices. Through the application, the user can analyze and monitor energy consumption, battery life, etc. The hardware device also knows when the user comes home or leaves and automatically turns devices on or off [60].

A Homey device is a spherical automation device, and it allows consumers to connect to over 50000 differ-

ent devices from over 1000 brands. This technology is also compatible with over 500 apps [60]. With the advanced technology called flow, it can detect whenever a person comes home and opens the door will light up the bulbs in a particular area, and lights will also adapt to the time when evening or morning. Communication with each device will further expand how much usage of energy, so they know how to **save energy** and pay the smaller bill next time.

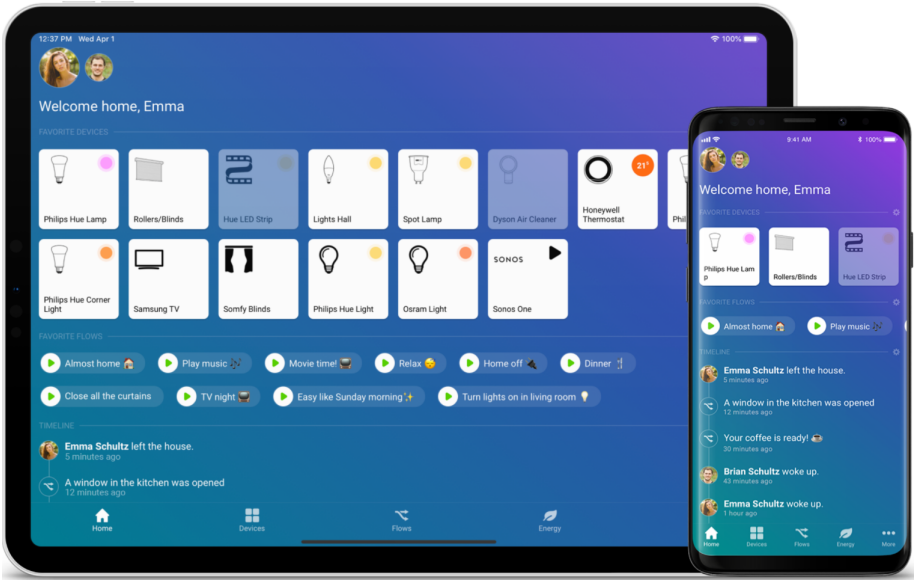


Figure 25: Homey insights - smartphone and tablet[61].

Application: All the functions are going through either the app by voice, or it can be seen with Homey insights to control all lighting, energy, etc., that has been connected through an app or install on a computer. After installation, the user must log in to do not enter when others are close to the Homey device. It can, for example, connect with their google smart home or Philips hue bulbs [60]. Homey allows people to communicate through either Google Home and Amazon Alexa and have conversations if necessary. It also lets its users turn on or off the smart devices they have, so there is no need to press a button because of the hands-free feature [60].
Not only will Homey help consumers save energy, but it can also:

- Know about the wind angle
- Measure CO_2
- See the temperature for devices
- See the power consumption of devices in real-time
- See battery life

Homey offers many more optional features that can help everyday life at home. All these data will be kept safe for 10 years if needed and be deleted if the consumer wants to do it [60]. Graphs and GUI figures can be displayed when using a smartphone or computer. All of the data are then going to the brain, and that's the Homey core, run by Linux distribution based on Debian, a non-commercial free software-control system. It is coded in JavaScript where it was from the Node.js application.

Additionally, its features come from an API where the Homey Core will create a webserver for this specific task. With the Javascript in place, the front-end task is also needed to develop figures, pictures, buttons, colors, etc.,

and the Homey developers have used HTML and CSS. Even the developers of Homey or other people can not gain access to most of the API without authorization, so only the owner and the people the owner have granted access to control Homey [62].

Hardware: Some of the technical specifications on how the Homey communicates is listed below:

- The Homey has a Zigbee and Z-wave to use communication protocol.
- Wi-Fi and Bluetooth is a well-known way to connect devices which is also available.
- It possible to connect with two frequencies, 433 MHz and 868 MHz, that are connected from the Microcontroller. The Microcontroller also controls the LED lights.
- Infrared if the consumer is using an old device to connect with Homey [62].
- The heart of sending all communication is the MicroSOM chip with various interfaces like USB, I2C, UART, and GPIO.
- Homey developers states that *Its IMX6 processor is clocked at a maximum speed of 1.0 GHz and has 512 MB RAM available.*, this will allow many users to simply connect many devices and have them turn on at the same time.

5.2.6 Lumin smart

Lumin primarily focuses on Solar energy, but we kept this solution because it allows the user to monitor how much and when most solar energy is produced and allows the user to schedule when to use it to improve home energy efficiency. This means that Lumin is a solution that can be used if the user wants to limit their households' CO_2 footprint [63].
Lumin smart panel has an app that lets the user control other appliances from a smartphone, and this is possible because Lumin converts an ordinary circuit into a smart circuit break panel, also known as a smart home electrical panel [63]. Instead of manually checking all regular devices, with Lumin, people can quickly look at how much solar energy is being produced, current battery energy, electricity pricing, and even schedules to improve home efficiency. With this technology, owners can decide without knowing what to do because the software has the information.

Figure 26: Lumin setup [63].

Application: In their application, the user has the optional choices for controlling energy since it could be nice if saving energy in a house when compared to an apartment. All the energy data is being stored, and the data will save on the Lumin insights [64]. There can be scenarios where users forget to turn off their appliances. Lumin can detect these abnormalities and alert the user to save even more energy [63]. The list below shows what else the Lumin app can help users be more energy efficient in a home.

- Enables Whole Home Battery Backup
- Right Sized Energy Storage
- Real-time Energy Monitoring
- Intelligent Load Management
- Solar Power Management
- Power Controls
- Customizable Controls for each device that is connected to Lumin from the circuit breakers
    1. Safety Alerts
    2. Money Saving Schedules
    3. Advanced Energy Insights that can monitor energy consumption [63].
- Easily automate your home's energy use by setting up schedules and modes from the app.
- Lumin can deliver substantial energy savings and make it easy to manage complex utility tariffs such as time-of-use rates and demand charges [65].

Hardware: The smart home electrical panel is being created not just one, but for each device that can connect with Lumin. According to Lumin, there are of the more preferred energy management among the top solar installers because users don't have to replace their load center or electrical panel in the home [63].

But how is Lumin installed, and what happens to the regular products? The method is that Lumin connects to other devices. Connecting to other devices will allow Lumin to access the energy storage and control it from the Lumin smart panel.

5.3 Comparison of selected solutions

Below is a table that shows what each solution can do compared to the others. We have marked them with colors to indicate what they can or can not do. Based on the colors, green means the solution has the characteristic. Red means the solution does not have the characteristic. This table is also based on what we have researched on the internet.

System	Remote Energy Monitoring[1]	Remote Control[2]	Action Advice[3]	Time Scheduling[4]	Smart Compatible[5]	Green Incentive[6]
Sense						
Smappee						
Eve						
ONE						
Homey						
Lumin						

Table 2: Comparison of existing commercial products

1. Ability to monitor energy usage independently from the meter's location over Wi-Fi or the internet.
2. Ability to control devices through a PC or smartphone over Wi-Fi or the internet.
3. Advice or recommendations given by the smart home system regarding the use and state of devices.
4. Ability to automate the activation of devices based on a condition or specific time of day.
5. Whether the smart home system can communicate and work together with other smart systems in the home.
6. Whether the smart system monitors sustainable energy production and electricity pricing when providing advice or recommendations about device usage.

5.4 Hardware Summary

The solutions we have looked at have several different hardware models that determine how they interact with devices in the home, either through the house's electrical wiring or wirelessly through independent, smart plugs. Even with the smallest detail, we have to check the communication protocols, sensors, connectivity, etc. and know the pros and cons. This is very effective for solutions that provide users more control of the devices they have. Some devices can let users see the energy consumption of devices where all data is generated from the hardware and processed with the help of the software if the device has that feature.

The first two solutions from the report are quite similar in that they primarily monitor devices' usage. They do this through a hardware component connected to the power meter, which can learn to "read" electrical signals from each device in the home. In some sense, both appliances do contribute energy savings. Still, there can

be different ways to do that, and Sense uses a simple method with a hardware box with antennas in the users' electric panel to send signals regarding energy consumption from another software cloud service - these cloud services come from either a smartphone or computer. Smappee is still using multiple appliances through a single application, which can cost many users who want to upgrade their smart home to a bigger scale, and sense can be cheaper with only 1.650,84 DKK vs 2601 DKK. Smappee can still do a lot more if it is upgraded.

In Smappee's and Homey's case, device control can be added by using smart plugs that communicate through Wi-Fi or another protocol like Zigbee, Z-Wave, or Bluetooth. Both of them has a small hardware piece, that can transmit signal from another device. Lastly, both of them have a way to advise about electricity usage to the user if he/she is using spend way too much than an average house or apartment.

ONE's solution does both device monitoring and control through the electrical wiring, while EVE uses only smart plugs to do this. Lumin, on the other hand, will only accept smart circuit breaks of their own that will afterward go straight to Lumin's smart circuit break panel. Eve can still use its smart plug effectively for one product, but there can be difficulties plugging more than one device since people need another one for two devices. All solutions apart from EVE and Lumin send device data to a central hardware unit, making the device data available to the user through an app. In contrast, EVE's smart plugs communicate directly with the phone app. The appliances still have in common that remote energy monitoring is a big element in the smart home industry. Without it, it can potentially lead to bad products for the consumers. This is why remote energy monitoring is essential for our solution.

5.5 Application Summary

The solutions we looked at, all had common elements. All applications had a graph for showing the energy consumption, either as a chart or diagram. Some of them can delete the data if it is no longer relevant for the user, and this is why many smart devices, from what we have gathered, can automatically delete the data in a set period of time. The solutions that could control the on and off state of devices had features scheduled in time based on different events.

Users can download an app or a control panel on a PC to control appliances and the energy consumption with Lumin and Homey. All smart products are designed to have a dashboard as the home screen to have a good overview of their smart devices.
Lumin can control devices using physical smart circuit breaks, register how much solar energy is being produced, and enable whole-home battery backup. Still, Homey can download apps and connect other smart devices without other hardware by using Bluetooth Wi-Fi, ZigBee, etc. In the end, there are different ways to establish a connection between devices and smart homes. The same goes for Smappee, Sense, Eve, and One Smart Control with monitoring energy in real-time. Though Sense has an addition to comparing other Sense users and predicting the size of the electricity bill for each smart device, the other competitors are lacking software-wise. Since Smappee focuses on energy consumption, there are not many other things it can do. Still, it can come in handy if people are interested in seeing input, output, gas & water modules for knowing where the energy is coming from. Eve works pretty well with a single product to control it from the app and is compatible with the Apple HomeKit. With the app, users can execute a command for multiple devices or a single one to save energy, where some of the competitors like Sense and Lumin struggle with this. There is a chance when someone is using it for fun and accidentally press something that they shouldn't; this is where ONE Smart Control can have different users where some of them have limited access to some controls in the home.

Above all these products have different views for helping the everyday life in a house or apartment. Each of them possesses at least three pieces of software that are relevant for energy consumption according to the table in Figure 2, and these have components for a solution to guide users to use less energy.

Recap

There is a great deal of overlap between the solutions we have taken a look at. They all provide the option to monitor energy consumption, and many of the systems can control the individual devices on a timed basis. Two solutions: Sense and Smappee are all in one solution, meaning you have one device to monitor every device's energy consumption. The others differ in that they have to have a smart plugin between the device and wall to measure energy usage. In general, the dashboards for the applications can show the current and past energy consumption and can make predictions of future consumption along with the price of electricity.
However, none of the existing solutions we have examined appear to provide information about the green energy production in the user-specific country or the cost of electricity for consumers who have non-fixed prices. This reveals an opening in the market for an application that provides its users with information about the availability of renewable energy and recommendations for when it is optimal to consume electricity.

What did we learn by looking at state of the art?

We learned the following from studying state of the art:

- No one markets themselves based solely on sustainability, instead the companies prioritize monetary savings.
- It is possible to map what devices are active based upon their electrical signature.
- Using a hub for data collecting and visualization and relaying to an app for remote control use is an industry standard.
- Automatically turning things on and off based on patterns and user's movements, including a dashboard where you can see recommendations for what you could do better, is state of the art.
- Using plugs to monitor every device individually quickly scales to become very expensive when every device needs it. While the hardware units connected directly to the power line is an expensive one time purchase, it easily scales for the entire house with no added costs.

6 Requirements

In this section we will be going over the requirements that we have gathered and will justify why we think these specific requirements are actually requirements for a solution to our problem statement.
The requirements fell into the different categories based on: What we have seen so far in Section 5, what ideas we came up with as a group, what was reasonable within the time-frame of the project, and what we thought could be implemented given the technical skills we obtain during our semester.
The requirements inside our "Must have" cell are all considered vital functional requirements for the project. A short description of what these "Must have" requirements entail, including a short justification for why these requirements are "must have", will be provided in the following section.
The other cells are a mixture of both functional and nonfunctional requirements. None of these requirements are set in stone, and therefore we will not give a description of these requirements.

Items described in "**Must have**" are the requirements that must be met, before this project can be considered usable: The Minimal Viable Product (MVP).
Items described in "**Should have**" are the things that the product needs before this project can be considered desirable.
Items described in "**Could have**" are the things this product needs in order to be considered ready to be used by society.
Items described in "**Won't have**" are the things this product does not need, but if there is time would put the cherry on top. [30]

Must have

- Update graphs according to the current sustainable energy production through an API provided by energidataservice.dk.
- A visual indicator for the amount of CO_2 the user has saved, the environment from.
- Add and remove devices from user profile
- Store device energy data in a database
- Advice system/algorithm to give feedback on how the user can use energy in a more sustainable manner
- User registration and login that saves user data in a database

Should have

- High-level dashboard overview
- Editing information on currently added devices
- Day/Month/Year energy consumption graph
- An option for which notifications the user wants to receive.
- Encrypted user login

Could have

- Update graphs according to the current electricity price through an API provided by energidataservice.dk.
- Monthly cost prediction of electricity consumption
- Dynamic updating of the web-application
- Add different roles for commands - administrator and default user
- Product information on each device that is added
- Categorisation of devices to be split into different rooms
- Option to be alerted of anomalies in device usage through pattern detection (e.g If you have gone to bed without turning off all the lights as you usually do, then the user will be alerted)
- Mobile app support

Won't have

- Remote control access of house devices
- Custom preference configurations
- Option to automate device usage through pattern detection (e.g turning off lights when you normally go to work).
- Add or invite more users to a home (e.g granting them the same possibilities as the house owner)

6.1 Requirement justifications

The following list gives a description and a justification of the must- and should have requirements taken in chronological order from the requirements list in the previous section.

"Update graphs according to the current sustainable energy production through an API to "energistyrelsens" website."

Description:
An interactive visual feed for what the current sustainable energy production looks like.
Justification:
During our survey, we found that 77.7% of the respondents were inclined to postpone their power-hungry activities in order to reduce their CO_2 emissions. Providing the end user with a visual representation of how the current energy production looks, will give them an opportunity to make conscious choices concerning their power consumption.

"A visual indicator for how much CO_2 emission has been saved."

Description:
A visual feedback that shows how much the user has saved the environment for compared to the average person.
Justification:
Due to the consumers' interests in keeping their electricity usage down with a high inclination to reduce their CO_2 emissions, keeping track of how much our solution contributes with for their specific household, we hope this will help develop proper power consumption habits for the consumer. The user needs to see, that our platform is actually working, and making a difference for the environment.

"Add and remove devices from user profile"

Description:
Add and remove devices that the user wants to monitor the power consumption of.
Justification:
From the responses to our survey we concluded that the consumers were willing to move several different power-hungry activities. Providing them with the option to add and remove different devices, would allow them to customize our solution specifically to their needs. A family of seven could be more inclined to modify their laundry habits, whereas a student living alone might be prepared to cook dinner at a peculiar time of day. Furthermore, the user needs to be able to add their devices in order for the platform to monitor, analyse and provide feedback on their energy usage.

"User registration and login that saves user data in a database table"

Description:
A register/login option that saves account information and user data to a specific account.
Justification:
This is important since in order to save added devices, and make sure no one else meddles with their household devices, and to achieve a certain level of privacy, there needs to be a login. This is also important in order to differentiate between users. Having an account on a platform is implemented in all of the researched state of the art solutions and therefore became part of our must have requirements.

"Advice system/algorithm to give feedback on how the user is doing"

Description:
A recommendation system that gives advice on which things the user needs to improve on the most, and what the user can do to improve it, in order to achieve optimal energy usage.
Justification:
Providing a feature like recommendations and advice, could be very useful for the users, since that they could actively make more sustainable choices with less effort, based upon their current energy consumption habits.

"Store energy data in a database table"

Description:
The data collected from the API and the users energy usage data will be stored long term.
Justification:
The data needs to be stored long term for other users to compare their performance with the average. It also needs to be stored in order for the web application to make proper recommendations based upon long term data.

"High-level dashboard overview"

Description:
A higher-level dashboard gives more opportunities to users, this provides a better experience and an easier navigation between the different functions of the dashboard.
Justification:
The research in the state of the art section showed that a lot of the solutions had high level dashboards. This will provide a better experience for users, as there will be no confusion of where are the functions and settings are located.

"Add product information to each device."

Description:
Product information on each different device, to see all device specifications.
Justification:
If a person has bought a device, they could promptly enter the new device information on the web application, where the information would be available for later use. Say if the same person had thrown away the packaging, then the would have the device information stored in the web application.

"Edit information about devices."

Description:
A method to edit information when update or personal preferences are in motion.
Justification:
After users have added some devices to the dashboard some of the information can be outdated, therefore it should be necessary to edit information so in the future there will be no errors, and the smart devices are always up to date.

"Graph of energy consumption in a set period of time."

Description:
Each year, month and day is not the same, this feature will get a graph energy consumption which let users see the day, month and year report.
Justification:
To have an impact of saving energy, people need to be guided on how to use smart devices efficiently. Giving them a realistic view of how much money each individual is paying per month, and by that providing them with graphs is a good visual effect to letting users know to change their way of using smart devices and reduce CO_2 emissions.

"Predicting the cost of electricity usage in a month."

Description:
Prediction is done by mathematical probability. After calculating this will, then we could present the results of these calculations in the web application through the users dashboard.
Justification:
Information about electricity price in real time is good but predicting the next month is better because it give users an idea if they need to change how to use smart devices and result in reduced CO_2 emissions.

"Choose what notifications users want."

Description:
Manage what should be in one's notifications. If there were to many or to few this could have an impact of the CO_2 emissions for each smart device.
Justification:
People are noticing notifications because it is easy to spot or hear with a sound, but balancing the notifications is important so the users will not be spammed all the time, so giving them an opportunity to customize notifications can effectively give reduced CO_2 emissions.

"Encrypted user login."

Description:
Encrypting users passwords, generates obscure and undecipherable hashes. This helps to ensure the safety of users private passwords. Third parties will be unable to view the passwords in plain text. The encrypted passwords is also inaccessible for developers.
Justification:
Encrypted user login method should help people to trust more the application rather than all the data can be accessible to other people. Another thing is that hackers can access a database from the application, but they can not find or do not know a user password.

6.2 User story

Alice is the mother of three children, who often play in the backyard. The kids like to play in sandboxes and the mud. Their clothes often come back dirty and must be washed in the washing machine. The family gathers around after a playful day for dinner, where afterwards they clean up the kitchen and load the dishwasher. Alice then relaxes in front of her TV and listens to the latest news. Alice is very conscious when it comes to saving electricity, and she wants to do good for the planet by reflecting on what she can do to lower her CO_2 footprint.

Alice thinks back to an article she read, about rescheduling the use of electrical devices, which helps the energy grid not overreaching its capacity. She then wonders if there were appliances in her day that she would be willing to reschedule and says to herself: "*Maybe I can change the time at which I start my washing machine and my dishwasher too!*". How can Alice know the time of day to reschedule these appliances?
She searches this question online and finds that around 8:00 AM when everybody wakes up and around 6:00 PM where everybody eats dinner, there is a sharp peak in energy demand. These peaks in energy demand often result in powering up fossil fuel power plants which leads to higher CO_2 emissions.

Alice wants to do good and therefore wants to shift the active time of certain appliances that she does not

feel have to be turned on at a specific time, counteracting the demand peaks. She begins to figure out which appliances, she is willing to move and writes them down. Alice looks up the renewable energy production forecast of the day and sees a surplus at around 4:00 PM. At that time, Alice decides to start her dishwasher and drying machine. "*That was easy!*", Alice thinks to herself. She wonders if she can do the same for more appliances other than the more obvious ones. Alice starts a new routine where she would look up the forecast in the morning and make a plan for all the appliances she can move. After a few days, Alice begins to get tired of checking and planning herself and looks online for a solution. She finds this very cool app (our app) which does just what she hoped for. The app shows her the forecast and current renewable energy production status, furthermore she can add her appliances with some information on their use. Alice finds it very convenient that the app can give recommendations for certain appliances she can move to align with the energy production. To make it even easier for Alice, she turns on reminders to be reminded of the peaks and recommendations. She does not need to move a finger anymore, and gets into a good habit that makes her feel good, and makes the environment feel even better.

7 System design

In this section, we look at the software design elements for our solution based on the requirements from the previous section and make a description of the entire system. We look at the high-level overview of the components that is required for building the application and then go into detail on the interaction of user, client, server, database and external API for different use-cases using sequence diagrams. We then show the process of designing the user-interface going from brainstorm to a mockup and how we ended up with an interaction diagram showing how the user navigates our site.

7.1 System description

The application will have a dashboard, from which graphs and information will be delivered to the user. The application's features will be hidden from unauthorized users by an authentication system. Users have to register as a user before gaining access to the functionalities within the app. The dashboard will contain a live-updated graph of the current renewable energy production, CO_2 emissions and home energy consumption, which more detail-oriented users can look into, but is not required since the graphs will be accompanied by small status cards with informative text, informing the user if there is an abundance of renewable energy. The application will include several other cards, described in Section 8.6 in more detail. Registered users can add devices to their profile to be monitored, which helps create a detailed picture of the users' energy consumption. The user can also choose to update information on devices they have already added, in case they change the usage pattern of the device. Users will have a complete overview of their added devices in a detailed list. The application also contains a settings page that allows users to set a climate goal which will determine how much they want to engage in shifting devices and how high or low the carbon emissions have to be before the user can begin to use electricity. There is also an option to update their personal information, should they need to.

7.2 System Architecture

In general the users interact with our application via a front-end interface which handles elements such as button clicks, showing forms, styling graphs and the structuring of the page elements.
The back-end of the application handles all the validation of forms, routing of incoming requests, communication with the database and fetching information from an external API. The back-end consists of a web server that will be serving the different pages, a file system for storing static assets such as: HTML, CSS and images files, and a logic component for handling and processing incoming data before rendering it to the user. The following diagram shows a high-level overview of the interactions between different components.

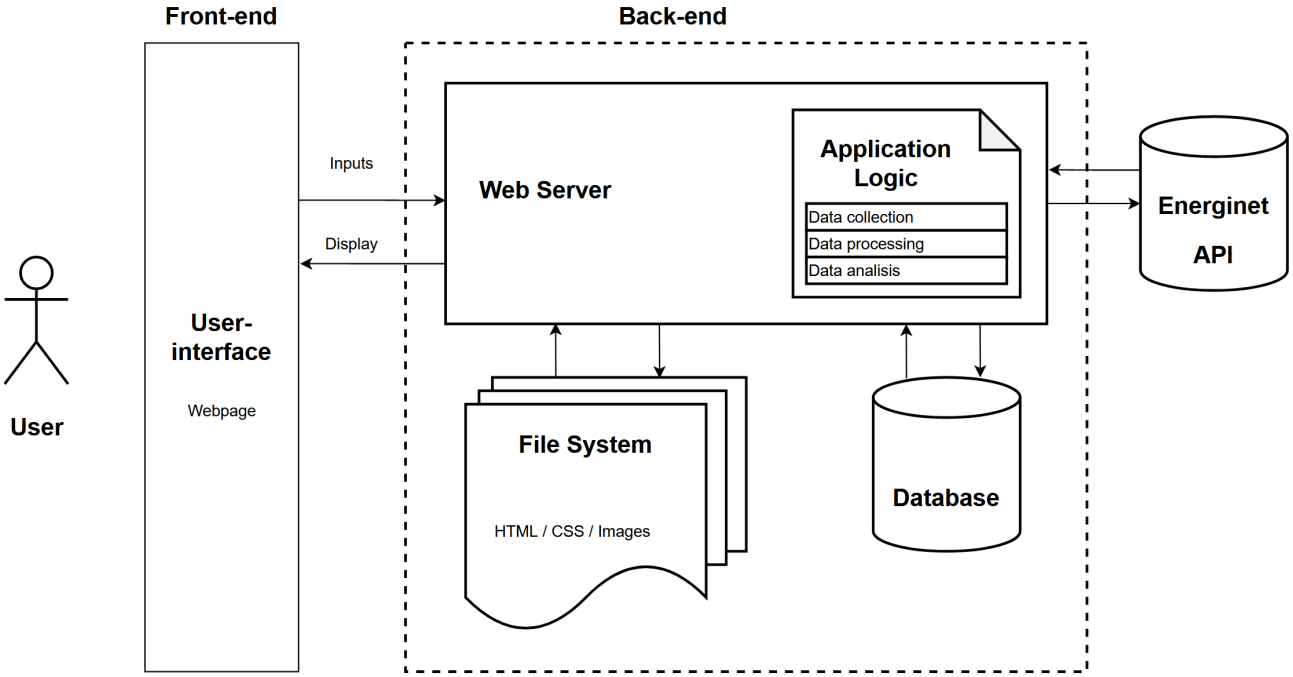


Figure 27: High-overview of the system

As seen on Figure 27 the user interacts with the webpage located on the front-end through their user-interface. The inputs which the front-end receives from the user is then processed by the back-end, which uses a combination of API, application logic, HTML, CSS and database communication to display the content back to the user.

7.3 Sequence diagrams

The following sequence diagrams illustrate different features of the application with different use-cases. We use sequence diagrams because they show interactions between different user actions and the back-end responses to these actions. This helps us better grasp the functionalities of the application. The sequence diagrams for login and registration can be found in Appendix D as they are a generic feature for most applications. Gathering and visualising data from the web is a vital part of the application, we chose to make a sequence diagram over this specific part.

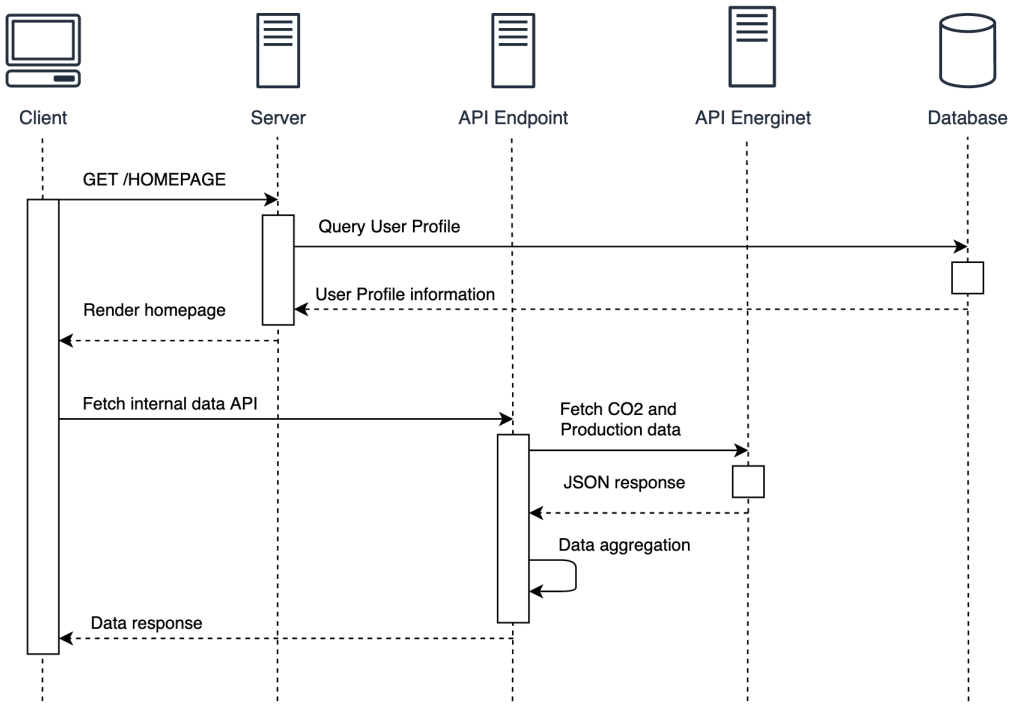


Figure 28: Sequence diagram for live-graphs

The sequence diagram on Figure 28 shows how the process starts with an HTTP request for our homepage dashboard. This request is handled by our server and makes multiple API requests to our data-provider energinet. We wait on the energinet API to respond with data before sending the data to one of our internal API endpoints. We do this to not halt the user's request by waiting for data but instead render the page without data, and then make the client fetch from the API endpoint when data is gathered. This way, all the page elements are loaded quickly, while the larger data request to energinet is loaded when ready.

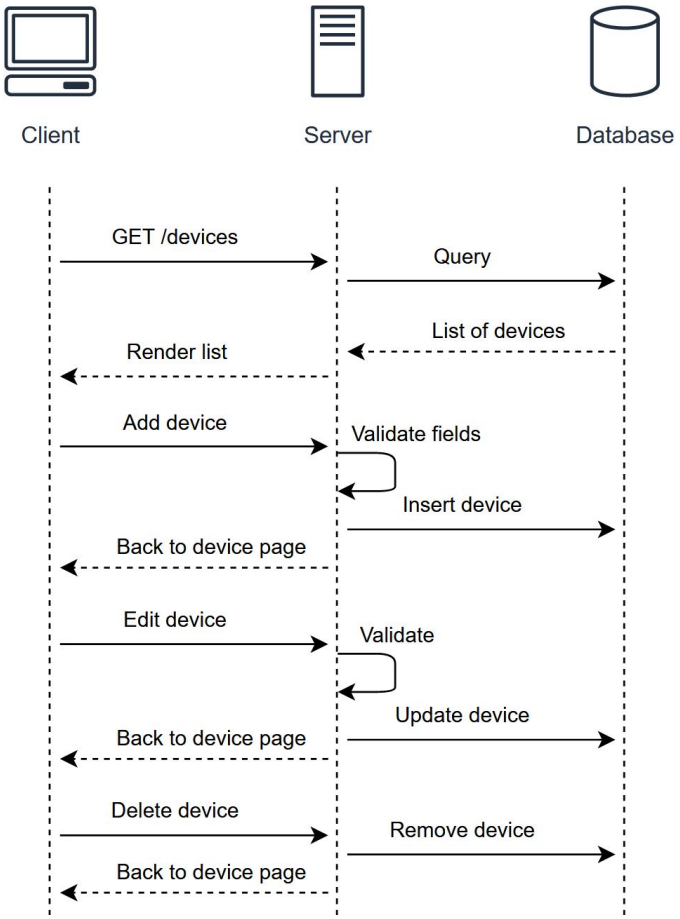


Figure 29: Sequence diagram for device actions

On Figure 29 a sequence diagram detailing some of the functionality, which the device page includes, can be seen. First the client sends an HTTP request for the device page, which is handled by the server once again. We then make a query to the database which returns all devices belonging to the user making the request. The next three actions depicts adding, editing and removing a device from our database respectively. The processes for these are quite similar. When a user adds a device, all the form input is sent to the server and validated. If the validation test is successful the device is inserted, if unsuccessful the server sends back an error message of what failed. The editing process is a little more elaborate, with different checks that prevent the user from having two devices with the same name. The update command for the device is then sent to the database. The last operation the user can make to their devices is removing them. When the user wants to delete, an alert-box will pop up to make sure if they really want to delete the device. If so, a deletion command will be sent and the user is taken back to the device page.

7.4 User-interface design

This section covers the prototype and the initial sketch we made to envision the final design. We started out with basic brainstorming on the design and came up with the initial sketch below:

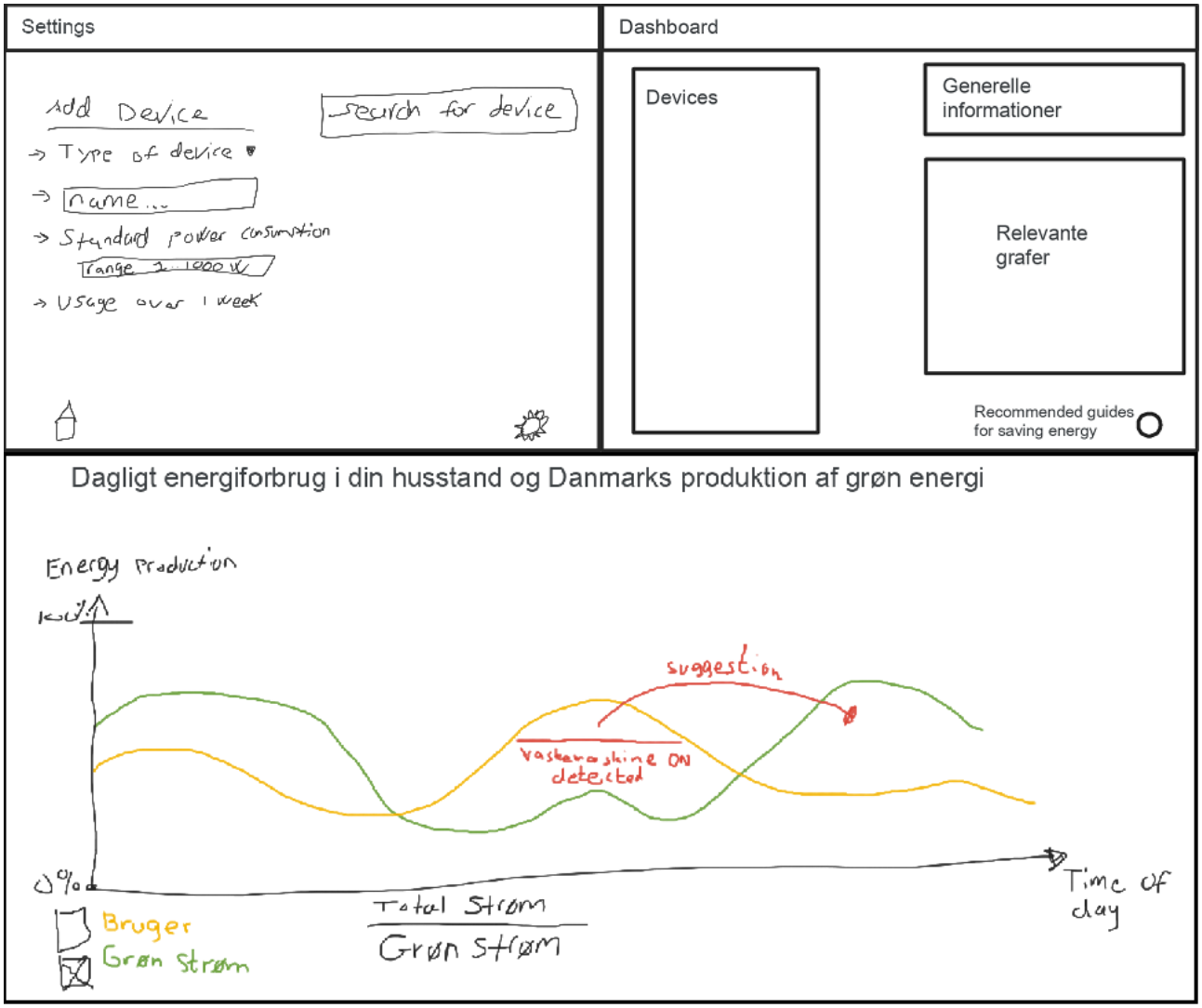


Figure 30: Initial brainstorming of the different pages

Figure 30 displays our initial draft, and although it looks a bit messy, it does show some of the basic functionality and core concepts that we want to include in our application. For example, we want to show a list of the devices that the user has added and a settings page for adding these devices. Another important aspect is the hand drawn graph, which shows the renewable energy production along with the specific user's energy consumption which, along with CO_2 monitoring, is central to the idea of what we aim to do. After having an initial thought about the looks of the application, we went on to develop a little more sophisticated design of the web application. This is done by making a mockup of what we want the web application to look like. The mockup on Figure 31 below shows our dashboard page and is made utilising a mockup program called Moqups [32].

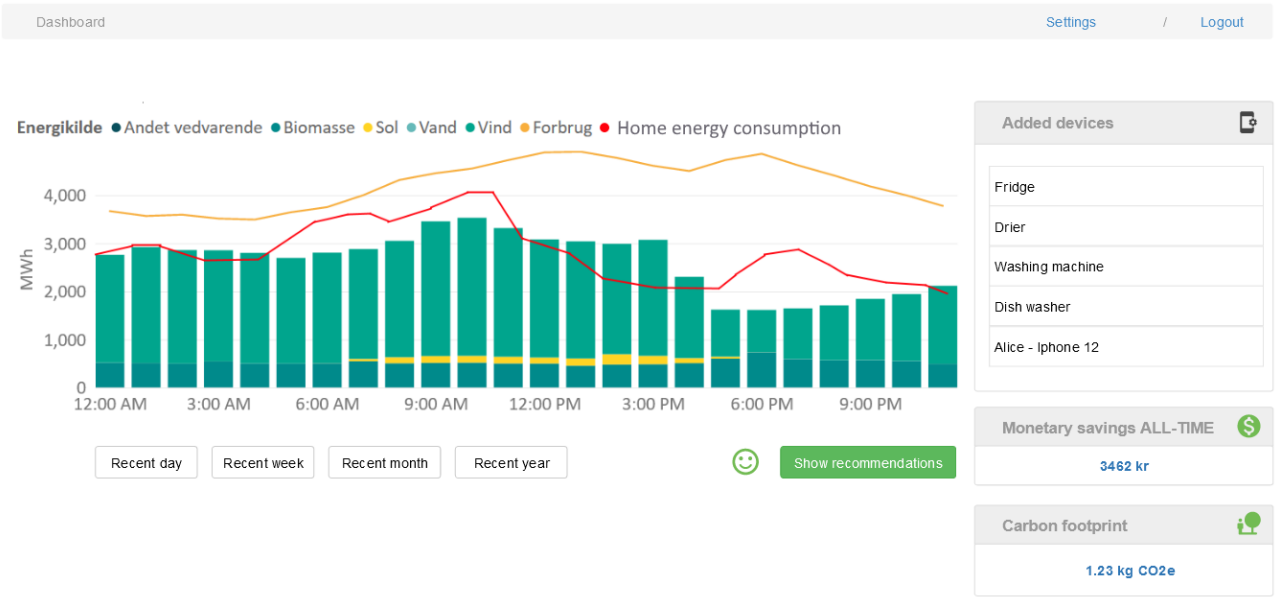


Figure 31: Mockup Dashboard page

Our draft shows a navigation bar at the top that allows the user to navigate to our different pages. On the dashboard the current energy production is displayed along with what the user's own energy consumption looks like. We also take into consideration how we want our recommendations to look like. Here it can be seen as a little smiley indicating how well the user is following the production curve. It also includes a button to show recommendations, to get a more detailed description on what the user might do to either follow the production curve, or when to shift their active devices to a time when renewable energy production is higher. We also want the user to have an idea of how much CO_2 the user is actually saving the planet for by following our recommendations. This also leads to users having a value that could be compared to other users, and they could get a sense of accomplishment, which could encourage continued usage of the application. We also include two boxes to show how much the user has saved in cost and in carbon emissions.

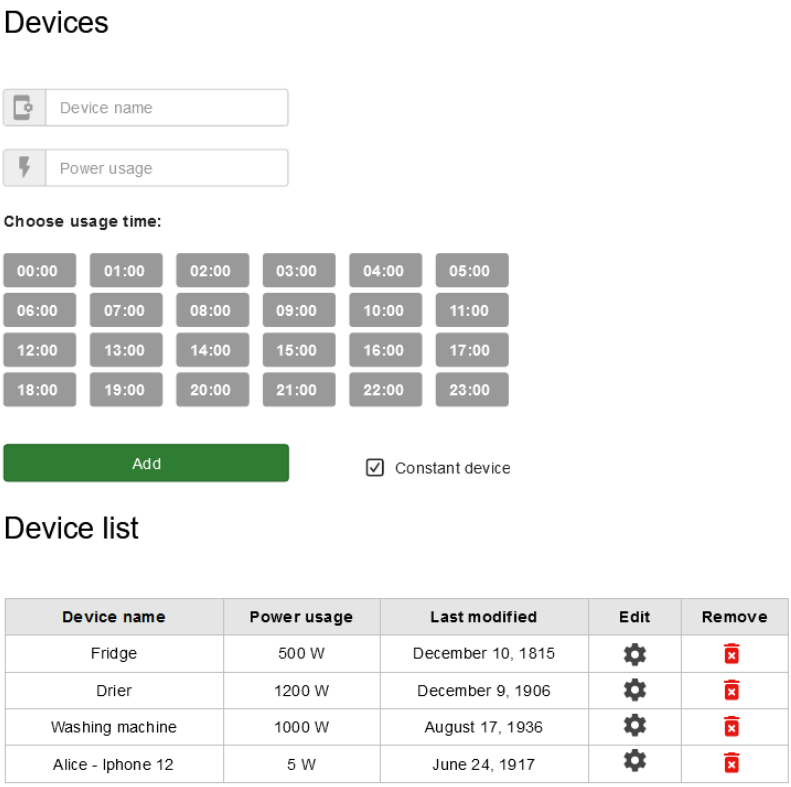


Figure 32: Mockup Settings page

The settings page mockup on Figure 32 consists of a form where the user has to enter the name, wattage and usage times to add a device. The usage time consists of 24 boxes representing every hour of the day, this take is inspired by the way one books meetings or schedule events. The idea is that the user clicks on the time where they "*normally*" have this specific device on. We say normally because it is not guaranteed that the device is on during the whole hour that is ticked off, due to a probabilistic function (See Section 8.5 for more detail on simulating use-time). We also included a tick-box to allow the device to constantly be on. This includes devices such as a refrigerator or freezer that is more or less always active during the day.

7.5 Data driven user advices

In order to meet our requirement of giving users advice on their energy use, we need to devise a solution that is based on the data we collect from energinet and the users' devices. Our initial idea includes graphs of the data, which contain a wealth of information, but leaves the interpretation up to the user. After the implementation of this feature, we feel this solution is too simple and does not adequately meet the advice requirement. This instigates a need to establish a more direct way to provide our users with advice. For this purpose we are inspired by the mobile application Spiir, whose approach implements advice cards, which gives Spiir users text based advice about their spending habits that are easy to take in, as shown in Figure 33.

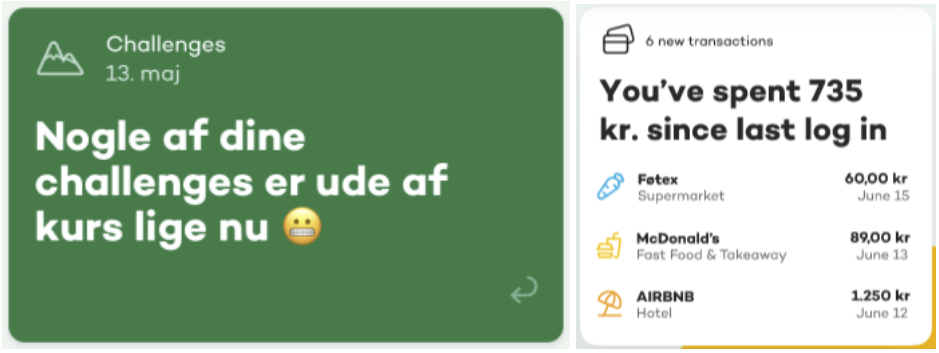


Figure 33: Spiir advice cards [66]

Spiir is not the only brand that use cards to present information, large brands such as Pinterest and Twitter also take this approach to organise the way their content is displayed [67]. Furthermore, the use of cards is such a well established practice in web development, that most of the larger CSS frameworks such as Bootstrap [68], Material [69], Bulma [70] etc. have made cards a part of their components library.

7.5.1 Why we choose Cards

Cards present a particular bit of information in its own little package. This makes the information stand out and helps the user find the information they are looking for, improving the scannability of a webpage and making the content easier to digest [67].

The point of choosing cards as the delivery method of our advice to users is to improve the usability of our web application. According to Matera et al. p. 146 [71], several definitions of the term usability exist. They proceed to describe the most commonly used definition of usability as consisting of the following five elements:

1. Learnability: the ease of learning the functionality and behaviour of the system.
2. Efficiency: the level of attainable productivity, once the user has learned the system.
3. Memorability: the ease of remembering the system functionality, so that the casual user can return to the system after a period of non-use, without needing to learn again how to use it.
4. Few errors: the capability of the system to feature a low error rate, to support users making few errors during the use of the system, and, in case they make errors, to help them recover easily.
5. Users' satisfaction: the measure in which the user finds the system pleasant to use.

Matera et al. p. 147 [71] continues to refine these definitions (of which we show 1. and 5.) in the following way:

1. Web application learnability must be interpreted as the ease for Web users to understand the contents and services made available through the application, and how to look for specific information using the available links for hypertext browsing. Learnability also means that each page in the hypertext front-end should be composed in a way, such that its contents are easy to understand and navigational mechanisms are easy to identify.

...

5. Users' satisfaction refers to the situation in which users feel they are in control with respect to the hypertext, since they comprehend the available content and navigational commands.

While, in future, we would be interested in applying all 5 items to our software design phase, only items 1 and 5 are the aspects of usability we intend our cards to address.

An additional concern we had, is that while our group is comfortable with reading graphs, the same may not be true for the intended users of our application. In an article on towardsdatascience.com [72], Eli Holder references an OECD study [73] that, among other things, examines adults' ability to interpret graphs. Their takeaway from examining the study is that 3 out of 10 American adults have difficulties interpreting graphs of even low complexity, while 6 out of 10 have issues with graphs of medium complexity, such as identifying trends on a line graph [72].

Since our graphs contain such rising and falling trends in CO_2 and energy production, they fall into this medium complexity category, and interpreting them requires that the user is comfortable with identifying these trends, if they are to successfully find the "best" times to use electricity. Although Denmark scored better in the aforementioned study, as can be seen on Figure 34, our estimate is that we would still risk alienating roughly half of our potential users if we only presented our data as graphs.

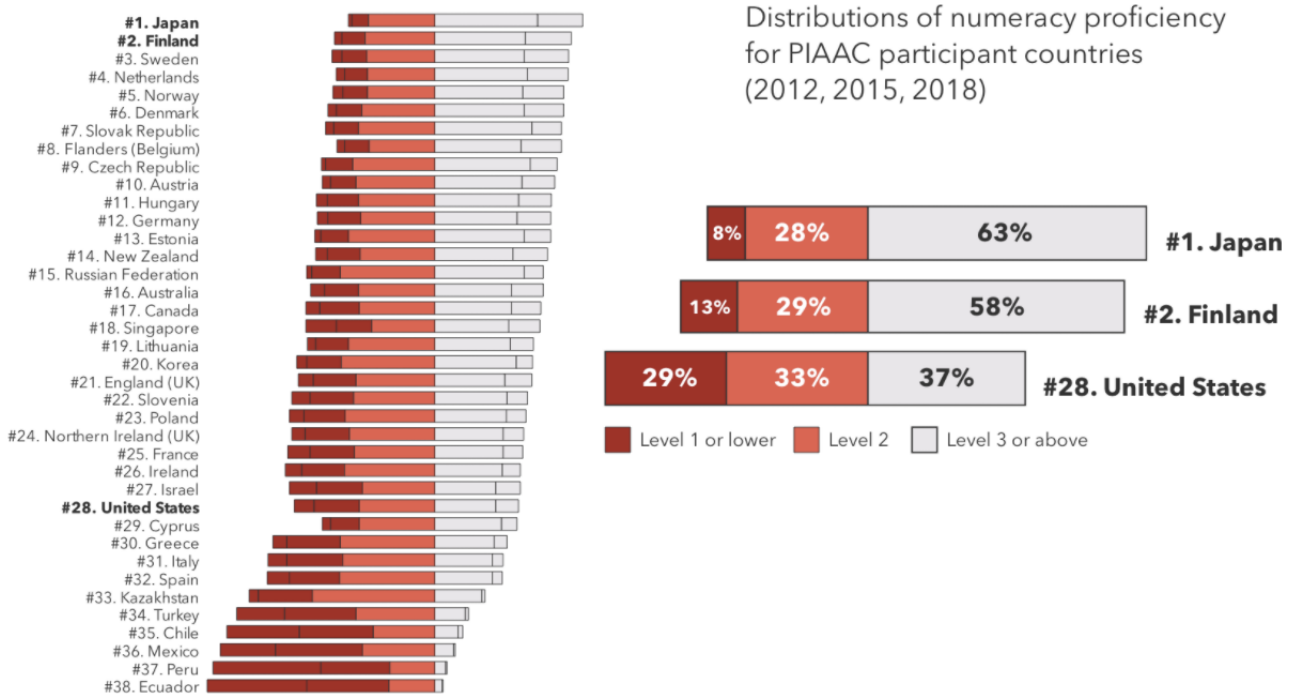


Figure 34: Numeracy proficiency [72].

In light of this, we believe Cards is a good design choice addressing both scannability and usability concerns. The graphs are still there, of course, for the users who prefer to do their own "manual" data processing by visually interpreting the graphs.

7.5.2 Designing the advice cards

To enable users from different backgrounds and levels of education to comprehend and benefit from the data we collect and present in our application, we want to give the users some actionable advice and recommendations

that are easily understandable. To properly design this advice/recommendation system, we follow some of the best practices for webdesign, we found some guidelines from a well-established web bureau called Tiller Digital [74], a company that produces copywriting and UI design services for multiple different large-cap companies, including Shell Oil.

One of the 12 practices for good web design is to have a clean design, and one way to do this is to implement a visual hierarchy for the advice cards. According to author Brandon Nickerson, who is Chief Design Officer at Tiller Digital, this means having a clear arrangement of composite elements in the design, and an order of importance structure to the element [75]. With that in mind, we set these requirements for the advice design:

- An Advice must be confined to a single parent element.
- Each type of advice must have its own visual identifier, which allows the user to easily "scan" the application for relevant advice and engage users.
- A card must contain a descriptive header which supports the user in quickly identifying relevant advice.
- Most importantly card must contain a main text area with the actual advice. This text supports the header and provides more detail on why the advice is given.
- Cards must have an area where buttons and call to actions (CTA) can be placed below the text area.

With these design principles in mind, we came up with our own framework for building pieces of advice in our application.

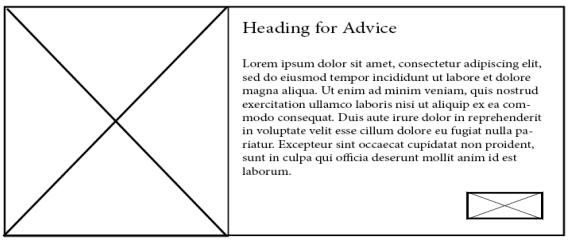
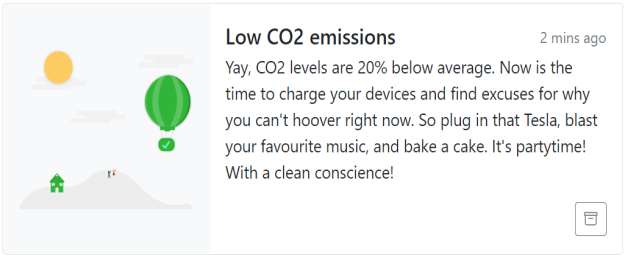


Figure 35: Early wireframe for advice component



Example of final design for an advice card

All of the requirements for the design is implemented in the final design of the advice card, in addition to the extra information about how long ago the advice was added to the user's feed, as seen on the right on Figure 35 above. And in the area where we make room for one or more CTAs, we add a button, so the user can remove the advice from his/her feed after they have read it.
One other design principle for good web design is Storytelling [75].

> "The very best stories deliver emotional impact. That's one of the primary ways they break down barriers to engagement and understanding. That's why you can connect more effectively with customers by leading with storytelling rather than with facts." - Brandon Nickerson [75].

In the design of the advice cards, we implement storytelling, because our earlier survey indicates that users, who are willing to move or postpone their energy consumption, are more motivated by reducing their carbon emissions rather than saving money. With this in mind, we phrase the text in our advice with a more emotional/ethical approach rather than showing only numbers and statistics.

Based on the data we collect and process, these cards provide the user with clear advice in plain language regarding which actions they can take, without the user having to process the data themselves. The implementation of these Advice Cards is described in Section 8.6.

7.6 Interaction diagram

We create an interaction diagram to provide an overview of where the user might navigate when using the application and what they can reach from different pages. The structure of the interaction diagram is chosen because it can easily be expanded upon and made into a navigation bar for implementation. As can be seen on Figure 36 below, the black dot at the top represents where the user begins. The arrow lines indicate a certain page direction and lines without arrows indicate bi-directional flow. It is possible for the user to reach any of the four pages with a single click after authenticating, but the user lands at the dashboard on login. What happens on the actual pages is cut down to a minimum, but can be found in the sequence diagrams from earlier in section 7.3 and in the following Section 8, regarding the implementation.

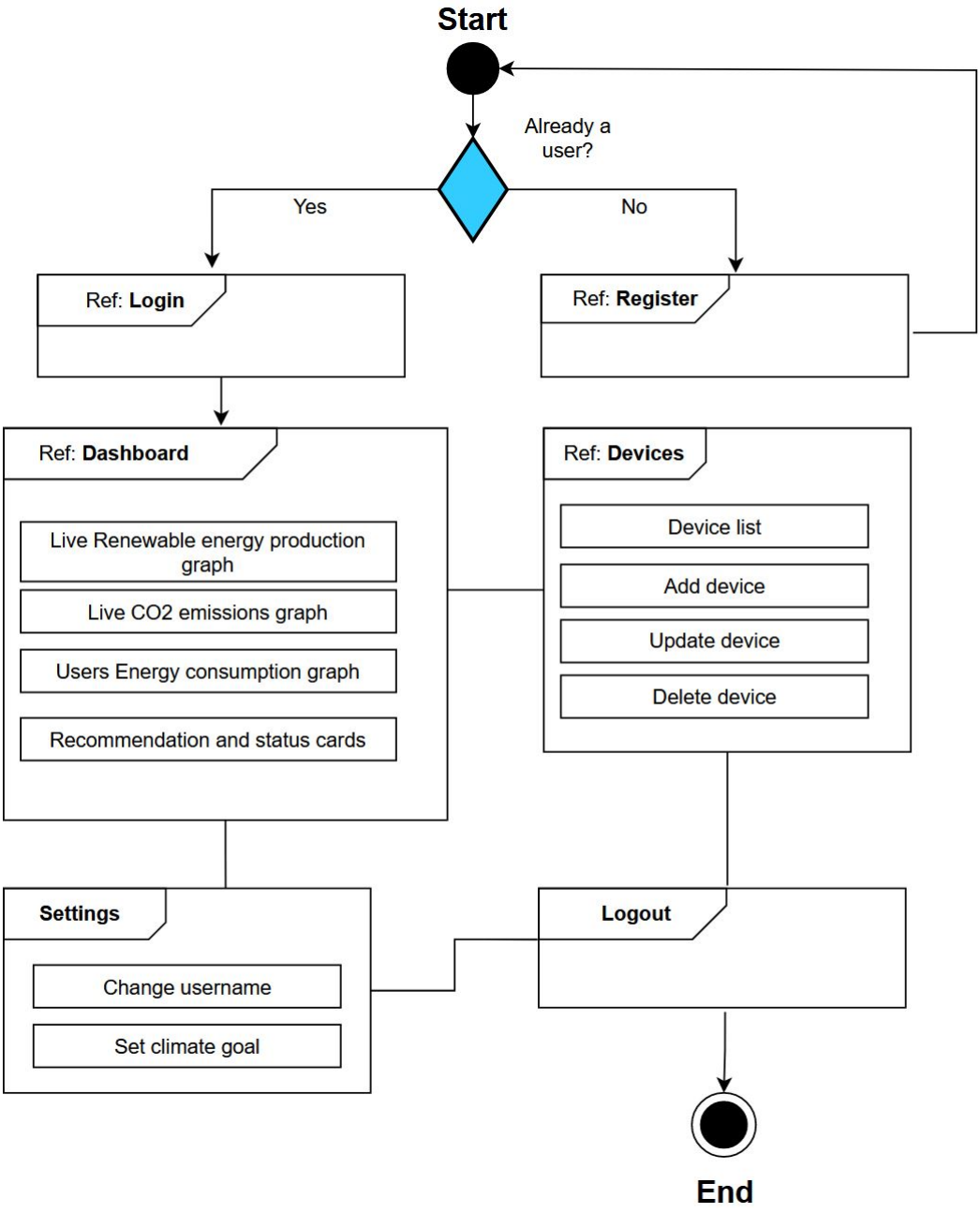


Figure 36: Interaction diagram for page navigation

7.7 System design conclusion

To recap, we took the requirements that lie in the minimum viable product (MVP) spectrum into account, in order to design an application that is an informational platform to help users, helping them make guided decisions on when to switch smart and non-smart devices' use-times away from peak CO_2 emission hours. The users should be notified of the current production and emissions data to make their own interpretations of what time of day to shift devices use-time to. As made apparent from our user-story, having users track the use-time of their own appliances allows for a home-energy monitoring system, that can be made to give recommendations on specific device use-times along with a perspective into the users' own energy peaks during the day. The requirements also specify a need for a user-base where different users have profiles to contain user specific information on their recommendations etc, which we try to implement. We have given a description

of the system to give a structural overview, and covered diagrams of the different components. We follow the waterfall model and go into the implementation phase, implementing the components and features from this design section.

8 Implementation

This section documents the implementation of our design requirements and the considerations we made along the way. Our application has the following structure, in order to make it easy to maintain and develop:

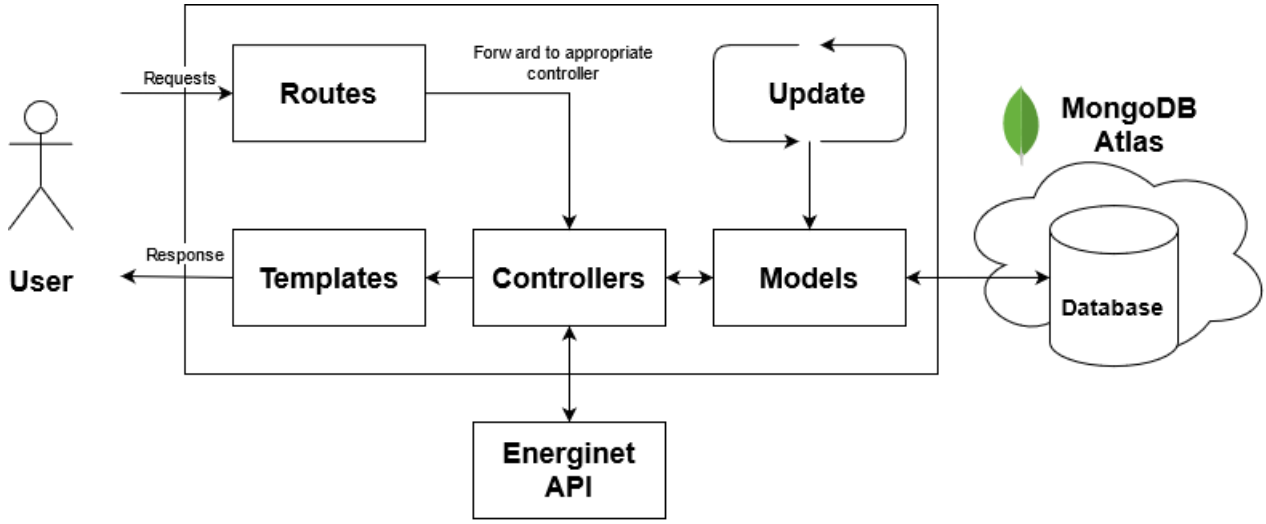


Figure 37: Model for organising the code

This model is called a M-V-C (Model-View-Controller) which gives a nice high level overview of our application. Therefore our implementation of our application will follow this structure.

8.1 Collecting graph data

The data collection is done through an asynchronous function since the entire webpage must not wait for the data to be collected and shown, as this can take a bit of time. The function for CO_2 emissions tries to fetch a month's worth of data, equivalent to an array of length 17280, each entry representing a 5 minute interval. Once the CO_2 data is fetched from the API provided by energinet.dk [76], the data is then consolidated into a graph where each time point is the average of n lagged days for the same time point to compare monthly averages with daily averages. This is done through the function: "createAverageGraph(days, data)", where days is the number of days worth of data to use, and data is the output. This process can be seen as a sequence diagram on Figure 38 below:

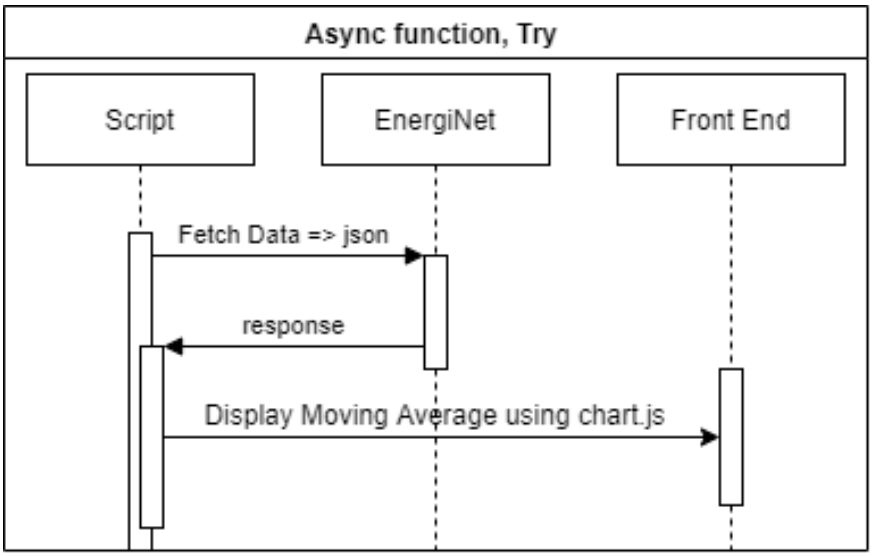


Figure 38: Displaying a graph to the user using chart.js

The average is calculated for each time point throughout the day, and the data will be displayed as a graph through the use of a package called chart.js:

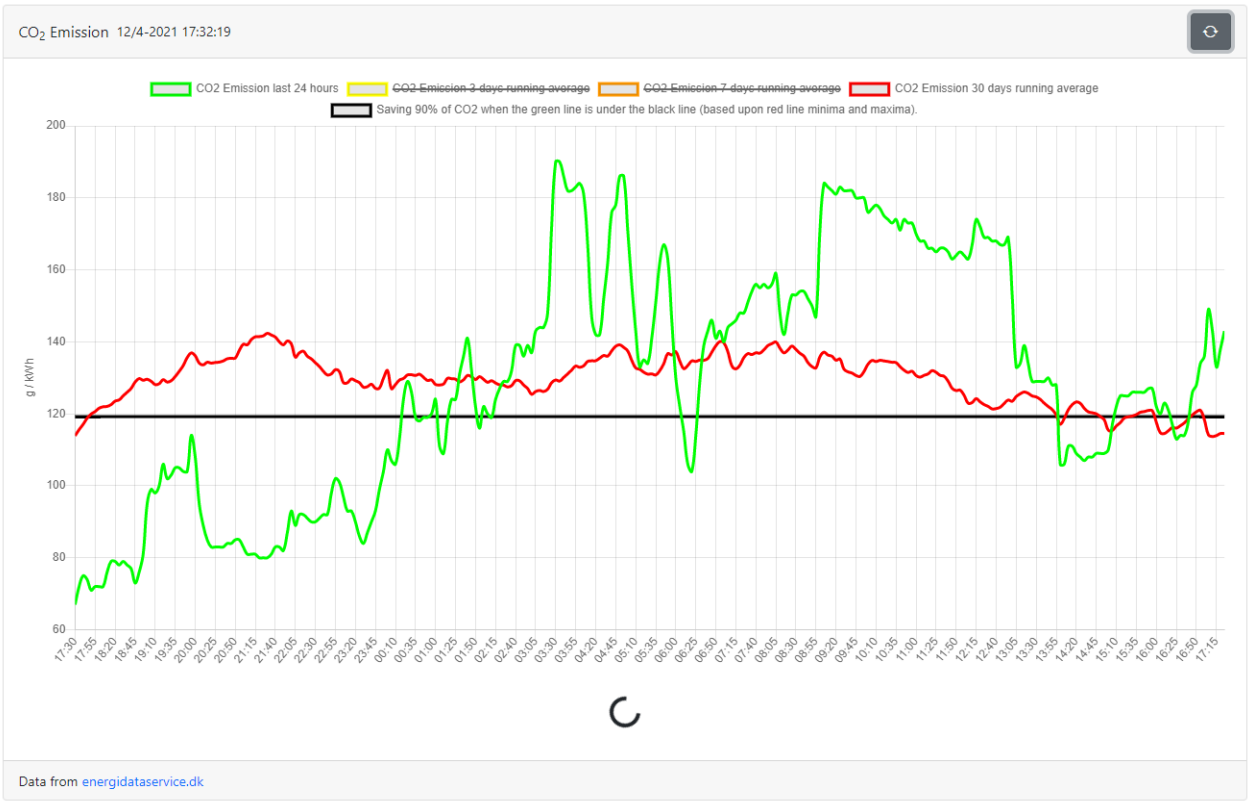


Figure 39: x-axis is the past 24 hours, green is carbon emissions per kWh in the past 24 hours, and red is the monthly average carbon emissions per kWh.

The black line is made to sort all the data points from the 30 days average graph. Then if you wish to save 50%, you would take the middle of the sorted array. If you wish to save 75%, you will take the value 3/4's through the array. If you wish to save 100%, you will take the last value in the array, indicating the largest value the

graph has. The black line then indicates how much carbon emission you can, as a minimum, save if you only use electricity when the green line is under it (today's emissions), compared with the 30-day average. Therefore it is possible to save 100% if at the current time it is much more environmentally friendly to use electricity than it has been throughout the past 30 days.

Beneath the graph, there is a loading icon. This loading icon appears while data is being fetched and calculated if the user chooses to refresh the graph.

In order to improve the user experience in the application, we have reduced the time, where the user is stuck waiting for all data computations to finish before loading the DOM-elements. To achieve this we moved all of the client-side computations onto the server, and made separate internal routes with the form /data/name to fetch the graph data from. Doing this the user can load the page before the computations are finished and then fetch the data asynchronously.
We have 5 graphs in total, but they all rely on the same methods for fetching and displaying data as those mentioned above. What is interesting is how the data is calculated, and that is therefore what we shall delve into with the other graphs.

8.2 Forecasting data

One of the more interesting data sets is the one making up the forecasts graph: In order to know which days are better for the environment when using electrical appliances, you need to be able to see if tomorrow is a bad day when it comes to carbon emissions per kWh than today.
To estimate this, we need to be able to, in some capacity, forecast future emissions.
There are many ways to forecast the future, but perhaps the easiest way is to forecast the future based on past events. In order to do this, we looked at how the ARMA (Auto Regressive Moving Average) and its family of time series models work.

ARMA models are composed of two components: The AR part, meaning Auto-Regressive, and the MA part, meaning the Moving Average. These models come in many variations, and as such, one model may work differently from another, but they work on the same basic principles:

AR(p):
An auto regressive process is based on the earlier lagged data of order p, forecasts what the next point should be. The AR model can be defined as such: $Y_t = \alpha + \beta_1 Y_{t-1} + ... + \beta_q Y_{t-q}$, where α is the intercept term (the mean of the model). β is some calculates weight, and Y is the lagged result [77].

MA(q):
A moving average process is based on the error of lagged data of order q, calculates what the next data point should be. The MA model can be defined as such: $Y_t = \varepsilon_t + \vartheta_1 \varepsilon_{t-1} + ... + \vartheta_q \varepsilon_{t-q}$ Where ε_1 is some constant, which is, on average, just the mean. $\varepsilon_{t-1..t-q}$ is the errors from the last data point, which is somewhere between the mean and the standard deviation, and $\vartheta_{1..q}$ is the errors weight/coefficient between -1 and 1 [77].
To illustrate how this works, let us assume the error weight/coefficient is a constant equal to 0.5, and the mean is 10. We have a crazy professor who wants cupcakes in our example, but you never know how many you need to bring. He will tell you how wrong you were each time, and therefore you can adjust how many you bring next time.

T	$\hat{Y}_t$	ε_t	Y_t
1	10	-2	8
2	9	1	10
3	10.5	0	10.5
4	10	2	12

Where $\hat{Y}_t$ is your guess, ε_t is the error from the guess and Y_t was the correct amount. Here the model only considers the last period (order), as you only try to adjust based on the error from last time. This is called a process MA(1). But the same process could continue indefinitely backward to forecast the next time point.

ARMA(p,q)
The ARMA model, is simply defined as such: $ARMA(p,q) = AR(p) + MA(q)$

Other variations of this model exist, such as ARIMA, which stands for Auto-Regressive Integrated Moving Average. The reason to integrate it is to make a process stationary, as it is hard to predict future values using ARMA if the time series has an upward or downward going tendency, such as a stock market. A stationary process is a process whose joint probability distribution (sum of 2 probability distributions, in this case, X and Y) doesn't change over time. Consequently, we can observe that is mean, and variance does not change over time [78].
Another variation is SARIMA, which stands for Seasonal Auto Regressive Moving Average. Let us consider a company that sells Christmas trees for instance. These trees are only really sold at Christmas time, while none are sold during the rest of the year. In this example, it doesn't suffice to simply forecast the future based on the months before, we have to take into account the seasonality of a time series.
These models can be used to predict stock markets or other time series. But it is not a perfect prediction if the system is chaotic or inhabits white noise, which many real-life time series do. Therefore, it is more of a rough estimate of the general direction, more often than not. This means that for every hour into the future you try to forecast, the more wrong the forecast may become.

White Noise: White noise is any noise in the graph that is unpredictable, or inherently random. If something is indeed predictable, then it is no longer white noise, since you could compensate for the white noise by adding a constant at the end of the model:
For white noise, the following apply:

1. A stochastic variable is uncorrelated with its previous variable: Cov$(\varepsilon_t, \varepsilon_{t+k}) = 0$.
2. A time series is 100% white noise if the mean is zero.
3. The variance is constant [79].

Chaos: A chaotic system is a system that, even though it uses the same initial input, the output varies more and more as time goes on. Chaos can be concisely defined as such:

1. The series is sensitive to initial conditions. This means that adjusting the initial condition will change the entire system.
2. It must be topologically transitive. This vaguely means that a set of points that are close to each other eventually get flung out to big sets that aren't necessarily close to each other [80].
3. It must have a dense periodic orbit. Meaning, even though not all points are a part of a periodic orbit, it is impossible to distinguish it from the points that are [81] [82].

In order to know which model to use, we have to check whether or not our data is stationary:

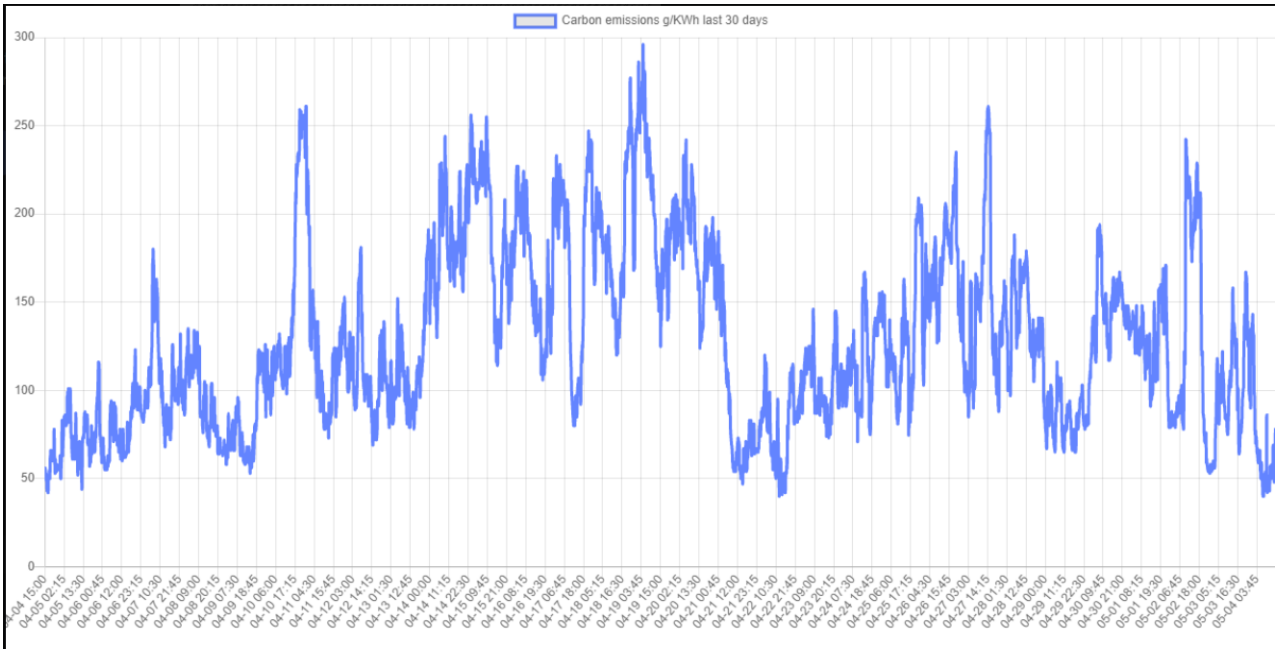


Figure 40: Carbon emissions throughout 30 days

As we can see from Figure 40, the process does seem to be stationary, even if it is only weak stationarity. Therefore, our model does not have to be integrated.
We already know from the analysis that at certain recurring times throughout the day, each kWh emits more carbon. Meaning that there exists some form of seasonality, if each time period is one day.
Therefore, our model has to be some sort of a seasonal ARMA model. The model we implement is not using the naive definitions of the ARMA model, but in practice the model uses the same principles to forecast data. The reason we choose not to directly copy the ARMA model, is that it gives us a incorrect and imprecise results. Due to time constraints, the algorithm is implemented as well as it can, while trying to fit it to produce the best possible predictions.
Though while trying to fit it, we were aware that overfitting might be an issue: Meaning that the forecasted data is too tuned to the lagged data, and therefore is bad at predicting large unexpected trends.

In order to account for seasonality, the data set is adjusted to a 2 dimensional datastructure, that not only has a time axis, but an additional date axis:

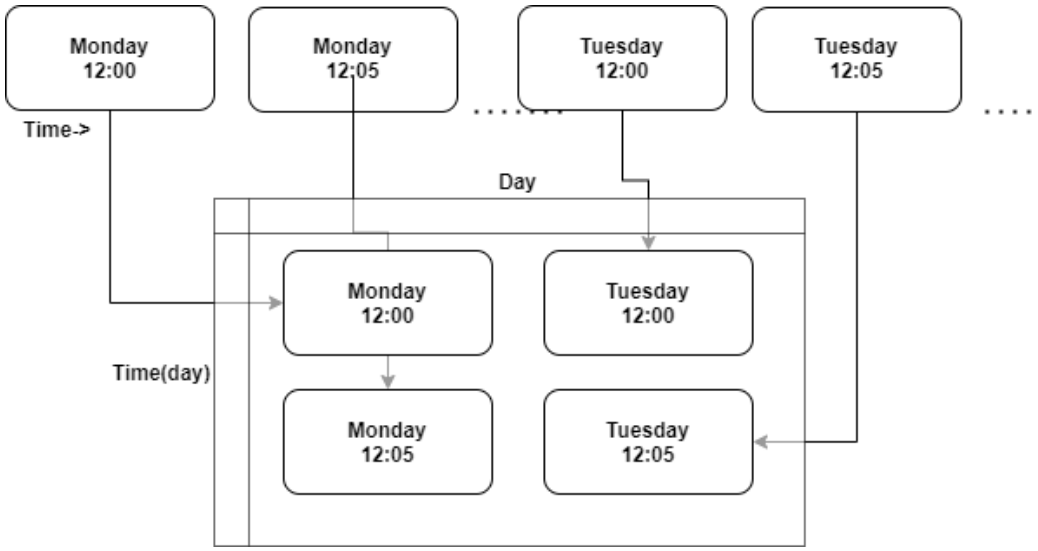


Figure 41: 2D datastructure to remove seasonality

This is to make sure that each data point is only forecasted based on the previous data points for the same time of day. Thus removing seasonality from the calculation while still displaying it in the finished graph.

Next, the forecast is calculated for each time throughout the day, throughout N days into the future. The structure of the forecasting algorithm that is used for each individual data point that we created has the MA part as the primary component, while using the AR part as a bound for how high or low a data point can be, such that it never steers too far from the previous data point. The weight function was calculated by finding the difference between the last datapoint and the new one. The larger the difference, the lower the weight of the error term on the forecasting function. Other bounds that we have implemented is a zero bound, so that there is nothing better than net zero carbon emissions. Furthermore, the entire forecasted graph needs to stay within standard deviation of the previous data points. Since the forecasting function only semi implemented the AR model, the entire forecasting function only uses one term for how far back the order is, instead of both p and q.

In order to test the accuracy of our graph, we plotted data from previous time points, while knowing what the result should be. Since our forecasting algorithm forecasts one day based on earlier days, it only really ever forecasted one day ahead, even though the result compares six days. This is the case, since everytime one day passes, we feed it the previous date's real data. When forecasting blindly into the future, only the first day can expect to have this accuracy, while the rest is going to quickly become random. We can then plot these two graphs against each other to check the accuracy. In order to test if the two graphs fit, we found the difference between each forecasted data point and each real data point, and averaged the error. The closer to zero, the more accurate our model is.

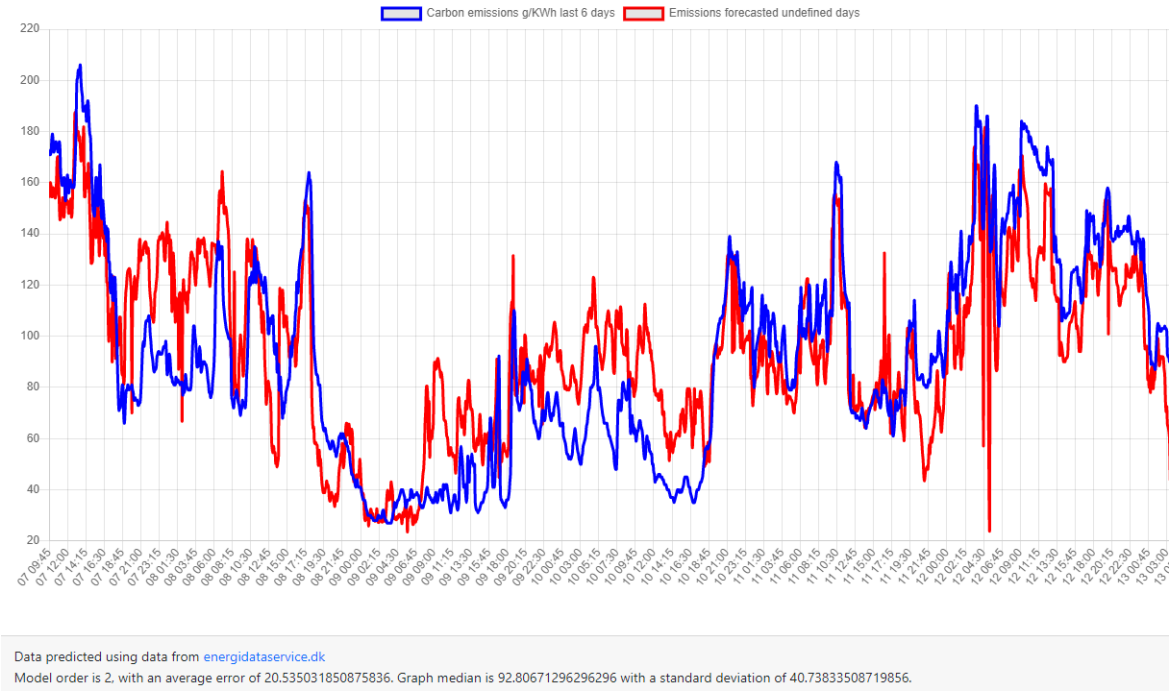


Figure 42: Average error: 20.., Median: 92.., STD: 40..

As we can see from Figure 42, the forecasted data for each day forecasted has an average error of 20 from the true data set, at this time. The more chaotic the behavior of the true data set becomes, the larger the error becomes. Having an error of 20 beats the STD of 40, making it a better solution than randomly plotting data using a Gaussian distribution, but not that much greater.

8.3 Database structure

When choosing a database for our application, we wanted to have a database where the learning curve is as entry-level as possible for newcomers. There were several databases in consideration when we chose, some of them include MySQL, PostgressSQL and MongoDB. Ultimately we choose to work with a MongoDB for a number of reasons:

1. We have worked with MongoDB in this semester and therefore know how to work with it.
2. MongoDB has good documentation, as it is one of the more widespread databases currently in use. This allows for easy lookup and code review if we get stuck.
3. It is easy to understand when you have knowledge on JavaScript objects and JSON formatted data.
4. We can intuitively create Object Data Models that sit on top of the database, making it easy to make CRUD[1] operations and validation.
5. With MongoDB Atlas cloud database, we can all work on code locally while sharing a database.

To help us interact with our MongoDB we have chosen to use mongoose [83] as an Object Data Modelling tool. Mongoose is well known and also well documented and makes it easy for us to think about and create data models. Using an ODM (Object Document mapping) is also a good practice as it allows us to think of our data in terms of JavaScript objects instead of MongoDB's native query language.

[1] Short for: Create, Remove, Update and Delete

The structure of our database is based on what data we want to store. We want to represent multiple devices, so it makes sense to make a model for this, which contains information on a device. A device must at least contain a name, maximum energy usage and a storage container to know when the device is be turned on.
We also want a way to store information on user specific things such as their devices, settings, name and password.

With the models in mind, we also consider what relations the models have to one another. The devices need to be associated with a user profile, such that a user profile contains a collection of devices, and a device has a relation to a single user profile. To separate the user's credentials from their user profile, we make a relation between a user profile and a user by giving a user profile a reference to its related user.

To provide an overview of our three models and their relationships we use a UML diagram. The diagram on Figure 43 below shows the fields that is contained within each model and the links represent the relation between them. The number above the relation line describes the relationship in terms of one-to-one or one-to-many.

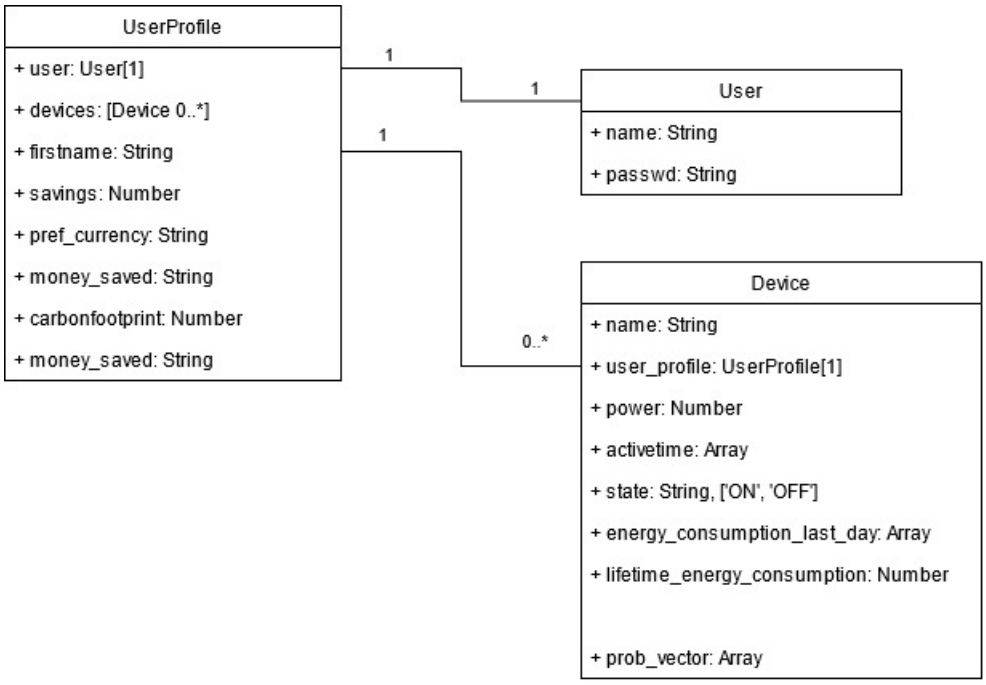


Figure 43: UML Document model

The three models are implemented using mongoose's *Schemas* that fit nicely with the above diagram, as the models can be written as JavaScript objects containing field-type pairs, with constraints on how the data should be stored. Putting constraints on the data-type allows mongoose to catch and stop data with wrong input format from being inserted into our database. This makes validation on user-input easier to handle when we get to create forms.

The relationships between models is done through record-id's that uniquely identify each record with an id string. The id key is made automatically by MongoDB when creating a record and is used every time we query on a specific record.

8.4 Login and registration system

The first thing the user experiences on our project application is the login page. Since our project application monitors devices and appliances tied to a specific owner, user authentication was a must have for the application structure, so that a user cannot access another user's devices. But to authenticate a user, we firstly need to have a user to authenticate. Therefore, we have implemented both user and user profile models.

8.4.1 User model

In order to minimize the amount of data needed in the process of authenticating the users, we chose to separate the user model and user profile model. Therefore, the user model only includes the information required to confirm that the user is registered.

```
1 // User Model
2 let userSchema = new Schema({
3     email: {type: String, required: true},
4     password: {type: String, required: true}
5 });
```

Listing 1: User Model

The user model contains two fields, the first being the user's email. For this project, we opted to use email for authentication instead of using a username. We made this decision is because, in our opinion, email is a more modern approach and secondly, emails are inherently unique, which makes the registration process smoother. The second entry in the user model is the password field, which is stored in the database as a plain text string. But from the screenshot on Figure 44, taken from the database, it is clearly shown that the stored password is remarkably obscure.

```
_id: ObjectId("608911a0070ea135f09e3705")
email: "fake@email.com"
password: "$2b$10$XMdzMTfTtUbsTS8KDwI1juaw7z/hWWHITfHNSpK258ZwXfGmyhexi"
__v: 0
```

Figure 44: Example of user document in our database

That is because we only store hashed version of the passwords in the project database.

8.4.2 Hashing - A brief explanation

Why do we encrypt passwords and store the hashed passwords, and what is a hash? Let's say a nefarious individual gets unauthorized access to our database, and we have not encrypted the users' passwords. This results in all of our users' authentication information being readable in plain text, and therefore easily abusable by intruders. Since we have implemented password encryption, the intruder who gained unauthorized access to our database would be left with only the emails, and a bunch of hashes, which can neither be used to login into our application or link a password to an email address.
To implement password encryption in the application, we used the node package **bcrypt** [84], which is a very popular library used for encryption in node.js applications. This library is based on a block cipher named blowfish [85]. In this project we will not expand on the theory behind this, but rather focus on the usage of the **bcrypt** library.

The functions from the **bcrypt** library, which we use in this project are:

- **Bcrypt.hash()** - This function is used to generate a hash which is 60 characters long, and conforms to this pattern: $[algorithm]$[cost]$[salt][hash]
- **Bcrypt.compare()** - This function is used to compare the user's input to hashed version that is stored in the database. The function hashes the user input under the same conditions as described in the hash we test against, and returns **true** or **false**, depending on the outcome.

8.4.3 User profile model

With the user models briefly explained in Section 8.4.1, we will in this section, cover the user profile model, which is associated with the user model. The user profile schema is meant to contain all information that is typically associated with a user but is unused in the authentication process. As can be seen in the user profile model in Listing 2, the user profile contains all settings and information about the user. The field 'user' contains a reference to the user document, that is associated with the specific profile.

```
1  // User Profile Model
2  let user profileSchema = new Schema({
3    user: { type: Schema.Types.ObjectId, ref: "User", required: true },
4    devices: [{type: Schema.Types.ObjectId, ref: "Device"}],
5    firstname: { type: String },
6    lastname: { type: String },
7    pref_currency: { type: String },
8    money_saved: { type: Number },
9    carbon_saved: { type: Number },
10   carbon_footprint: { type: Number },
11   sustainable_goals: { type: Number },
12   total_energy_consumption_last_day: { type: Array, default: new Array(288).fill(0) }
13 });
```

Listing 2: User Profile Model

Having a user profile that contains this type of information, enables us to only query the database for the smaller documents with authentication information when the user is trying to log in, and then the rest of the information, once the user is logged in and the information is needed.

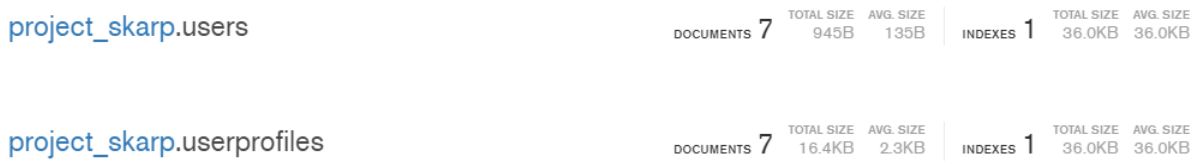


Figure 45: Size difference between user and user profile documents

As shown on the two screenshots in Figure 45 taken from our database, the average size of a user document is 135B, and the average size on a user profile document is 2.3KB, which shows that splitting our user and user profile, drastically lowers the size of the query response on the login page.
We do not know if it makes a real difference in the grander scheme of things. Even though we do the authentication check on every page, the amount of data required to do so may be insignificant compared to the rest of the data required to display the page. However, the user profile could grow to contain more information and in terms of scalability, we believe that keeping data load minimal is a good practice.

8.4.4 Registration Controller

Now that we have established the underlying models used in both the registration and login controllers, we can look at the process of registering as a user in our application. The registration controller is the function that manages the server side process of registering as a user, illustrated in Figure 46:

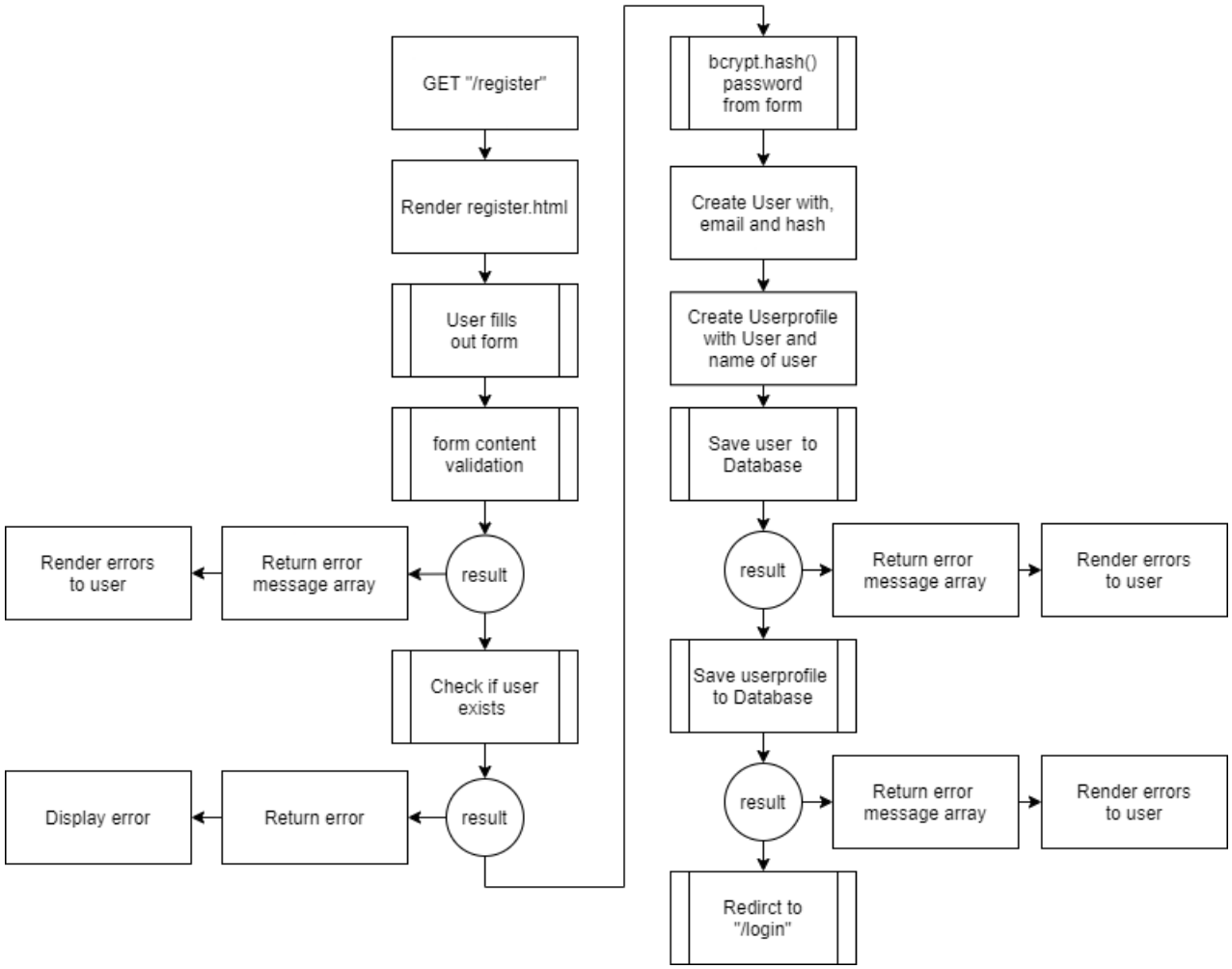


Figure 46: The registration process

As seen in the flowchart, the user sends a http GET request to the server when visiting our registration route "/register", where the user is met with a registration form. Once the user has filled in the form, the information is sent to the server with a POST request. Once the server receives the information, the supplied information is then validated via the library express-validator. We have the following validations and normalizations set up:
email

- isEmail() - validating that the content of the field is an email
- normalizeEmail() - normalizing the email, according to the several "rules" for email formatting, such as lowercasing, removing dots in gmail, etc. For further details see validator.js reference [86].

Password

- trim() - trimming or sanitizing the password, by removing whitespace on both left and right side of the first and last non whitespace character. Used for preventing the user from signing up with a password with unintended spaces before and after password.

- isStrongPassword() - validating that the user's password is categorized as a strong password. In the project we use the default parameters, which are: { *minLength: 8, minLowercase: 1, minUppercase: 1, minNumbers: 1, minSymbols: 1, ...* } [86].

Password Confirmation: To confirm that the user's password and password confirmation matches, we have implemented a custom validation method. The method checks whether the value from the input field "password confirm" contains the same content as the value in "password". This method either returns true or an Error object with a feedback message, which is then added to the array validation results.

Now that we validated the contents of the user's request, we check if the validation process has yielded any errors. If there are no errors, the controller starts the registration process by first checking if the users email is already in database. If the email is not in the database, the application advances to next step, which is encrypting the user's password via the bcrypt, as such:

```
bcrypt.hash(req.body.password, saltRounds, function (err, hash) {
    ...
}
```

Listing 3: hashing with bcrypt function

We use the function *bcrypt.hash()*, explained earlier in Section 8.4.2, which takes the user password, and either returns an error or a hash in the callback function. This means that the hash is available to the application in the callback function. Once the user's password has been hashed, the application proceeds to create both the user and user profile documents by using the predefined models for both as can be seen on Listing 4 below:

```
let new_user = new User({
    email: req.body.email,
    password: hash,
});

    let user_profile = new user profile({
    user: new_user,
    firstname: req.body.firstname,
    lastname: req.body.lastname,
    money_saved: 0,
    carbon_saved: 0,
    carbon_footprint: 0,
    sustainable_goals: 0,
});
```

Listing 4: The creation of user and user profile

After the two objects have been created, the next step for the application is to save the documents to our MongoDB database. To do that, we use the mongoose function *save()* shown on Listing 5

```
// Save new user
new_user.save(function (err) {
    if (err) {
        return next(err);
    }
    // If User save is successful save Profile
    user_profile.save(function (err) {
    if (err) {
        return next(err);
    }
    // If Profile save is successful redirect
```

```
12      res.redirect("/login");
13      });
14  });
```

Listing 5: Saving a user to our database

As the code shows, the save function either returns an error or a successfully inserted document. So we try to save the new user to the database and if that succeeds, we try to save the new user's profile to the database, and then redirect the user to the login page, if everything worked.

8.4.5 Login Controller

The process of authenticating a user, is almost the same process as registering a user. The user sends a GET request to the server when visiting the authentication route "/login". Once the user has filled the form with their credentials, it is sent back to the server with a POST request. Once the server receives the information, the supplied information is then validated and normalized in the same manner as registration from Section 8.4.4. **Email** has the exact same validation and normalization as in registration. For the **password**, we still use trim, for unwanted whitespaces, but we omit the use of *isStrongPassword*, as we do not need to check if the password is strong, because the stored password already fills that requirement.

Now, if the form's values have passed the validation without any errors, the next step in the process is to query the database for a user document, that has an email, which matches the email supplied in the form. If a user with that email is found in our user collection, the **bcrypt.compare** function is used to check that the password from the form matches the requested user's hash from the database.

Now that the user has provided both matching email and password, we make use of the functionality provided by the package "cookie-provider", to set a cookie with this content:

- name: Auth
- content: found_user.id
- maxAge: 900000
- httpOnly: true

The user's id from the database is stored in the cookies field: content, and with this cookie in place, the user is now authenticated. While the cookie persists, we can use the user's id from the cookie, to request the users information throughout the application.

Checking that the user is authenticated:
To verify that user is logged in/have a valid cookie, we implement check on Listing 6 below.

```
1  exports.isAuthenticated = function (req, res) {
2    if (req.cookies["auth"]) {
3      return true;
4    } else {
5      res.redirect("/login");
6    }
7  };
```

Listing 6: User authentication check

This authentication check is called on every route in our application, in order to check if the user's cookie is set, before rendering any page in our application.

8.5 Data simulation

On the device page, users have several options. Firstly they can see an overview of all the devices they have added. Secondly they can edit or delete an existing device. Lastly, they can choose to add devices. Adding a device requires the user to input the name, power usage, and active time for the specific device. The active time specifies when the added device is usually in use. The device added by the user is then sent to our database. The database stores each device with a unique id associated with, which allows the application to differentiate between devices. This also allows the device to maintain it's id, if the user decides to edit the power-on time, name or power consumption.

We simulate our device data because using physical devices requires setting up a device at home with contact to the server to allow energy use data to be transferred. Furthermore, as we concluded in Section 2.5.2, most of our respondents have somewhere between 0 and 2 smart devices in their homes. The effect of our solution has a direct correlation with the number of different devices that can be monitored. Although it would be quite interesting to get a device to communicate with our server, it is outside the scope of our project's learning goals, which focus on server/client communication. It would slow down the application development process in which we only needed static dummy data to make a functional prototype. Once we had that, we implemented a way to simulate device data. We describe this in the following segment.

To simulate incoming sensory data, our application generates sensory data on a timed basis to simulate a smart-plug sending information on the current device that it is monitoring to a server. We have chosen that the simulation collects sensory data every five minutes. This is somewhat arbitrary but fits nicely with the API energy production data, which also uses a five-minute interval. The thing we are after is to collect the data about a device's active time. This is because we can use the active time and its maximum rated power to calculate the power consumption of multiple devices that make up an entire house of devices. So essentially, knowing when all the devices turn on and off, we can get a clear picture of how much electricity a house is consuming, which helps with calculating the users' total CO_2 emissions.

We assume that the user knows roughly when their devices are on during the day. The user therefore has to enter their devices normal use-time. This is done by making 24 "hour-boxes" (one for each hour in the day), which the user can tick off. This is where it becomes a bit more complicated. As an example a user has checked off the hour-box 19:00, it is not a guarantee that the device is turned on exactly at 19:00. Rather there is a probability that the device might turn on at any given moment around that time. The closer in time to which the device is set to be active, the higher the probability is that it suddenly turns on. On the other hand, if a device is far away from the next time slot that is set to be normally on, then the probability is very low. Relating this to a real-world example, a user usually only brews an evening coffee around 8:00 PM. The probability of this user brewing coffee at 7:30 PM is more likely, compared to the user brewing coffee at 6:00 AM because it is further away in time. We have decided that a Gaussian distribution around the active times determines the probability of turning on. A graph of a single device probability of turning on can be seen below (Figure 47).

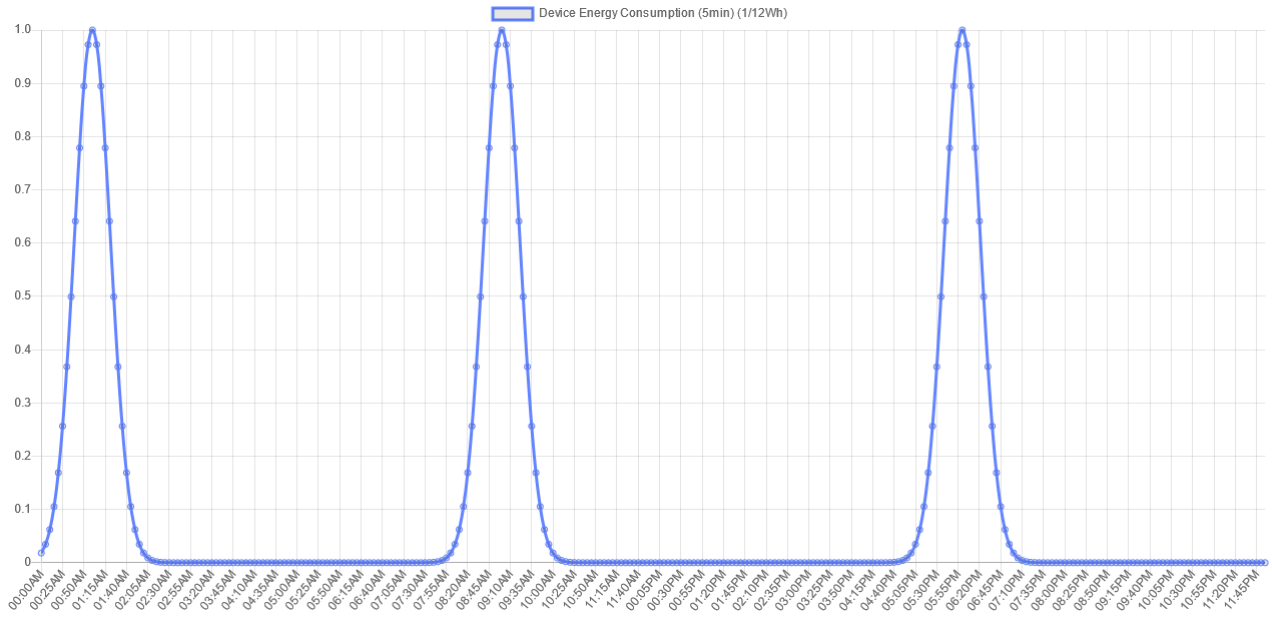


Figure 47: Probability distribution of a device having active-times at 1:00AM, 9:00AM and 6:00PM

Such a probability distribution is calculated for every device and helps in determining if a device is currently active or inactive at any current moment in time.

As the simulation has to run in real time, we want to have the simulation collect sensory data every five-minutes. To do this we start a timer by calling the built-in *setInterval()* function from JavaScript when the application starts, and make it run an update function every 5 minutes. To ensure that the update is run on the same intervals every time we start the application, we wrap the *setInterval()* in a *setTimeOut()* that waits until a five-minute mark is hit. As an example, if the local time is 9:41 we wait until 9:45 before initialising the interval. This is done to prevent overlapping the timed measurements to not get out of sync with the update loop.

The update loop lies as its separate component and is responsible for showing and calculating the live-feed data necessary to be shown on the home screen graphs. The update loop will first find every profile stored in our database and then go through every device from the profile. On every device, it grabs the probability of it activating in the current moment, using the probability distribution as described above. Having the probability, it then generates a pseudo-random[2] number between zero and one and compares with the probability. When a random number is below probability, the device turns on, and the state is updated. Otherwise, the device state is set to be off.

Apart from updating the state of the device, the device power value is appended to a list containing the device energy consumption the past day. This is to have a record on when the device got turned on during the day. When all the devices have been iterated over, the profiles' energy consumption gets updated by summing the power of each device that is turned on. This value will correspond to the total amount of energy the profile is using at the given time. To save the value, we append it to a list on the profile containing the energy consumption of the profile over the past day.

[2]A deterministic number if you have the seed of the generator function

8.6 Generating Advice Cards

As we can see, the first field in the advice card schema is a reference to the user profile that the advice is tied to. The second field in the schema is the class of the card, which determines which of the three classes the advice is. The field "class" is used to determine which type the advice belongs to and can take the form of status, event, or recommendation. ex. An advice card could be created with the class: event and have the grade: 1, which creates an advice card related to the current solar energy production. Title and message are the fields that contain the information we want to convey to the user. Finally, the field "created" gives the advice a timestamp, so the user can see how long it has been since this advice card has been created.
There are 2 different kinds of cards a user can get: A global event and a personal advice card. The general data structure for the two cards is practically the same. The biggest difference is that the event card is mirrored to all user profiles and is entirely dependent on the general data.

8.6.1 Event cards

Event-driven cards are the first type of advice cards, which we will explain. But what are the events that trigger the creation of these types of advice cards? In our application, we have defined an event as something that the user cannot affect but still will benefit from knowing when these events happen. In our application, we have set up a function that monitors for these type of events:

- Solar energy production - advice card is created when the current solar energy production exceeds the last week for average production in Mega-Watt/hour (MWh). The user knows that there is being produced more solar energy than usual.
- Wind energy production - an advice card for this event is created when the combined current offshore and onshore wind energy production exceeds the last weeks average production in mWh, so the user knows that there is being produced higher amounts of wind energy than usual.
- High Carbon Emission levels - an advice card for this type of event is created when the current carbon emissions levels exceed the average carbon emissions levels measured in g/kWh. This card tells the user that right now is a bad time to do energy-heavy tasks.
- Low Carbon Emissions levels - the opposite version of high carbon levels, tells the user that the current carbon emissions levels are lower than usual.

To explain the process of monitoring these events, we will only explain monitoring solar energy. The method for monitoring these events is almost the same but differentiates in what data we use.

1. fetch the last seven days, solar energy production data from https://www.energidataservice.dk/
2. Calculate the average solar energy production from the data.
3. Calculate the increase or decrease in energy production for the current production levels compared to the average levels as a percentage point.
4. Check if the current energy production is higher than the average, if not then return false, if true then, call the function **recentExists**, that connects to the database and check if there has been created similar advice in the last hour.
    - **True**: returns [true, pctIncrease]
    - **False**: returns [false]

5. if the function **monitorSolar** returns **true**, then we call the function **createAdvice**, which takes the parameters **pctIncrease** and **type**, which is the type of advice which should be created, for solar energy it would be: 1

6. Now the advice has been created and saved in the database, the last thing to do is add the newly created advice to every user profiles list of advice. So we use mongoose to query our database for all user profiles and receive them as an array of documents.

7. Then we iterate over the array of user profiles, and for each of the profiles, we call the function. **shiftEvents**, which removes the oldest event card from the users' array to make room for the new advice card.

8. after we have removed the oldest of the user's event entries in the array, we push the new advice card to the array.

9. And lastly, we save the user profile with the changes made.

The same process is then repeated for the three other events. To show how the function **createAdvice** is used to save the advice cards with the correct class and type of advice, we can take a look at how the function uses a switch statement on the type, to determine the format of the advice that is going to be created and saved to the database as shown in the code below:

```
1  switch (type) {
2      case 1:
3          advice_card = new AdviceCard({
4              class: "event",
5              grade: type,
6              title: "The Sun is out!",
7              message: `Heyooo, sun's out guns out.. we're seeing ${pctIncrease}% increased
       solar energy production at the moment, enjoy the clean energy. And probably the great
       weather, too! Remember to use sunscreen! Unless it's raining... we didn't check for that.
       Sorry.`
8          });
9          break;
10
11     // Other cases are removed in this code example
12     }
13
14 // Save AdviceCard to MongoDB database
15 return advice_card.save().then((new_card) => {
16     console.log(new_card);
17     return new_card;
18     }).catch((err) => { return err; });
```

Listing 7: Mongoose model for advicecards

Note that on line 11, we have mentioned that the other cases in the switch statement have been taken out of the example to give a clearer example while still giving the context needed to understand the functionality. To give an example of how all of the advice cards look when they have been created and rendered to the user in the application, the image below visualizes that.

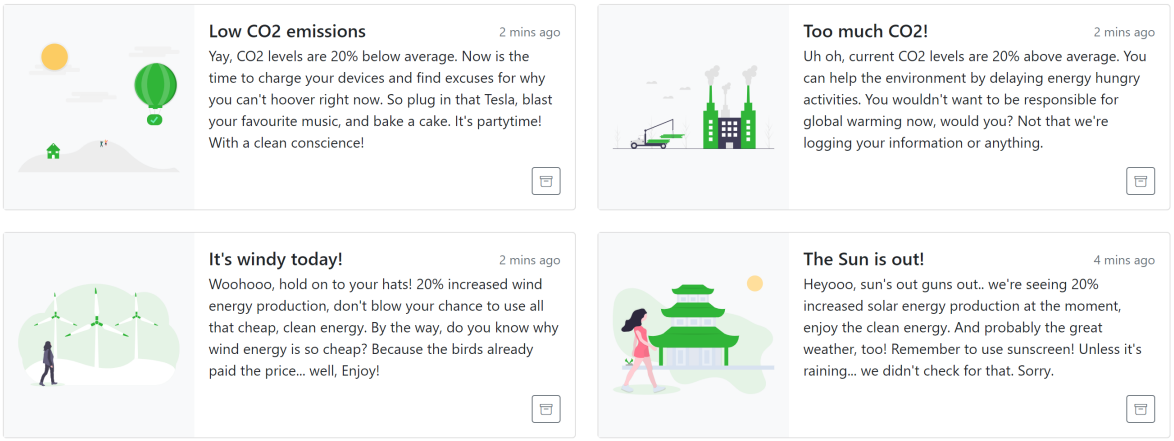


Figure 48: Screenshot from application, showing all event advice cards.

8.6.2 Recommendation card

Our second advice card, is based on the following logic control:
It finds all forecasted data, that is both less than the current carbon footprint, and that fulfills the users' sustainability goal. A function then loops through all the sets that fulfills these properties, and finds which one is most relevant for the user. It does this naively, by simply dividing how much is saved if the user waits until this moment by how long into the future it is:

$$\frac{\%Saved}{Time}$$

This means, the longer the user has to wait, the larger the saving has to be, before this is the optimal recommendation. Next it finds the one with the largest value and displays how long you have to wait to achieve this saving. Once the user gets a recommendation, it is displayed as such:

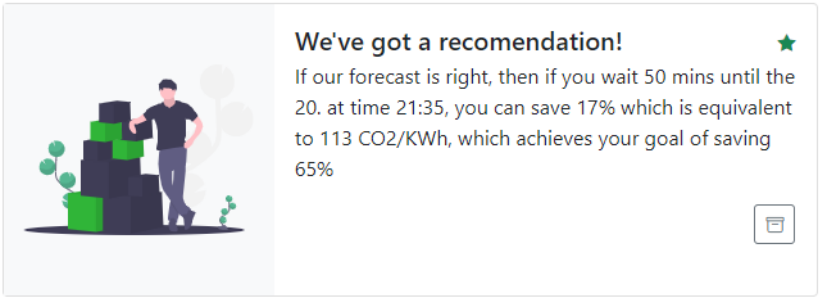


Figure 49: Recommendation Advice Card

Once the user gets this recommendation, then it is up to the user to switch off the power consuming items, as this is merely an advice for the user, not an automatic process.

8.6.3 Status cards

Status cards is the last type of advice card that we show to the user. The purpose of the status cards, is to give the user feedback on how well they are following their goal of reducing CO_2. We choose to give a status card each day and make sure that there is always a status card on the users dashboard.

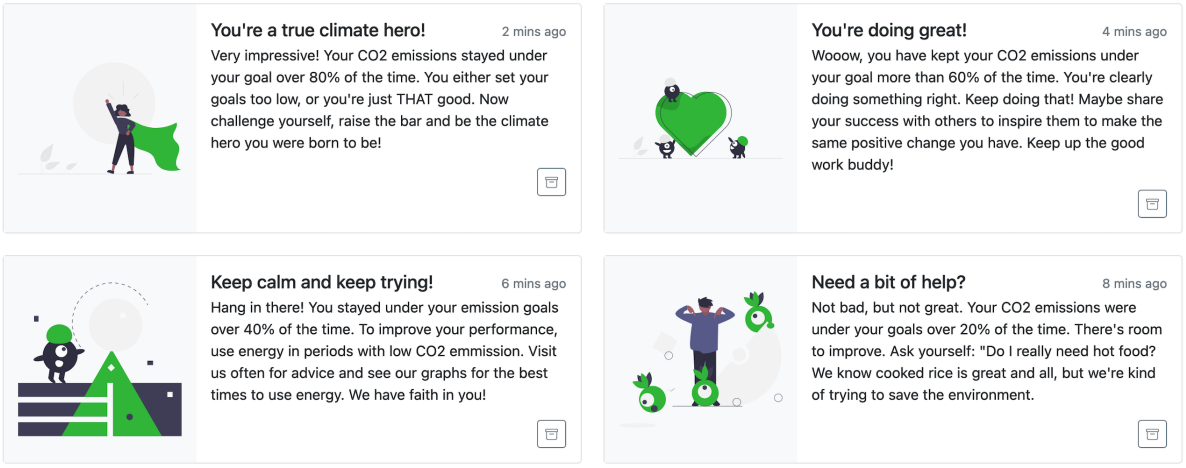


Figure 50: Four of our five status cards

The status cards is dependent on the users sustainability goal. This goal is a measure on the percentage amount of CO_2 the user wants to save and is calculated by taken the average CO_2 emissions over a 30 day period. The goal is satisfied if the current CO_2 levels is under the goal percentile of the 30 day average. As an example, a user have set their goal to save 25% and we have the following 30 day period CO_2 data $[4.3, 5.7, 9.6, 2.0, 13.4]$. The median of this data is 5.7 and 25% of the data is 4.3. This means to save 25% the user must only use energy when the CO_2 levels is below 4.3.

To calculate which status card will be shown, we look at how many times the user have stayed below their set goal during the day. Every five minutes we check if the user has met their goal based on the previous mentioned method and append either a true or false value to a list. Every day when the time hits $00:00$ we look at the percentage of true values in this list and show the appropriate status card. The five grades is shown in the table below:

%	Status description
100 - 80	You're a true climate hero!
79 - 60	You're doing great!
59 - 40	Keep calm and keep trying!
39 - 20	Need a bit of help?
19 - 0	Greta Thunberg would like to know your location!

Table 3: Different status cards

Every advice card more than a day old is destroyed so to not store unnecessary advice cards in our database.

8.7 The Users' performance at a glance

When all these functions are done, and the user has or has not used our application, we then show at the top of the home page how large the users carbon footprint is, along with how much the user has saved. This information is the total sum since the user profile was created:

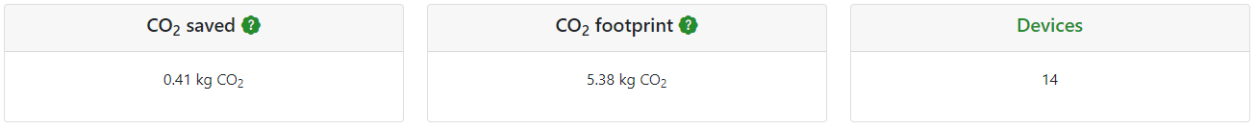


Figure 51: User's Carbon Footprint

For the total carbon footprint, the amount of electricity used each 5 minutes is multiplied by how expensive that electricity is as measured by g CO_2/kWh.

In order to calculate the saved CO_2, the difference between the 30 day median and the current CO_2 emissions is found. If the current CO_2 emissions is less than the 30 day median for the same time, then we use it, otherwise if the current CO_2 emissions is more than the median, then it is not considered saved.
If not discarded, then this difference is multiplied by the current electricity usage in kWh.

Example 1: Median CO_2 is 120g CO_2/kWh, current is 100g CO_2/kWh. 120-100 = 20g CO_2/kWh. Current electricity usage amounts to 100kWh: 20g CO_2/kWh * 100kWh = 200g CO_2
Example 2: Median CO_2 is 120g CO_2/kWh, current is 150g CO_2/kWh. 120-150 = -30g CO_2/kWh. No CO_2 has been saved.
In other words, the CO_2 saved, is actually a measure of how much of your CO_2/kWh used is under the median CO_2/kWh

9 Testing

Implementation is done, and following the modified waterfall model from Section 4.1.1 we begin the testing phase. Testing is done to identify any errors in the application, as well as ensuring that all the functionality, defined in Section 6 regarding requirements, is fulfilled. If our requirements are met after testing, then our application is considered a success.

9.1 Types of tests

The following list describes the different types of testing methods that we have researched.

- Functional testing is to validate functions of a software application. The result of a function test gives either a pass or fail at the end of the test[87].
- Unit testing has units or components to do an individual test of them to check whether or not these works in the application, and it validates the software code to see if it performs as expected[88].
- UI integration test runs an application in a browser to test the UI like buttons and other clickable without disrupting or modifying data on, as an example, the server or database. It is mainly used for front-end developers. By only using the front-end, it is fast to run the test[89].
- A scenario test will look at a user's angle and test any functionality, and this is called a Test Condition or Test Possibility. Scenario test uses cases of an application to complete end-to-end problem[90].
- Performance testing can test one's application speed, response time, stability, reliability, scalability by giving a load of tests to see how many people can visit a website or play a game before something breaks[91].
- End-to-end testing is going through an application from start to the end. The end-to-end testing purpose is to build user functions, conditions, and test runs from dependencies, data integrity, and communication with other systems, interfaces, and databases to complete from start to end[92].

9.2 Deciding on a testing framework

When choosing a testing tool, we are after a light weight framework that is easy to start working on, without the need for too much prior knowledge on testing. We are also after a framework that can do end-to-end testing as we see that, as the most fitting testing method for fulfilling our functional requirements. We now go out and find the appropriate testing framework that fit our needs.

9.3 Testing frameworks

During the process of picking a testing solution for our application, we considered several different options. First we did experimentation with unit testing using Jest, as we thought this method was the test method for JavaScript, and some of the group members had previous experience with it. We therefore tried to write tests with Jest [93], but felt it difficult knowing exactly what input should go into the tests to cover proper edge cases. Another reason for not using Jest, is the fact that some of our functional requirements cannot be said to be fulfilled based on single input and output parameters. There are more testing methods out there, not included in this report, but the following sections briefly describe some of the methods we researched before picking Cypress.

9.3.1 Jest

The testing framework Jest is used to perform unit testing. As stated before it was the first testing framework we experimented with before moving on to Cypress. Jest is a framework optimized for JavaScript, where it can run fast by first running all the failing files, afterward running the passable ones. Jest also runs the code in parallel by tracking how long it takes to organize the test files. Jest uses the notion of expectations for asserting if the test is successful, where the command for this is $expect(sum(1,2)).toBe(3);$. Since Jest is a unit testing framework, it will not be easy to test user experience where it is essential to see if different user interactions crashes the application. Therefore, we decided not to use Jest since Cypress is more fitted in simulating interactions from an end-user than Jest is.

9.3.2 HCL OneTest

During our research, we discovered HCL OneTest, which is a test suite that consists of API testing, functional testing, UI testing and performance testing[94]. First HCL OneTest can quickly test the applications performance and afterward validate and compare these results across multiple URL web pages [95]. Typical factors for performance tests are capturing and tracking the impact of load on applications, visualizing the process of the test, allowing to share the test results with others, and identifying run-time errors from the website [96]. HCL OneTest is also able to generate data to test with instead self generated data and they even have a method to do API testing for other data to see if these will work. We believed HCL OneTest was an excellent opportunity to use since it has multiple methods to test. In the end, most of the features offered by HCL OneTest are less necessary for our requirements and overly complicated for our testing purposes. In comparison Cypress is much more user friendly and entry level while focusing on end-to-end which is what we want.

9.3.3 Appium

Appium is a mobile test automation framework and it can be an excellent tool to do tests on multiple platforms at a time so that test developers can do this on their mobile devices, iOS, Android, or Windows SDK[97]. All the commands are at the command prompt to use, and this will create an Appium session to launch the Appium app. Appium has some of the same programming elements as Cypress, but it has more problems than Cypress since developers need to create a server in order to use this testing method. Appium also has limited flexibility in allowing users to use commands, which is terrible for new test developers, while Cypress makes up the testing creativity. Cypress is also easier to use for the web environment.

9.3.4 Cucumber

Cucumber has a few things that can help our group test for errors [98]. These following features is described below.

- Collaboration can be done by taking a Github repository and put it inside Cucumber's platform.
- Can capture code developed by linking an account to Jira. Jira is a software development tool to aid with time-management and task delegation.
- Git control is where the test code can always be up to date for all group members utilizing version control.

The cucumber testing method is using scenarios to tell if one scenario is true or false. Developers can, for example, write a scenario to find out if it is Friday or not, and the test will afterward give a message about the test result in the terminal. Cucumber has some similarities in writing in logical steps like "$given(todayisFriday)$" an action like that is easy to understand since it is high-level programming. Cucumber has some of the same features in Visual Studio Code, i.e., collaboration, live share, and git control, but it is not necessary for us since we already are working in VSC. As for the scenarios they fall under the category of unit testing and can be good for doing assertions, but in our case, it is more about an end-to-end test. Some of the features for Cucumber are still good, but we do not have much experience on testing, and we have not used the Jira software before and therefore deem the testing framework not suitable.

9.3.5 Wire Mock

Wire Mock is used to test the workings of an API. Wire Mock can use matches to check if there is an error in the request and response captures. Wire Mock is also able to use the API to do flexible deployment of different programming languages, but using Java is recommended because most of the commands are in Java. The matching method can be a good tool for our group to know whether our own API is behaving as expected. Taken from Wire Mock [99] the request matching can be done by the following methods:

- URL
- HTTP Method
- Query parameters
- Headers
- Basic authentication (a special case of header matching)
- Cookies
- Request body
- Multipart/form-data.

Wire Mock has two features to record and playback API traffic captures by using it's URL. Wire Mock has a simple web UI where developers can write the wished API endpoint and begin traffic captures by pressing a record button.

Ultimately we chose not to work with Wire Mock, because it is still only a software testing tool for API's, and does not have any other interaction with login, register or device page. Although this test method could be seen as very useful in testing the workings of our own API, we decided that it was of less importance to our testing, as we assume that the API from energinet.dk has been tested thoroughly before a publication, as the data is run and released by a public infrastructure company.

All in all, we have examined a number of different testing frameworks that all have a wide range of functionalities that might come in handy when doing tests. We were after a beginner friendly testing framework that offered end-to-end user testing and settled on Cypress.
The test method can be done over a browser using a URL address or using the integrated approach by downloading Cypress using the 'npm install cypress' command. Using an end-to-end test framework like Cypress has the following benefits compared to other testing software tools:

- Time travel - Takes snapshot of each iteration in the test
- Real time reloads
- Spies[3], stubs[4], and clocks [5]
- Consistent results - uses other test architecture than Selenium or WebDriver
- Debuggability - like in Chrome DevTools
- Automatic waiting - Never add waits or sleeps to your tests. Not using Async.
- Screenshots and videos [100].

Before describing the testing phase for each of our application's parts, we have illustrated what parts we are going to test in Figure 52 below.

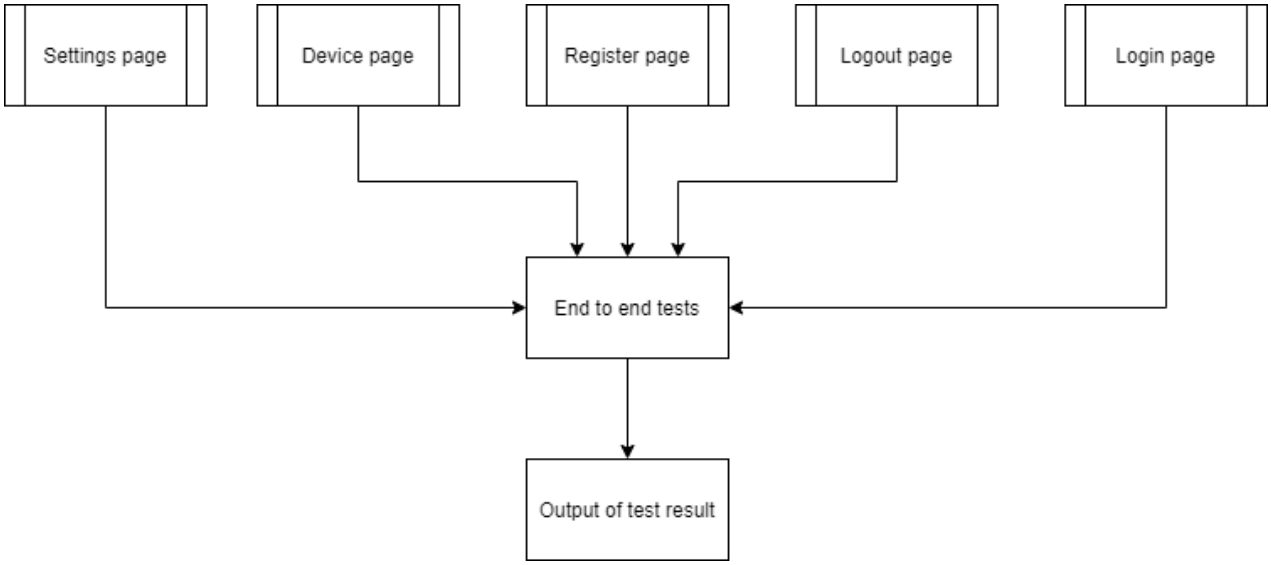


Figure 52: Cypress end-to-end test [101].

As the figure depicts, we make tests for our settings, device, register and login page, as well the logout function. These end-to-end tests made with Cypress culminates in an output with the test results.
First, we test our login and registration system since these two functions are vital for the user to start using our application. Letting users add, edit and delete their devices is also part of our must-have requirements, which is why these functions are also part of our tests. Lastly, we test settings, as the user is required to set a sustainability goal to receive recommendations.

9.4 Login and logout test

The first thing a user sees when accessing our web application is the login screen, so we have devised a login test, to test if an incoming user is able to login with hassles. When someone is attempting to log in, let us assume they do not have an account and wants to register. Recap that the login page contained a link to registration. Cypress has a command to use called 'click', which we use to click on the register link, and afterward, the test will switch sites back and forth from login to register to log in again. The last part of the test will type in all the

[3] Stubs is for asserting that code was called but preventing it from executing
[4] Spies is for asserting that code was called without interfering with its execution
[5] Cypress can control time for deterministically testing code that is time-dependent

necessary information fields to log in. In our case, it is the email and password that gets typed in.

When the user has logged in our application is supposed to store an authentication cookie on the users browser. We will be testing if the user actually get this cookie after log in. Cypress has a method to get stored cookies that we compare with a preexisting cookie that was associated during an initial registration. When the tasks described above are finished, the user lands on the homepage. At the top right corner there is a small user icon for seeing if it is the real person who just logged in. As for how to logout it does require the user to first login before, that is why logout test is almost the same as login. To logout, the test will simply press the person icon and then press logout. The implementation of the test code can be seen below 8.

```
1  // Registration and login button
2  cy.get('.action-register').click();
3  cy.get('.action-login').click();
4
5  // Login phase
6  cy.get('.action-email')
7  .type('fake@email.com')
8
9  cy.get('.action-password')
10 .type('Ming#1359');
11 cy.get('.action-submit').click();
12
13 cy.getCookie('auth').should('have.property','value','608911a0070ea135f09e3705');
14
15 // Logout phase
16 cy.get('.action-menu').click();
17 cy.get('.logout-button').click();
```

Listing 8: Login and logout test

9.5 Register test

The Register test has almost the same code as the login test, but requires first and last name in addition to email and password in order to create a profile. To confirm a user password, there is another text box to write the password a second time. If any users already have an account, then there is no need to register, therefore, the test will attempt to log in instead of being stuck at the register. Afterward, it will return to login and submit the given email address and password. Below is a code snippet showing what the register test needs beyond what the login test does.

```
1      // Get a first and last name input
2      cy.get('.action-first-name')
3      .type('Test')
4
5      cy.get('.action-last-name')
6      .type('Testsen')
```

Listing 9: Register test

9.6 Device page test

The device test has five functions to see if it works to add, inspect, edit, and delete devices. First off, we perform a test to see if we receive an HTTP response from the device overview page. It should show users a list of devices, the power consumption, status, edit icon, and delete icon if there are any devices. The URL to find and

go to the overview of the device is $/devices$. If there are devices, their names are a link that will lead to the device detail page, so users can see more information about a specific device if the test works.

9.6.1 Add device test

While the device list is initially empty on first time registration, this test will go through the process of adding a single device. In the beginning, the test will click the $AddDevice$ button. Add device site is going to visit a device site without changing the URL for this test. Afterward, the name and energy consumption will be written using .type. Those active time boxes are just like regular buttons, and it will simply press a selection of them for turning on devices during these times. At last, the test is going to click on the add button to add a device. The code snippet for this test can be seen below:

```
1     cy.get('.add-device').click();
2     cy.get('.name-input')
3     .type('samsungtv')
4
5     cy.get('.energy-consumption')
6     .type('136')
7
8     /*
9     This will enter the time interval for the device to turn on.
10    More of them in the real test.
11    */
12    cy.get('.time-00').click();
13
14    cy.get('.submit-device').click();
```

Listing 10: Add device test

9.6.2 Device edit test

After adding and seeing their device, users will most definitely want to edit the device if they have miswritten something. The device edit test has the functionality to see if a device can be changed in the database. If a device can vary, this test will first press on gears to go to the edit site, and let's say the name, power consumption, and the active time are not correct. To clear the text, the test uses the command .clear() to delete all text from a specific text box, and then it changes which active times are set with .click(). After these things have been done, the edit test will click on update to save the device's updated values to the database.

9.6.3 Delete device test

Device delete test will delete a certain device after it have been added and edited it for saving storage in the database, this will not directly delete a device after clicking on trash button, but it will first give an alert message to ask the user to delete the device. After clicking OK to the alert message the device will be deleted.

```
1     cy.visit('http://localhost:3000/devices');
2     cy.get('.delete-device').eq(1).click();
```

Listing 11: Device delete test

9.7 Settings test

Settings test is about updating basic user profile information, and these changes can be the name and currency. Another thing is sustainability goals: how many percent of the CO_2 a user wishes to save, and the user can

change that at any time, so this test will do it one time to confirm that it is changed. The first part of the settings test is going to the home screen where, at the top right corner, by clicking the settings button, the test will see if it has some errors with the HTTP, minor mistake, or syntax. If the test succeeded, the settings page would appear. In settings, the test will first edit the first and last name, but there is also currency to be edited, and this will select and not write down Euro, for example. By clicking update, the homepage screen is updating with Hello Fake instead of the last name, and the test will also go to the homepage afterward.

```
1     cy.get('.first-name-text').click()
2     .clear()
3     .type('Fake')
4
5     cy.get('.last-name-text').click()
6     .clear()
7     .type('Fakesen')
8
9     // Invoke is a function on the previously yielded subject,
10    then use that to be equal to the subject.
11    cy.get('select').select(['EUR'])
12    .invoke('val')
13    .should('have.equal', 'EUR');
```

Listing 12: Settings test par 1

The second part of the settings test will test whether or not the sustainability goals can be updated by writing a new percentage of how much will a user try to achieve the sustainability goals of their choice. However, still, it is not required, and devices will therefore not turn off automatically. The test goes to the settings page again. Compared to the first test, where it selects a currency, this test has a text box to test by just writing down a number. Afterward the test will click the update button, as shown below:

```
1     cy.get('.sustainability-choice')
2     .clear()
3     .type('46')
4
5     cy.get('.update-sustainability-choice').click();
```

Listing 13: Settings test part 2

9.8 Result of end-to-end test

Along the way there were some errors while writing the tests, where many of them would say they "cannot find the class" or basic syntax errors which caused the test application to crash. The hardest part was the logical errors that causes it to choose the wrong HTML element since two or more devices(objects) have the same classes and id. We managed to fix them with Cypress's help to indicate which and where the errors are and describe them to find other functions to support our application and make it find the right element.

After running tests on the different parts with Cypress, we have come to the conclusion that all of our test above comes out as a success where no error is detected throughout. The below screenshot53 is taken after we have run the tests.

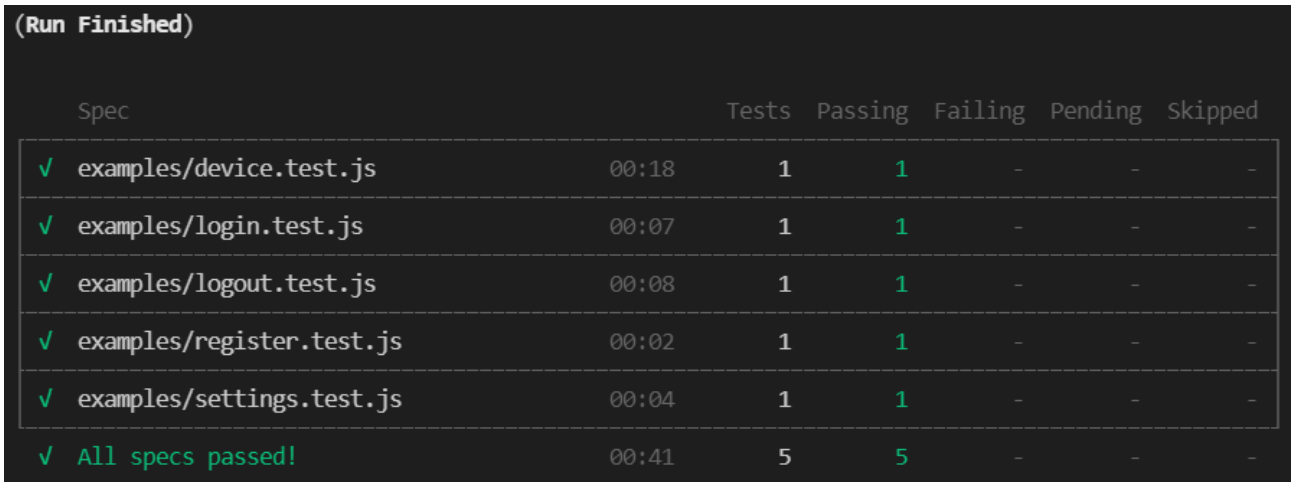


Figure 53: End results of end-to-end test from Cypress

9.9 Testing with users

After testing our application with a software testing tool, that is supposed to simulate a user, we then decided to bring in actual users to test the application. The testing had a special focus on usability, how informative the application is, user experience, and user's opinion on usefulness of the application, and how easy the application was to understand and figure out. The reason we did a user test, is to test if our solution is viable, and that if our implementation is wanted by users in general. When you implement something, it is really clear what the different things do, in the end you start to assume everything is easy and simple, yet a user who has never seen the product might think otherwise.

We considered asking our fellow students at AAU to be our test users, but reasoned that they would likely be too technically competent to represent the average user. Therefore, we each asked a couple of people chosen from among our friends and family to be our test users. We chose these people because we assessed this group would represent an average user base, possessing numeracy and web-related skills in a range from low to high. Out of the 17 people we asked, 9 of them filled out the test survey containing 22 questions, while we are not sure how many of the 17 test users actually partook in the test. Though 9 people is a relatively small number of respondents, a study indicates that just 4-5 test users is enough to find 80% of usability issues [102].
The people who agreed to be testers were given the set of instructions listed below, on what they needed to do, and then left to their own devices to experiment with using our web application for themselves, figuring things out. As such the test took place online, at each tester's convenience, with no interference from us. We cut up the test into 2 parts ranging over 2 days.
Day 1:

- Log onto "Sustaininator.eu"
- Create a profile
- Find 10 devices in your household where its energy consumption is given in watt. Add these devices to the "device" page, and add what time of day you normally use these devices.
- Go into the settings page in the top right corner, and adjust the "sustainability goal" [0-100].
- Explore the web application, experiment with the different options and get a feel for the web application's general layout and usability. After you're done, logout.

Once day 1 was over, they had to wait a day in order for the newly added devices' data to be collected and compiled.

Day 2:

- Try to edit some of your devices
- Try clicking on one or more devices, and see their power consumption.
- Observe some of the messages on the homepage (Events/Advices)
- Go down to the bottom of the homepage and explore the graphs
- logout of the web application once you're done exploring, you will then be given a survey you need to fill.

After the testing was done, the user had to answer 21 quantitative questions, some of whom were multiple choice, (on a scale from 1-5), some were yes/no questions and 1 was a qualitative question (F). From the quantitative survey the following may be concluded:

- On the questions surrounding usability (3 questions), most people agree that it is somewhat (4/5) to very easy (5/5) to navigate the application and use its functions, such as adding and editing devices, with the mean being a 4.22 with a standard deviation of 0.6.
- On the questions related to how useful the application was (7 questions), most people's answers varied a lot, ranging from a 1 to a 5, with the mean being 3.62 with a standard deviation of 1.2.
- Out of 6 yes/no questions surrounding how insightful our application was, 80% answered yes on average that the application was insightful with a standard deviation of 10 percent points.
- When it comes to if users prefer graphs or advice cards, 11.1% preferred graphs, 33.3% preferred advice cards, while the rest actually like a combination of both. Meaning that most users actually like having multiple options to help them with understanding the data, and not just a site that tells people what they need to do.
- when we compared which of these informative functions was most useful, 87.5& thought the bar at the top was useful, compared to 50% who thought the graphs and advice cards were.
- a little over half of all people didn't find their carbon footprint surprising, but 66.7% said that they would follow the advice cards recommendations and wait with using devices in order to reduce their carbon footprint.

For the qualitative question: "Here you can provide any additional feedback you may have. Eg. anything you thought worked really well or very badly.", people had the following to say:

- + It was awesome to see a visualization of the Danish electricity consumption and carbon footprint.
- + Loved the advice cards
- + Very useful and user friendly, if one wanted to monitor the households energy consumption.
- - The web application seemed to imply prior knowledge, that the user didn't have, which made it confusing to understand the different cards and graphs.
- - The graphs were quite confusing and hard to work with: The text describing the graphs were too long, too confusing and too badly worded.

- - When adding devices, you can only adjust the hours throughout the day, not how long you use it. A lot of devices are not used for that long. As an example: You really need to love toast to use a toaster for an entire hour.
- - When adding the devices, you have to go around the house looking for the power consumption of devices, and the web application ends up timing out in the mean time.
- - Many users had no idea what "Watt" is, how to find it, or how to conceptualize it.

User suggestions:

- Have a front page, that describes what the web application is about, what its goal is, what it does, and why you should use it.
- Since adding devices is so important, try to make the web application guide the user through the different functionalities of the web application, so that the user isn't overwhelmed and helpless.
- It would be nice if there were standard values for different home appliances, such as a fridge, so that the user didn't have to find the exact power consumption.
- The advice cards were enormous on safari, maybe support safari?
- When you have to type in your password, there should be an example password.
- In the device page, there is a red/green icon, make it say "on" or "off", as i didn't understand what the green/red color symbolized.

A lot of these suggestions and critique are based on the fact that you have to add your own devices. If we were to actually implement this application in a smart home, this would not be a problem at all, as it would be automated.

If this application was to be developed further, an introductory page should be created to guide the user through the application, and explaining the different values, graphs and cards. A lot of things seemed to invoke confusion, and should therefore be given with examples and some explanatory text, instead of merely symbols.

From this 2 part survey, it may be concluded, that the web application is very easy to use (4.2/5), but invoked a lot of confusion surrounding terminology. The usefulness of the application was reasonable good at a 3.6/5, though it did seem like it was quite a dividing force, as half seemed to want to use it, while half did not seem to be that motivated to use it. This may suggest that our application is a solution to a problem that users may not have realized they had, or believe they have. The information was generally very informative, and simply for curiosity's sake, the web application was interesting for most users.

It's worth mentioning the possibility that our test survey responses were skewed favourably since we asked friends and family.

10 Discussion

In this section we look at some of the changes and further development that could be made, to improve the application.
Based on oral user feedback provided alongside the survey responses, we realized that users are inclined to do what they are used to, and were quick to give up when things did not work the way they expected them to. This makes small things, such as security improvements with the password, a major inconvenience for the user, as the added security requirement of a symbol in the password proved to discourage some of them, even though in our eyes it was clearly defined on the very same page. The same goes for the survey results, even though the users may answer yes to one question, they may answer no to a question with the same meaning, but slightly different wording.
At the same time, the users seem to both hate and love some functions at the same time, finding them both useful and confusing. It nearly seems as though the users are unsure what they really want or like.
It may be worth it to do preliminary testing through the development process in order to never confuse what we believe is useful and understandable, with what the users believe is.

10.1 Improvements

Our application is complete to the extent of our outlined limitations and requirements, but we could add many new functions to improve the application, and thereby provide a better user experience. As an example, an API could crash, but the users would not be aware it had happened, because we do not have a notification system that can notify them. This could result in confused users, whose graphs are created from data fetched from the API. Another example is the device page, this could be improved by implementing a drop-down menu with pre-made devices, which the user could pick from.
When we want to test our web application, one test could be sufficient, but the web application could have multiple means of testing. Right now, we simulate a user with cypress, but we cannot test whether or not a function is working. The following subsections will go into greater detail with some of these possible improvements.

10.1.1 CO_2 and forecast graph improvement

The forecasted data is only a bit better than if you were to randomly plot data points within standard deviation. This means that our forecasting is quite bad, either the dataset is too unpredictable, or the algorithm is too poorly implemented. It is likely both scenarios at once. The problem was that the entire ARMA model kept running into crazy numbers and errors that made no sense. What could have helped the model was the knowledge of PACF (Partial Auto Correlation Function) and ACF(Auto Correlation Function) to be used, to find out which order terms should be included. It was only after the final iteration of the algorithm was implemented, that the theory was really proven to produce correct forecasts.

For further improvements to the forecasting, would be rewriting the entire forecasting function, as an improvement for ourselves, the math and theory behind such functions should be clearly understood on a deeper level before experimenting with implementing the function. Nonetheless, it might not be an issue, as it is an acceptable model, to showcase our product.

Another problem the user might have is, that it can be difficult to grasp what X kg CO_2 actually is. As an immediate improvement, there could be a visualization for how much it actually represents. This could be by showcasing if it's under or over average consumption and what it amounts to in units we can understand. As an

example, the application could convert the saved CO_2 into something more fathomable, like "you have saved *X* kg of CO_2, which amounts to *Y* KM's driven in a car or *Z* kg beef eaten".

10.1.2 API improvements

Since our application depends heavily on the use of third-party APIs, namely the energy production information APIs from energidataservice [76], one could argue that this could lead to a weak point in our application. We have in the group discussed that we should be aware of that API downtime could lead to our application not functioning correctly. On one or more occasions, we encountered the error - fetchError: socket hang up as shown in the image below.

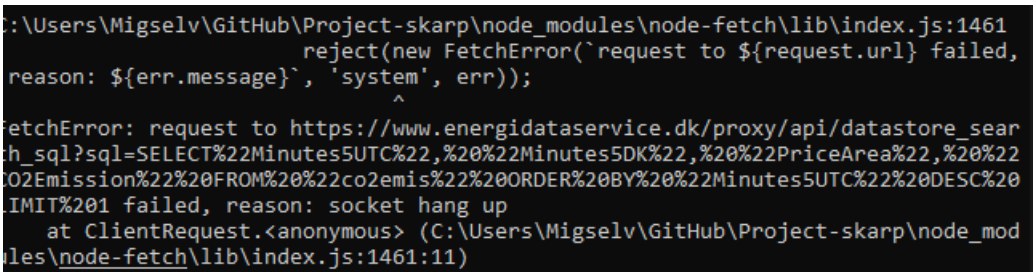


Figure 54: Screenshot from console encountering the error

This fetching error is encountered when we send a request to the API, and no timely response is received, and the connection is then closed. This error would be relatively easy to correct in the application, and we could catch the error and rerun fetch after a short period of time.
A more unsolvable problem we encountered during the development of the application was API downtime when the energidataservice ran maintenance or updates on the servers running the API. This only happened once during the development phase, and it affected the functionality of the CO_2 emission forecast, where we need consecutive historical data to provide the user with a more accurate forecast. We can't solve this problem because we are only consumers of the API and not the provider.

Even though this might seem like a significant problem, it was so uncommon that we only encountered API downtime once during the project duration, and this happened at night. Furthermore, the API is owned and administrated by the Energinet [103], a publicly owned company under the Danish Ministry of Climate, Energy, and Utilities [104], and therefore it is highly unlikely, that it would be down frequently.

Even though the downtime of energidataservices API is something we cannot alleviate, we could provide a better service for our users by providing some indication of the APIs stability right now and over time. As a good example of this practice, we refer to the images below taken from Coinbase Pro's dashboard and API status page [105].

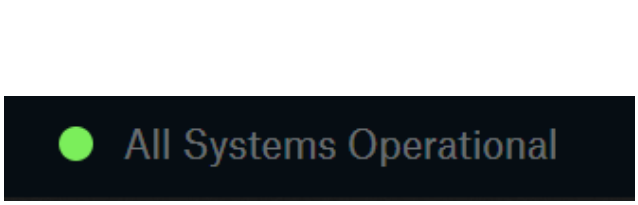


Figure 55: API health indicator from Coinbase Pro dashboard

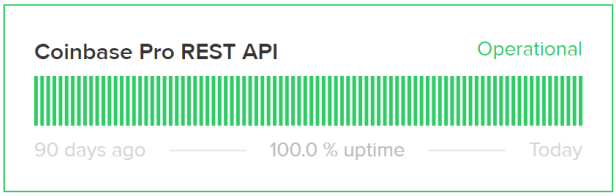


Figure 56: API uptime monitoring from Coinbase Pro

If we, as a group, have had the time to implement a feature like this, it could have been an improvement to user experience. Another improvement could be to store the old data that we collect from the API on our database. Since at the moment, even though highly unlikely considering the API is state-run, if that API was ever to be shut closed on malfunction over a period of time, our application would be rendered useless.

10.1.3 Adding device improvements

Currently adding a device requires the user to input a name, power consumption in watt, and active times. Our application is not capable of understanding what type of device is added or how energy-efficient it is. One improvement to devices could be the option to select a device type, like "Refrigerator" or "Television". In addition to device type, another improvement could be giving the users the option of selecting an Energy label (A, B, C,...). The energy label is a ranking system for how energy-efficient a product is, and is controlled by The EU energy labelling and ecodesign legislation [106]. This addition to the application could allow for device-specific recommendations, like "Your dishwasher is Energy label "E", save X kg CO_2/year by investing in a label "A" dishwasher".
Lastly, a big improvement to devices, could be a continuously updated list of specific devices. This list would be made by scraping wattage information of consumer electronics data sheets, allowing the users to search for their specific device.

10.1.4 Testing improvements

We did a test to simulate a user, to ensure there were not any errors in our application processes and to prevent users from having a negative user experience, but this is just about testing the processes the user goes through, and not unit testing the code block by block. As we mentioned in the other tests section 9.3, we are aware that there are other methods of testing, such as Jest, which is for unit testing, and it would be beneficial for us to test our application functions to narrow down if there are any overlooked errors in the functions. In the application, we have a lot of functions that we are expecting to work without any issues, but we cannot be sure if it works all the time because if we use API that is from third-party sources, then we do not know if the API would not function as expected, have any errors or many other scenarios. In a scenario where we have functions that depend on previous data requested from a previous API request, that could lead to a chain reaction of malfunctions for parts of the application.

11 Conclusion

In this section, we conclude on our combined research, which centers around our problem statement "*How can we build a web-based application to inform Danish smart-home owners when to use their appliances based on the availability of sustainable energy*" and its three sub-questions.

How do we collect the data necessary to help users monitor electricity prices and carbon footprint?
We have collected data on CO_2 emissions and renewable energy production necessary to provide actionable recommendations and advice. When it comes to electricity prices, our survey results indicated that users are more inclined to postpone their energy consumption to support the environment rather than their economy. This result shifted our focus away from electricity prices and instead focused our attention on reducing CO_2 emissions.
Thus we ended up only collecting the CO_2 and renewable energy data. This data was freely available by Energinet, the organisation responsible for the Danish electrical grid, on their web-service energidataservice.dk. This, combined with storing simulated device data in a database, allowed us to keep track of users' fictive electrical consumption and inform them of their carbon footprint.

How do we present the data in an organized way?
As our problem formulation states, our primary goal is to help owners of smart-home technology better understand and know how to utilize energy from sustainable sources such as wind and solar energy. Throughout our research and the implementation of the project, we found that a large part of understanding the current energy production is to interpret and understand data from multiple datasets. During our research, we found a cyclical study called "Program for the International Assessment of Adult Competencies" (PIAAC), where they examine adult competencies, including their abilities in numeracy and understanding graphs. They found that 3 out of 10 Americans have trouble understanding even low complexity charts/graphs, and 6 out of 10 have issues with medium complexity, such as identifying trends in a line graph. Therefore, to not alienate a significant section of the potential user base, we had to develop multiple ways to present data, such that most users can benefit from this knowledge. From this research, we came up with a system of ordering the data we display to the user, in a top-down fashion, into three distinct parts of increasing complexity: the informational bar, the advice cards, and the graphs, which permits users to grasp their data in increasingly complex levels. This way, users are not overwhelmed but can refer to the graphs for additional details. Our final survey of user testing found that 80% of the surveyed users felt that the application was providing useful insights. When it came to the users' preference in informational tools, 11.1% preferred graphs, and 33.3% chose the advice cards, while the rest of the respondents liked a combination of both. And in usefulness, 87.5% thought that the informational bar at the top was useful. With this, we can conclude that we, through multiple informative tools, successfully narrated complex information to our users.

How do we collect data on non-smart devices to cooperate with our web app?
The data for non-smart devices are created by simulating the use of devices. Due to our limitations, our solution does not include a hardware aspect, thus there is no way for users to connect actual devices. Having the users enter the specifications of their devices, we can simulate any device in their home and collect simulated consumption data. The generated data is accurate, as long as the information provided by the user corresponds to an actual device in their home. While connecting to actual devices is an interesting and important aspect of sustainable homes, the use of simulated devices allows us to reach both smart and non-smart device owners collectively. As we concluded in our initial survey in Section 2.5.2, only 13% of the users had more than 5 smart

devices, making them a subsidiary audience. Ultimately, the chosen solution supports our problem statement while permitting us to work within our limitations.

In conclusion, we have created a web-based application that solves the first two sub-problems of our problem statement. At the same time, the solution to the third sub-problem ended up being simulated devices based on real users' supplied data. Our application is informative to the user. Additionally, the application gets a usability score of 4.22 and a usefulness score of 3.62 on a scale from 1-5 based on our user survey. The biggest improvement is implementing energy monitoring on real devices to solve the third sub-problem in our problem statement. The largest point of critique we received regarding the application was its usefulness. Some users did not completely understand the purpose of the application, as it lacked an explanatory element. Another point of critique was that you have to add devices yourself instead of connecting automatically with actual smart devices. From this, we can conclude that our project was a success with a clear vision of where this project should be headed with points of improvements if we were to develop it further.

12 Process analysis

In this process analysis, we will reflect upon the group's process in completing the P2 project. We will explore what competences we have acquired, as a group, during the project and what competences we have from the former projects to solve both project-related and interpersonal hardships. This process analysis aims to improve our understanding of the process involved in identifying a problem area and narrowing it down to a specific solvable entity. Through the this analysis, we will be able to identify tools or methods that can be used in future scientific work.

12.1 Analysis framework

PBL is 'problem-based learning', and in this context it means learning by collaboration in a group on a specific topic. The idea is that theoretical learning is coupled with practice. The application plays a major role in this project, since a product must be developed to show the process of the group's work. The prepared product must then be tested to gain knowledge of whether the product solves the problem formulation, which can be done by testing with users in our case. The students take responsibility for their own learning, and using the PBL methods to structure their learning through the projects.

12.2 Process towards the problem statement

The process towards the problem formulation began with group discussions and a lot of research about the topic of smart home energy monitoring. We thought a lot about the environmental issues we have in society and wanted to make a difference by applying smart home technology. The project was conceptualized when we got the idea of moving power hungry activities to a time where the energy produced was coming from renewable energy sources only. With this idea in mind, we had to research if it was actually a viable standpoint that would have any impact on reducing CO_2 emissions. We did an analysis on the theoretical maximum of CO_2 that this idea could possibly save. We found that it would make a small difference but enough that we went with the idea. To confirm that other users would be interested in our project idea we made a survey. The survey confirmed that users would actually be inclined to move the use-time of certain appliances if it meant help reduce their CO_2 impact. These survey results formed the final problem formulation. We had gone through many iterations of the problem formulation leading up to the final one, but settled on "*How can we build a web-based application to inform Danish smart-home owners when to use their appliances based on the availability of sustainable energy.*".

12.3 Group collaboration

During the majority of this project the campus of AAU have been under lockdown because of the covid-19 outbreak. This meant that we have been forced to work online from home. Our day to day work would consist of having daily 30 minute meetings to discuss what needed to be done and by whom and end the day at around 4:00 pm. This online group work affected the motivation for individual members and seemingly also for the supervisor quite a lot. Having worked together for months on end, without knowing how each other looked put a strain on the efficiency of work. In the middle of the semester, we felt it necessary to switch up our group collaboration method, as the delegation of work tasks and their completion would be increasingly harder to keep track of as the project grew in size. It was also a wish for a more structured way of working together. The new method of working together was a team based approach where we would split the group into three different work groups. The smaller groups would be responsible for a subsection of tasks that needed to be completed. The work was then evaluated every week to summarize what had been worked on and to create

new work groups. The new groups were chosen so that it complimented the members' competences in a way where there would be room for competence development. As the project went forward, we ended up phasing out this method of work as we lost track of the different work tasks that could be out delegated and it required a lot of effort and long discussions coming up with good specific tasks. This meant that we spent less time just diving into the subject and figuring out the details along the way. A future improvement, which might have helped us stick to this method could be a weekly talk, where we could discuss things in the work model that needed improvements. This new method was heavily inspired by two of the group members earlier positive project experiences, but it might be that the method is not suitable for every group.

On the topic of competences we have one in the group who was already familiar with a lot of the web development part, which meant he would lead a huge part of the implementation decisions. His work helped immensely in taking the design from the mockups and create a structure to work from. Apart from this, we have two very analytical group members where detail would matter a lot. This would mean that almost everything had to be looked at and discussed before it could become part of the application or report. Taken from Belbin's nine team roles [107], we have had a good mix of competences ranging from imaginative to specialists. During the initial phases of the project, we all came with very bright ideas that helped kick start the project.

12.4 Supervisor collaboration

The supervisor collaboration was great in the start, as it was highly structured and provided the group with a clear vision of how to progress with the project. Our group prepared questions and sent them to our supervisor ahead of supervisor meetings, which helped to structure the meetings. Throughout the project, the supervising style seemingly slipped into Laissez-faire guidance. A possible reason for this is that we didn't prepare questions ahead of time to the same degree as we had done earlier, and additionally due to the fact that it is hard to connect with people and stay motivated when you are not working face to face. As such the online meeting format did complicate communications and, combined with the fact that none of us have English as our first language, sometimes resulted in minor communication issues causing us to not always understand the feedback given even though we thought we had. As a group, we should have talked about the exact form of supervision we wanted in the start, as that may have helped a lot in settling expectations, and for the next project, that is definitely something we should place a greater focus on.

12.5 Project management

Since the project consists of both a software solution and a written assignment, we have used multiple tools to manage the process of completing the project. For the overall management of the project, we have used Trello boards to keep track of our tasks, both unfinished, finished, and tasks on hold. Since the process of completing an assignment with multiple moving parts and collaborators, it has been an enormous help to visualize the tasks and keep track of who is working on what. When it comes to collaboration on the software implementation part, source control software like GitHub has helped us collaborate on the programming side of the project. Since we were multiple collaborators on the software, the use of GitHub helped us manage the programming aspect of the project. As a source control software, GitHub can prevent us from accidentally overwriting each other's work and keep each group member up-to-date on what each member has contributed lately, ensuring both equal workload and correctness of the individual contributions. For the written part of the project, we have used the LaTex editor Overleaf, which is software, we have used since the first semester for collaborating on written assignments. Overleaf is especially useful for us as engineering students, as it is very easy to insert parts of code, with good formatting, to describe our software solution. In conjunction with Overleaf, we also used

Zotero, which enables us as a group to share and collect resources such as articles, websites, etc. conveniently. Maybe the biggest benefit of using Zotero is that it was very easy to export resources to Overleaf with correct formatting.

12.6 Meta reflection

During the completion of this process analysis, we talked about how we as individuals felt about the group and how we could have improved on working together on this project. One of the things we talked about was that we as a group often spent significant amounts of time at the start of each working day, discussing what we had done the day before, what could be better, and what we should be doing for the rest of the day. These discussions yielded many useful insights, but quite often, it ended up being just a productive talk, where we didn't take any notes. Which resulted in spending additional time after these discussions, to figure out what we had agreed upon and sometimes discussing the same topic multiple times. For future projects, we would use the option of taking turns with the designated role as a note-taker during these morning discussions. All of the group's members agree that this could be something that would streamline the process of these sessions. This was something we did during our supervisor meetings, and quite successfully. During meetings, we always had one to two people volunteering as designated note-takers during the meeting. This ensured that we would always leave the meeting knowing we had all of the recommendations and information that our supervisor gave us, written down. Even if one of the note-takers, missed something the other would have written it down. This was a tremendous help to the group and ensured a high productivity from the meetings.

Something more directly linked to the process of implementing a product, was that the design aspect of the project proved to be more complicated than we had anticipated, as this project is the first in which we have this visual component. With mock-ups and flow charts providing a foundation for the upcoming work, we could have made designs that more closely or realistically depicted what our end product might look like. We discussed if it could have been prevented if we had dedicated a specific frame of time to make these design choices. This leads to how having a timeline containing our milestones and setting deadlines would have helped us structure the different phases of the project so that every group member would have a clearer vision of what we needed to accomplish.

We kept working on the application quite close to the project hand-in date. This made the last few days of writing the report more stressful, than if we had kept to the deadline we originally made. From being ahead in the project's start, we perhaps ended up biting off more than we could chew and ended up being a little strapped for a time closer to the finalization of the project.

Part of the challenges we faced during the project, could be attributed to the obstacle of being physically distant from the group due to covid-19. The dynamic in digital communication is very different from open discussions amongst group members sitting across from each other. Furthermore, the physical distance somewhat evolved into a lack of mutual understanding. We believe that members of the group would have more profound respect and sense of responsibility if we had a chance to see each other in person more often.

12.7 Conclusion

These are the key takeaways from working together as a group we believe will help us in future project work:

Reflecting on the process towards a problem formulation has taught us the importance of establishing the foun-

dation for the whole project. This involves making sure the problem we try to solve is a problem and that the solution makes sense to pursue. Naturally, the research we did was essential. Still, the inclusion of a user survey in the early process proved valuable for our further work and is something we will keep doing in future projects.

Our collaboration as a group has been made difficult by the need to conduct our studies online. Discussing issues and reaching an agreement as a group is more problematic when it doesn't occur in person. Delegating tasks to teams of two has been fairly efficient because each team can work out the details of their particular part without disturbing the others. In the future, we could benefit from giving each other more feedback on their completed work in weekly meetings.

In future projects, we must settle our expectations for our supervisor, including which role the supervisor is supposed to play ahead of time, to maximize the utility of the supervision. In order to manage our project, we have utilized a large variety of software tools to help us manage and optimize our development process. These have made us more efficient since the last semester project, as we slowly learn more and more tools, which works for what situations.

To improve the value of our discussions we will assign one person the group role of noting key elements so we don't waste time on discussion having something concrete to take away from it.

In conclusion, despite the significant hardships that Covid-19 and its lockdown has bought along, the group has been a joy: Everyone has notified each other, contributed, and helped each other where one may be strong and another weak.

References

1. *Electric Household Appliances Market Size & Share | Global Report 2024*, en-US, (2021; https://www.gminsights.com/industry-analysis/electric-household-appliances-market).
2. 1. Jan 2020, *Tech Nation: number of internet-connected devices grows to 10 per home*, en-GB, Jan. 2020, (2021; https://www.aviva.com/newsroom/news-releases/2020/01/tech-nation-number-of-internet-connected-devices-grows-to-10-per-home/).
3. K. G. Journalist, T. P. F. i byggetekniske løsninger, *Så meget el, vand og varme bruger en familie i gennemsnit*, en-US, (2021; https://www.bolius.dk/saa-meget-el-vand-og-varme-bruger-en-gennemsnitsfamilie-279).
4. Energistyrelsen, *2019 energistatistik*, Jan. 2019, (2021; https://ens.dk/sites/ens.dk/files/Statistik/energistatistik2019_dk-webtilg.pdf).
5. Energistyrelsen, *Dit elforbrug*, da, Sept. 2015, (2021; https://sparenergi.dk/forbruger/el/dit-elforbrug).
6. U. N. F. C. on Climate Change, *Paris Agreement*, en, Text, Nov. 2016, (2021; https://ec.europa.eu/clima/policies/international/negotiations/paris_en).
7. S. Holly, J. Randal, C. Susan, B. Daniel, *Graphic: The relentless rise of carbon dioxide*, (2021; https://climate.nasa.gov/climate_resources/24/graphic-the-relentless-rise-of-carbon-dioxide).
8. A. H. M. Wanscher, *Fakta om Danmarks udledning af drivhusgasser samt energiforbrug (opdateret)*, da, (2021; https://www.dst.dk/da/Statistik/bagtal/2018/2018-12-06-fakta-om-danmarks-udledning-af-drivhusgasser-samt-energiforbrug).
9. *Fakta om vindenergi*, da, Apr. 2016, (2021; https://ens.dk/ansvarsomraader/vindenergi/fakta-om-vindenergi).
10. *Signe Horn Rosted – Vice President, Business and Markets, Energinet Electricity System Operator – Energinet | LinkedIn*, da, (2021; https://dk.linkedin.com/in/signehrosted).
11. *Brug for fintælling: 2020 i uhyre tæt opløb med 2019 om dansk vindrekord*, da, (2021; https://energinet.dk:443/Om-nyheder/Nyheder/2021/01/03/Brug-for-fintaelling-2020-i-uhyre-taet-oploeb-med-2019-om-dansk-vindrekord).
12. *smart home definition - Google-search*, en, (2021; https://www.google.com/search?q=smart+home+definition&oq=smart+home+definition&aqs=chrome..69i57.7012j0j1&sourceid=chrome&ie=UTF-8).
13. A. Easton, *What is a Smart Hub? | Home Automation Explained*, en-US, Section: Uncategorized, Nov. 2020, (2021; https://www.iotashome.com/what-is-a-smart-hub/).
14. L. De Silva, C. Morikawa, I. Petra, *Engineering Applications of Artificial Intelligence* **25**, 1313–1321 (Oct. 2012).
15. *MariCare - Fall prevention system, fall detection systems for nursing homes, senior homes, private homes, rehabilitation centers, hospitals - MariCare*, (2021; https://maricare.com/da/).
16. T. Agnes, N. Monika Bille, *Publikation: It-anvendelse i befolkningen 2020*, da, 2020, (2021; https://www.dst.dk/da/Statistik/Publikationer/VisPub?cid=29450).
17. *Smart home*, en, (2021; https://www.statista.com/study/27165/smart-homes-statista-dossier/).

18. M. Singleton, *The smart home is getting cheaper*, en, Sept. 2017, (2021; https://www.theverge.com/circuitbreaker/2017/9/28/16361394/smart-home-getting-cheaper-nest-august-ikea).
19. Energistyrelsen, *2020 Basisfremskrivning*, da, June 2020, (https://ens.dk/sites/ens.dk/files/Basisfremskrivning/basisfremskrivning_2020-webtilg.pdf).
20. U. Nations, *13. Climate Action*, 2015, (2021; https://www.theexplorer.no/goals/climate-action/).
21. C. for Disease Control, Prevention, *Climate Effects on Health*, 2021, (2021; https://www.cdc.gov/climateandhealth/effects/default.htm#:~:text=The%5C%20health%5C%20effects%5C%20of%5C%20these,and%5C%20threats%5C%20to%5C%20mental%5C%20health.).
22. Seas-nve, *Seas-nve Live Price Feed*, 2021, (2021; https://energi.seas-nve.dk/kundeservice/aftaler-og-priser/timepris/).
23. AUra, *AURA Live Price Feed*, 2021, (2021; https://www.aura.dk/stroem/aura-flexel/timepris/).
24. Energinet, *Energysystem Right Now*, 2021, (2021; https://energinet.dk/energisystem_fullscreen).
25. *What is User Story?*, (2021; https://www.visual-paradigm.com/guide/agile-software-development/what-is-user-story/).
26. *Waterfall model*, en, Page Version ID: 1008907455, Feb. 2021, (2021; https://en.wikipedia.org/w/index.php?title=Waterfall_model&oldid=1008907455).
27. *SDLC - Waterfall Model - Tutorialspoint*, (2021; https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm).
28. *Software Engineering | SDLC V-Model - GeeksforGeeks*, (2021; https://www.geeksforgeeks.org/software-engineering-sdlc-v-model/).
29. *Agile Model (Software Engineering) - javatpoint*, en, (2021; https://www.javatpoint.com/software-engineering-agile-model).
30. *What is MoSCoW Prioritization? | Overview of the MoSCoW Method*, en-US, (2021; https://www.productplan.com/glossary/moscow-prioritization/).
31. *Primer: Understanding Software and System Architecture*, en-US, Dec. 2019, (2021; https://thenewstack.io/primer-understanding-software-and-system-architecture/).
32. *Online Mockup, Wireframe & UI Prototyping Tool · Moqups*, en, (2021; https://moqups.com).
33. *How to Build and Structure a Node.js MVC Application - SitePoint*, en, (2021; https://www.sitepoint.com/node-js-mvc-application/).
34. *Express Tutorial Part 4: Routes and controllers - Learn web development | MDN*, (2021; https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs/routes).
35. *Zigbee Smart Outlet Energy Metering 15A*, en-US, (2021; https://smartenit.com/shop/zbmskt1/).
36. *Loxone - Create Automation | Home & Building Automation*, en-GB, (2021; https://www.loxone.com/enen/).
37. *Yonomi — Bring your home to life.™*, en-US, (2021; https://www.yonomi.co).
38. *Generac Power Systems - PWRview*, (2021; https://www.generac.com/all-products/clean-energy/pwrview).
39. *Efergy Home Energy Monitors: Electricity Usage Power Monitor*, (2021; https://efergy.com/).
40. *Positive energy systems*, en-US, (2021; https://positiveenergysystems.com/).

41. *Swell*, en, (2021; https://www.swellenergy.com/homeenergysystem/).
42. *Solar Analytics*, en-AU, (2021; https://www.solaranalytics.com/au).
43. *carbonTRACK - Smart Energy Management System*, (2021; https://carbontrack.com.au/).
44. *Constellation*, (2021; https://www.constellation.com/solutions/for-your-home/home-services/constellation-connect.html).
45. *Bosch Energy Manager*, en, (2021; https://www.bosch.com/stories/smart-home-energy-management-system/).
46. *E.ON*, (2021; https://www.eon.com/en/private-customers/home-energy-management.html).
47. *ems3.com*, en-US, (2021; http://ems3.com/).
48. *Honda Smart Home US | Introducing: the Honda Smart Home*, (2021; https://www.hondasmarthome.com/post/72692414664/welcome-to-the-honda-smart-home-us-hondas-vision).
49. *Square D Wiser Energy Smart Home Monitor Solar Edition-WISEREMPV*, en, (2021; https://www.homedepot.com/p/Square-D-Wiser-Energy-Smart-Home-Monitor-Solar-Edition-WISEREMPV/309169207).
50. *Sense: Track energy use in real time to make your home more energy efficient.* en-US, (2021; https://sense.com/).
51. *Buy - Spring 2021*, en-US, (2021; https://sense.com/buy/).
52. P. E. King, *641 days with the Sense Home Energy Monitor*, en-US, Feb. 2019, (2021; https://pocketables.com/2019/02/641-days-with-the-sense-home-energy-monitor.html).
53. *Living with the Sense Energy Monitor: Frustrating But Helpful Overall*, en-US, Apr. 2019, (2021; https://restechtoday.com/living-with-the-sense-energy-monitor-frustrating-but-helpful-overall/).
54. *Smappee home*, en-US, (2021; https://www.smappee.com/).
55. *Smappee Energy Monitor Pakketilbud hos SmarterLife.dk*, da-DK, (2021; https://www.smarterlife.dk/produkter/smappee-energy-monitor-pakketilbud/).
56. *Smappee Infinity Stor Startpakke - Fleksibel løsning til energimonitorering*, da-DK, (2021; https://www.smarterlife.dk/produkter/smappee-infinity-stor-startpakke/).
57. *Analyse and use energy data with the Smappee Dashboard.* en-US, Mar. 2020, (2021; https://www.smappee.com/blog/smappee-dashboard-energy-data/).
58. *Eve Energy | evehome.com*, en, (2021; https://www.evehome.com/en/eve-energy).
59. *Home automation without the need for building work in your new or existing home - Discover ONE*, en-US, (2021; https://onesmartcontrol.com/en/).
60. *Dit hjem, dine regler*, da-DK, (2021; https://homey.app/da-dk/homey/).
61. Homey, *We heard you like it big.* en-US, (2021; https://homey.app/en-us/blog/homey-app-for-tablets/).
62. Homey, *A technical introduction to Homey*, en-GB, (2021; https://homey.app/en-gb/blog/a-technical-introduction-homey/).
63. *Smart Electrical Panel | Responsive Energy Management | Lumin*, en-US, (2021; https://www.luminsmart.com/platform/smart-electrical-panel/).

64. *Lumin Smart Panel App Monitoring | Energy Management*, en-US, (2021; https://www.luminsmart.com/quick-tips/).
65. *Smart Electrical Panel for the Home - Residential Solar | Lumin*, en-US, July 2020, (2021; https://www.luminsmart.com/homeowners/).
66. *Meet Spiir. Your money's best friend. - Spiir*, da, (2021; https://www.spiir.com/).
67. *What is Card-Based Design and Should you be Using it? — Medialoot*, en, (2021; https://medialoot.com/blog/what-is-card-based-design-and-should-you-be-using-it/).
68. J. T. Mark Otto, *Cards*, en, (2021; https://getbootstrap.com/docs/5.0/components/card/).
69. *Cards*, en, (2021; https://material.io/components/cards).
70. *Card*, en, (2021; https://bulma.io/documentation/components/card/).
71. M. Matera, F. Rizzo, G. T. Carughi, in *Web Engineering*, ed. by E. Mendes, N. Mosley (Springer Berlin Heidelberg, Berlin, Heidelberg, 2006), pp. 143–180, ISBN: 978-3-540-28218-1, (https://doi.org/10.1007/3-540-28218-1_5).
72. E. Holder, *Americans struggle with Graphs.* en, Feb. 2021, (2021; https://towardsdatascience.com/numeracy-and-graph-literacy-in-the-united-states-ea2a11251739).
73. OECD, *The Survey of Adult Skills*, p. 128, (https://www.oecd-ilibrary.org/content/publication/9789264258075-en).
74. *Web Design and Branding Agency, Calgary | Tiller Digital*, en, (2021; https://tillerdigital.com/).
75. *12 web design best practices for 2021 | Tiller Digital*, en, Jan. 2021, (2021; https://tillerdigital.com/blog/12-web-design-best-practices-for-2021/).
76. Energinet, *TSO Electricity*, en, (2021; https://www.energidataservice.dk/tso-electricity).
77. S. Prabhakaran, *Time series forecasting*, 2021, (2021; https://www.machineleaarrningplus.com/time-series/arima-model-time-series-forecasting-python/).
78. Wikipedia, *stationary process*, 2021, (2021; https://en.wikipedia.org/wiki/Stationary_process).
79. Wikipedia, *White Noise*, 2021, (2021; %https://en.wikipedia.org/wiki/White_noise).
80. Mathworld, *Topological Transitive*, 2021, (2021; https://mathworld.wolfram.com/TopologicallyTransitive.html#:~:text=A%20function%20is%20topologically%20transitive,together%20in%20one%20localized%20clump.).
81. Goodmath, *Dense Periodic Orbit*, 2010, (2021; https://scienceblogs.com/goodmath/2010/01/26/more-about-dense-periodic-orbi).
82. Wikipedia, *Chaos Theory*, 2021, (2021; https://en.wikipedia.org/wiki/Chaos_theory).
83. *Mongoose ODM v5.12.6*, (2021; https://mongoosejs.com/).
84. *bcrypt*, en, (2021; https://www.npmjs.com/package/bcrypt).
85. *Blowfish (cipher)*, en, Page Version ID: 1003368329, Jan. 2021, (2021; https://en.wikipedia.org/w/index.php?title=Blowfish_(cipher)&oldid=1003368329).
86. *validatorjs/validator.js*, original-date: 2010-10-06T06:58:48Z, May 2021, (2021; https://github.com/validatorjs/validator.js).
87. *Automated Functional Testing for Web & Mobile Apps | Perfecto*, (2021; https://www.perfecto.io/functional-testing-web-mobile-apps).

88. *Unit Testing Tutorial: What is, Types, Tools & Test EXAMPLE*, (2021; https://www.guru99.com/unit-testing-guide.html).
89. *Component vs (UI) Integration vs E2E tests*, en, (2021; https://dev.to/noriste/component-vs-ui-integration-vs-e2e-tests-3i0d).
90. *What is Test Scenario? Template with Examples*, (2021; https://www.guru99.com/test-scenario.html).
91. *Performance Testing Tutorial: What is, Types, Metrics & Example*, (2021; https://www.guru99.com/performance-testing.html#7).
92. *END-To-END Testing Tutorial: What is E2E Testing with Example*, (2021; https://www.guru99.com/end-to-end-testing.html).
93. *Jest - Delightful JavaScript Testing*, en, (2021; https://jestjs.io/).
94. *OneTest Product | HCL Technologies*, en, (2021; https://www.hcltech.com/software/onetest).
95. *HCL OneTest | HCL Software*, en, (2021; https://www.hcltechsw.com/onetest).
96. *HCL OneTest Perfomance - HCL Software*, en, (2021; https://www.hcltechsw.com/onetest/offerings/onetest-performance).
97. *Running Tests - Appium*, (2021; http://appium.io/docs/en/writing-running-appium/running-tests/).
98. *Cucumber Open - Get Started with BDD Today | Cucumber*, (2021; https://cucumber.io/).
99. *Request Matching*, en, May 2021, (2021; http://wiremock.org/docs/request-matching/).
100. *Open Source JavaScript Test Runner*, en, (2021; https://www.cypress.io/features).
101. *What is End-to-End (E2E) Testing? | All You Need to Know*, (2021; https://www.katalon.com/resources-center/blog/end-to-end-e2e-testing/).
102. R. A. Virzi, en, *Human Factors* **34**, Publisher: SAGE Publications Inc, 457–468, ISSN: 0018-7208, (2021; https://doi.org/10.1177/001872089203400407) (Aug. 1992).
103. Energinet, *About*, en, (2021; https://www.energidataservice.dk/about/).
104. *Klima- Energi- og forsyningsministeriet*, da, (2021; https://kefm.dk/).
105. *Coinbase Pro Status*, (2021; https://status.pro.coinbase.com/).
106. *About the energy label and ecodesign | European Commission*, (2021; https://ec.europa.eu/info/energy-climate-change-environment/standards-tools-and-labels/products-labelling-rules-and-requirements/energy-label-and-ecodesign/about_en).
107. J. Holgaard, T. Ryberg, N. Stegeager, D. Stentoft, A. Thomassen, *PBL: Problembaseret læring og projektarbejde ved de videregående uddannelser*, Dansk (Samfundslitteratur, ed. 2, 2020), ISBN: 9788759333969.
108. *FusionSmart*, (2021; https://www.fusioncharts.com/demos/dashboards/smart-energy-monitoring-dashboard/#).

13 Appendices

A Definitions

- Sustainable/renewable/green energy
  In this report sustainable and renewable energy is combined to mean the same thing. Sustainable/renewable/ energy is defined as the following: Solar energy, wind, hydro power, geothermal energy, biomass energy, biogas energy and heat pumps.
- The terms appliances and devices are used interchangeably, defined to mean the same thing in this rapport: A thing made or adapted for a particular purpose.

B Sustainable electricity consumption in homes - Survey

All the responses to our sustainable electricity consumption survey can be found in the sheet below.
https://docs.google.com/spreadsheets/d/1RdLGKdRFQkxyIfhq3EcnRv84pHmT_c0nk4v6HXdZeeI/edit?usp=sharing
The figures used in Section 2.5.1 are made by combining the Danish and English results.

C Discarded solutions

FusionSmart Dashboard

The FusionSmart dashboard is a design mock-up of what could be a dashboard for monitoring smart home appliances. It consists of 5 dashboard pages, to monitor everything from the household's carbon emissions to predicting the households energy-consumption over a few days. The dashboard is made with a modern minimalist design that is packed with important information. The dashboard is only a design choice, and is not a full solution to consumers. The FusionSmart company makes good-looking dashboards from home automation to finance dashboards and everything in between. Some of key charts on the dashboard for smart home appliances includes: *Cost prediction, Active appliances, Carbon footprint, Usage-by-rooms, Energy saving tips* [108].
The fusionsmart dashboard is not on the list, as it is not a commercial product like the others and was not in on itself an energy monitoring product. With that said, it gave inspiration to what a dashboard could look like.

Zigbee Smart Outlet

ZBMSKT1 is the shorted name of a smart outlet using the Zigbee communication protocol. With an app on your smartphone or webbrowser, you will be able to monitor the energy consumption on individual devices. The outlet is then remote controlled, and can turn devices ON/OFF that will stop devices from consuming energy when they are not in use, also known as Stand by power. The outlet can also be scheduled to turn ON/OFF depending on the time of day where the energy cost is lowest.
If every device was connected to this smart plug then it would be possible to monitor the entire house. The downsides would be the cost being 45$ pr. outlet, and that it would not be possible to monitor devices that are not wall plugged such as your stove and possibly oven [35].
The ZigBee smart outlet was not on the list, since it had a lot of overlap in features with the other smart plugs already described.

Loxone

https://www.loxone.com/enen/products/apps
Loxone is a complete home automation solution letting the user monitor and control just about everything in the house from lighting, heating and energy consumption to solar panels and pool management. The solution comes its own app which is similar in functionality to the apps provided by the other solutions we've looked at. We chose not to include this solution since it didn't seem to provide any particular functionality that would make it stand out from the rest.

Yonomi

https://www.yonomi.co/yonomi-app
The Yonomi app seeks to integrate all the different smart home apps a user may have into one place from which they can be controlled such that the user doesn't need to access every specific app connected to different brands of smart home devices. While this functionality is interesting in its own right, the app itself does not provide the energy management capabilities we were looking for, and therefore the app did not make it onto our list.

D Sequence diagrams

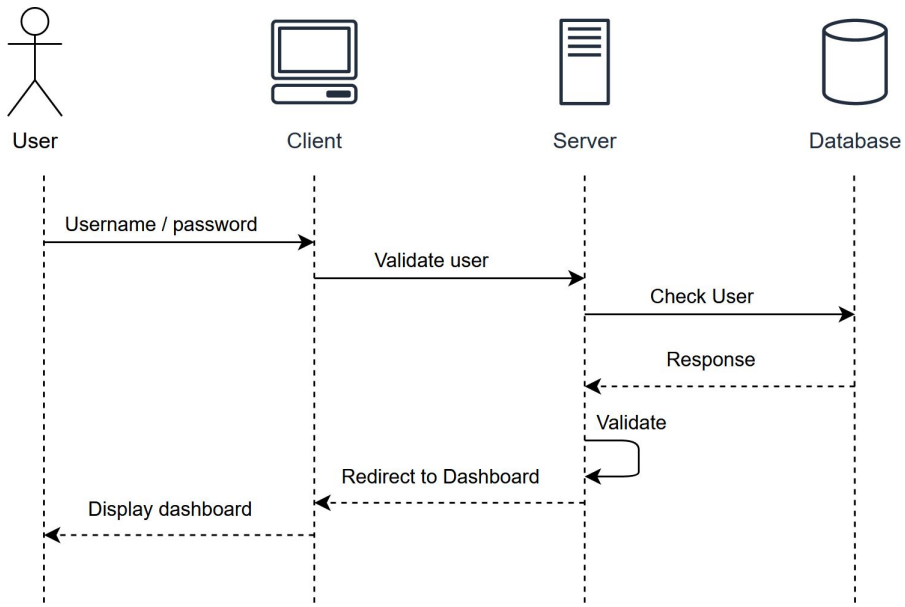


Figure 57: Sequence diagram for login

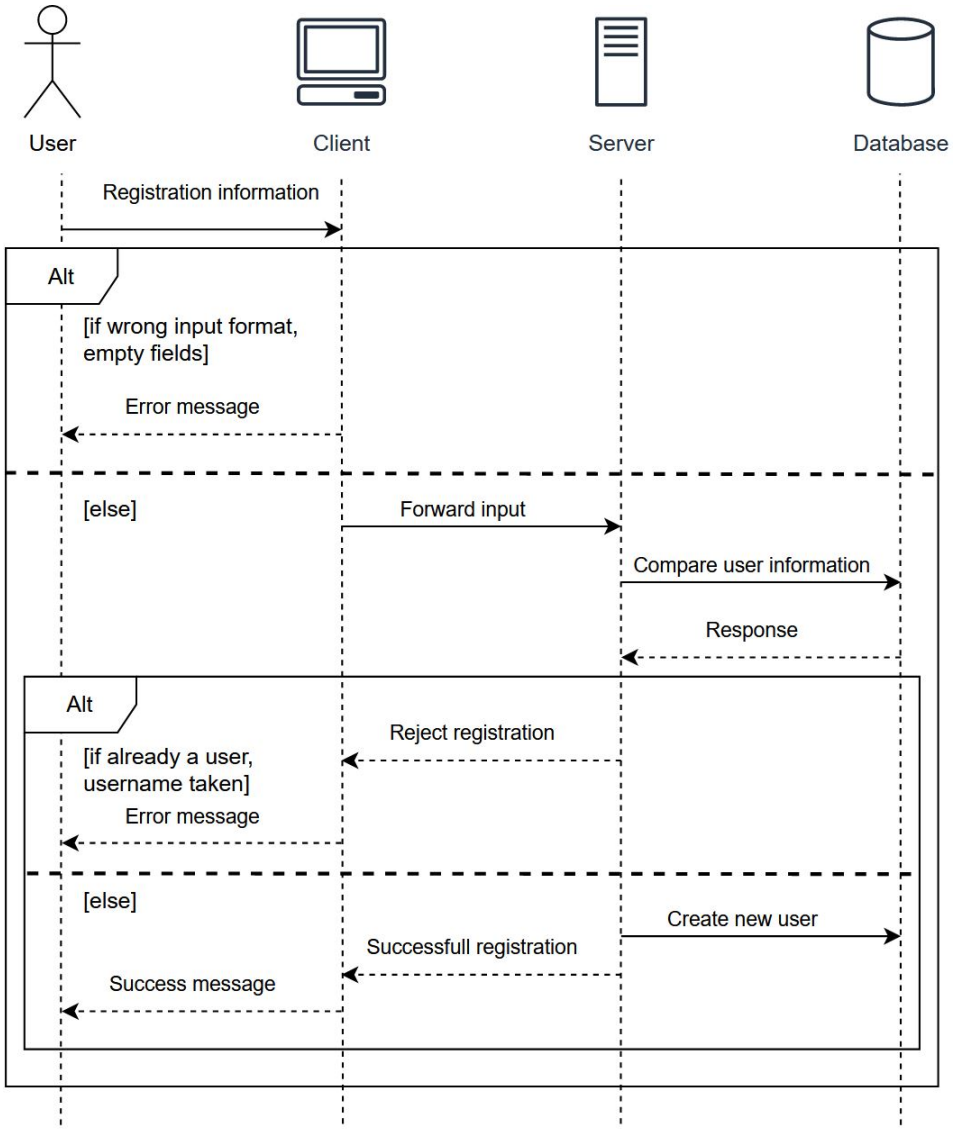


Figure 58: Sequence diagram for registration

E Group contact

Arbejdsdagen

§1. Hver arbejdsdag startes med scrum-statusmøde, en dagsorden og valg af referent.

§2. Projektmål og -struktur revideres hver arbejdsdag.

§3. Alle deltager i alle samtaler om projektrelevante emner.

§4. Små opgaver kan uddelegeres til enkelte medlemmer, men alle skal have det fulde overblik.

§5. Visuelle hjælpemidler benyttes i videst mulige omfang – eksempelvis benyttes whiteboardet ved alle gruppemøder og -diskussioner.

Arbejdsfordeling

§1. Vi holder scrum møde i sub-teams hvor der diskuteres arbejdsopgaver for alle teams.

§2. Hver Onsdag holdes Status for hvordan teams har arbejdet med opgaverne.

§3. Hver Fredag laves der nye teams.

Status

§1. Status starter efter frokost pause hver Onsdag

§2. Status mødets længde skal ikke være meget længere end 30 min

§3. Der skal opsamles hvad der er blevet lavet i hver team

§4. Der skal aftales en dagsorden for torsdag med Sokol

Tidsplanlægning

Alt information omkring mødetider skal være lagt ind på Google Kalenderen, så alle er klar over hvornår vi i gruppen mødes.

Programmerings skik og arbejdsform

§1. Der skal laves en funktionsbeskrivelse der forklarer hvad funktionen tager af input og hvad den gør'.

§2. Tænk over gode logiske variabel og funktionsnavne.

§3. Vi benytter camelCase navngivning i Javascript og snake_case til C

§4. Konstanter defineres i ALL-CAPS.

§5. Lav små meningsfyldte commits, med en beskrivelse af hvad der er blevet ændret.

§6. Sørg for at køre koden inden koden bliver committet for at undgå fejl.

§7. Rapportens dele skrives på engelsk (så Sokol kan forstå).

Faglige forventninger

§1. Alle medlemmer skal vide, hvad gruppen laver.

§2. Faglige diskussioner holdes indenfor emnet.

§3. Visuelle forklaringer benyttes om muligt.

§4. Alle skal deltage nogenlunde ligeligt i såvel programmering som rapportskrivning.

§5. Aftalt hjemmearbejde skal overholdes.

§6. Produktet skal gennemgås og afleveres gennemarbejdet.

§7. I udgangspunktet forventes timer brugt svarende til normeringen (ca. 20 timer om ugen for P2), og denne forventning kan om nødvendigt øges nær deadline.

Sociale forventninger

§1. Gruppens primære kontakt foregår gennem Messenger.

§2. Alle skal tjekke Messenger (mindst) dagligt.

§3. Gruppen mødes på campus alle hverdage, med mindre andet er aftalt.

§4. Alle møder på det aftalte tidspunkt, og der gives besked, hvis man er mere end et kvarter forsinket.

§5. Alle lytter til og respekterer hinandens person og meninger.

§6. Konflikter og uenighed løses ved demokratisk afstemning i gruppen.

Fravær

§1. Alle skal i udgangspunktet møde til alle forelæsninger og møder.

§2. Bliver man syg, eller har man anden god grund til fravær, gives der besked i rimelig tid.

Konsekvenser

§1. Gruppen kan når som helst stemme om, hvorvidt et gruppemedlem skal tildeles en advarsel.

§2. Uddeling af advarsel kræver almindeligt flertal blandt gruppens øvrige medlemmer.

§3. Er et gruppemedlem tildelt en advarsel, kan gruppen efter yderligere overtrædelser når som helst stemme om, hvorvidt gruppemedlemmet skal ekskluderes.

§4. Ekskludering kræver enstemmighed blandt gruppens øvrige medlemmer.

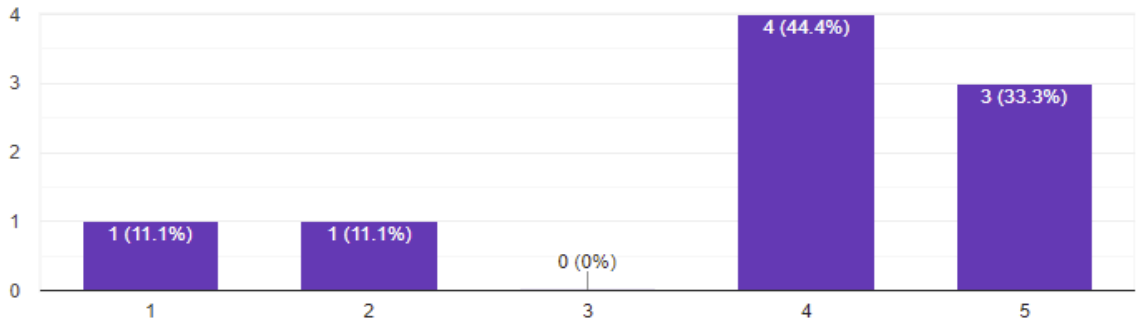
F User Test Survey Responses

The following 7 images show the response we received from our user test survey.

Dashboard Overview

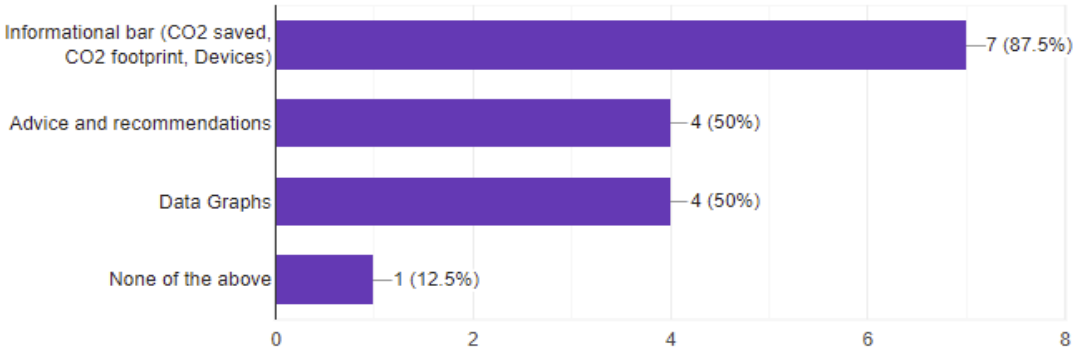
Does the dashboard overview provide a clear structure and indication of where to find the information you are looking for?

9 responses



Which of these informational tools did you consider useful?

8 responses



How useful was the Informational bar?

9 responses

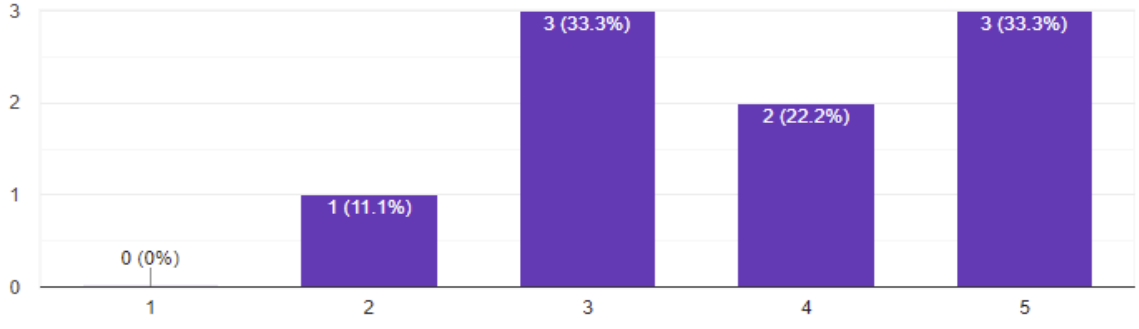


Figure 59: User test survey responses part 1

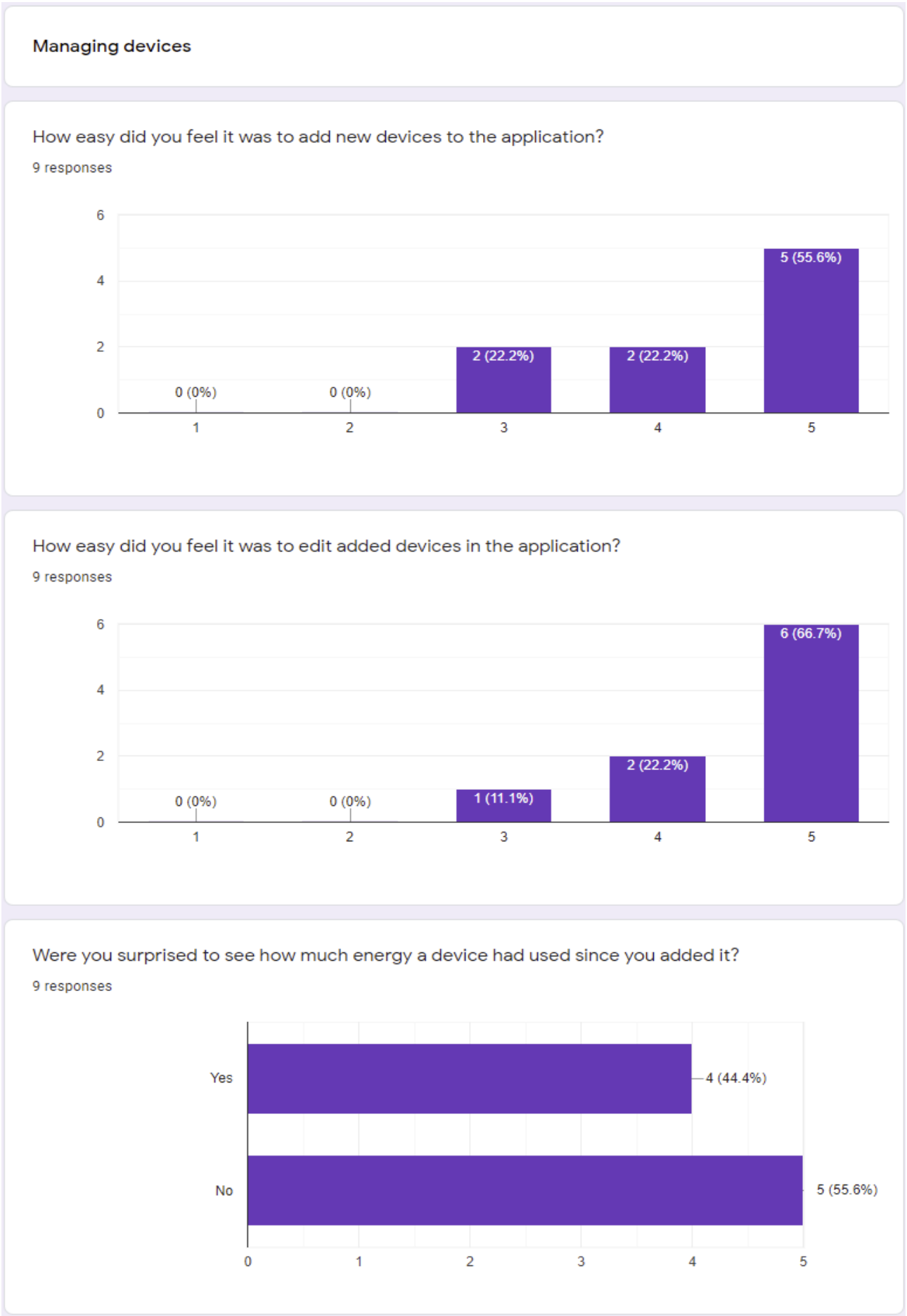
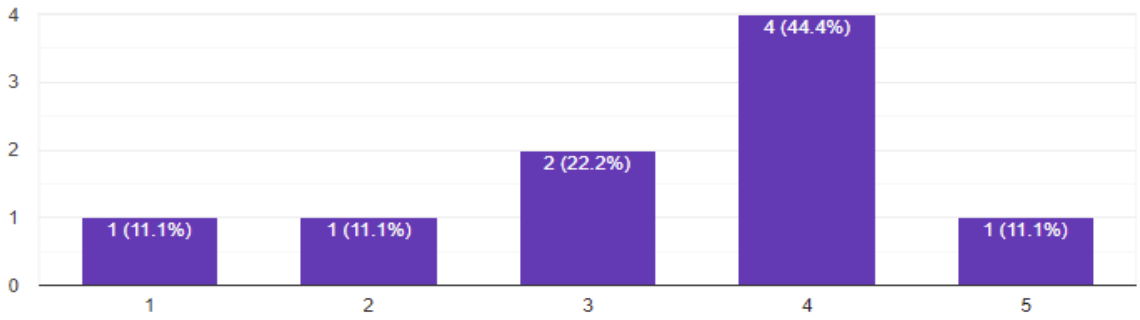


Figure 60: User test survey responses part 2

How useful was each device's own energy graph?

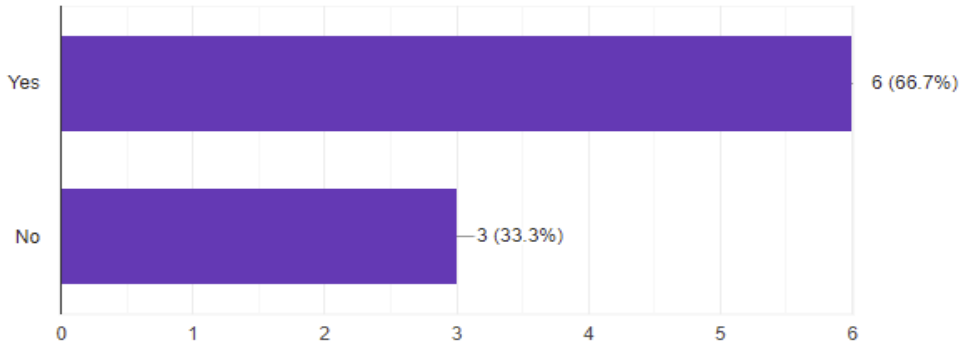
9 responses



Advices and recommendations

Recommendation cards: Would you be willing follow this advice and wait the recommended time, to save the environment from unneeded extra CO2?

9 responses



Event cards: Did this type of advice help you to better know when the energy was coming from more climate-friendly sources?

9 responses

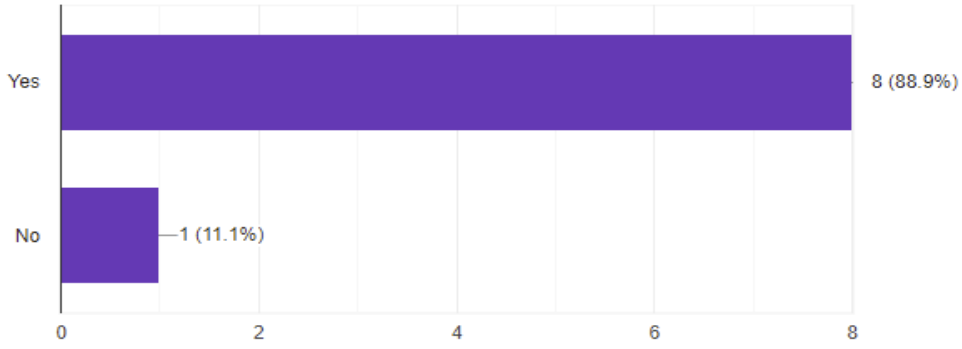
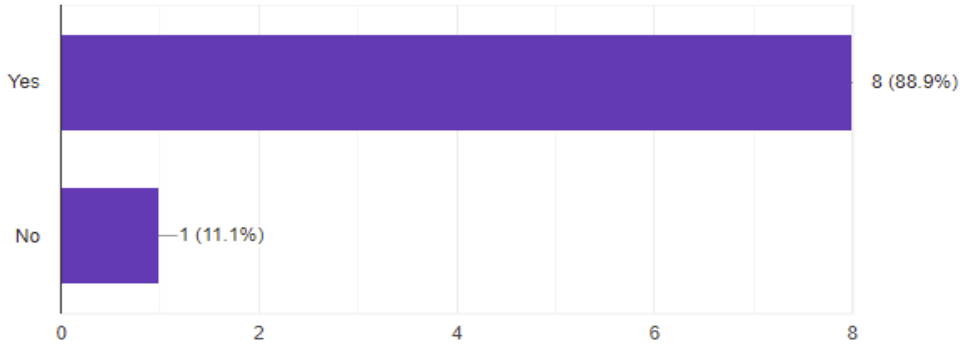


Figure 61: User test survey responses part 3

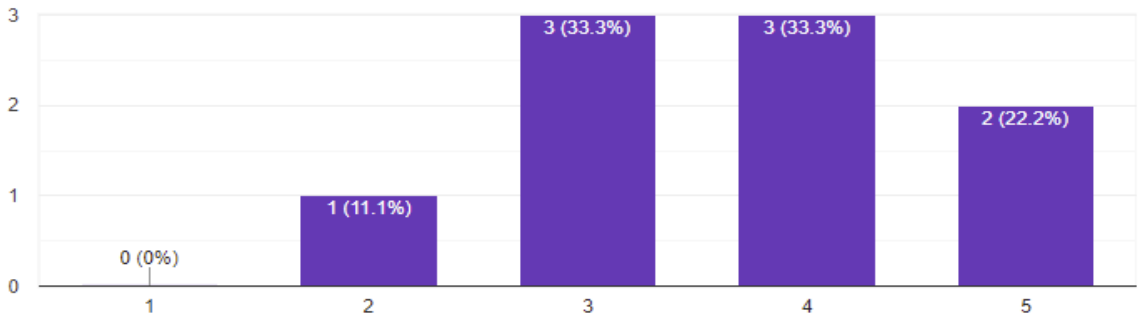
Status cards: Did this type of advice help you to better know how well you actually "performed" against your expectations to lower you carbon footprint?

9 responses



How useful did you find the Advice cards?

9 responses



Data and graphs

Danish CO2 emissions: Did this type of graph help you, as a user, to better understand that the amount of CO2 produced is changing at any given time?

9 responses

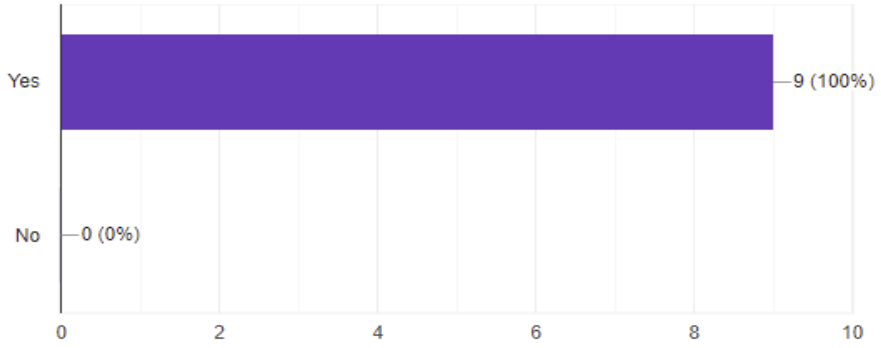
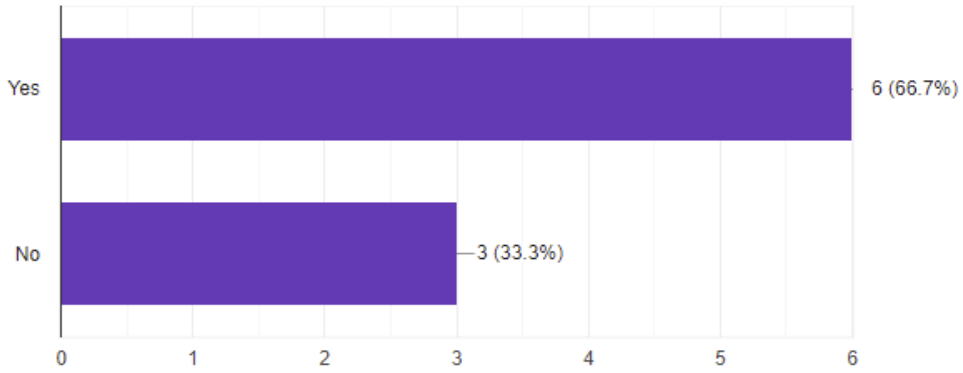


Figure 62: User test survey responses part 4

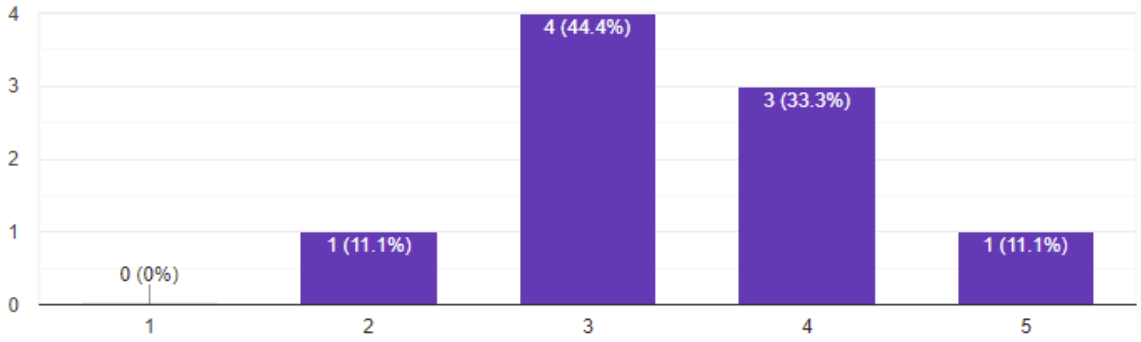
Danish CO2 emissions: After watching this graph, are you aware of when to use electrical devices in a sustainable manner?

9 responses



How useful was the CO2 graph?

9 responses



Danish CO2 forecast: even though the forecast is not 100% accurate, did this visualization help you better plan your energy usage?

9 responses

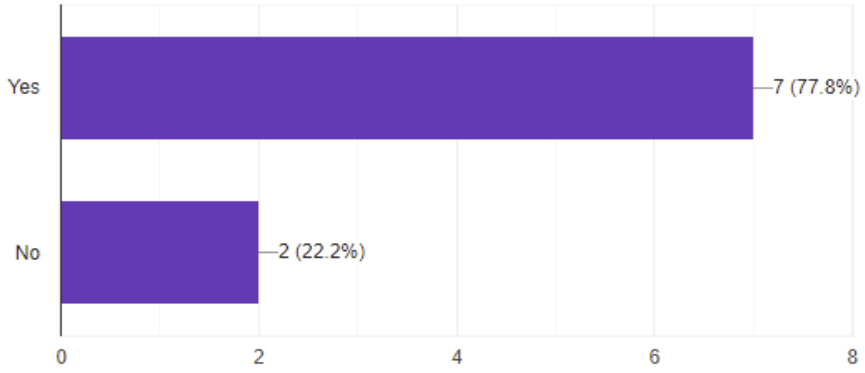
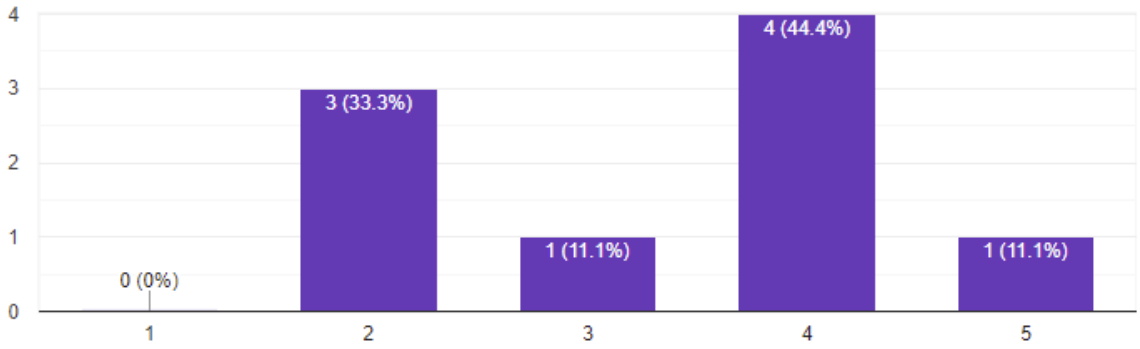


Figure 63: User test survey responses part 5

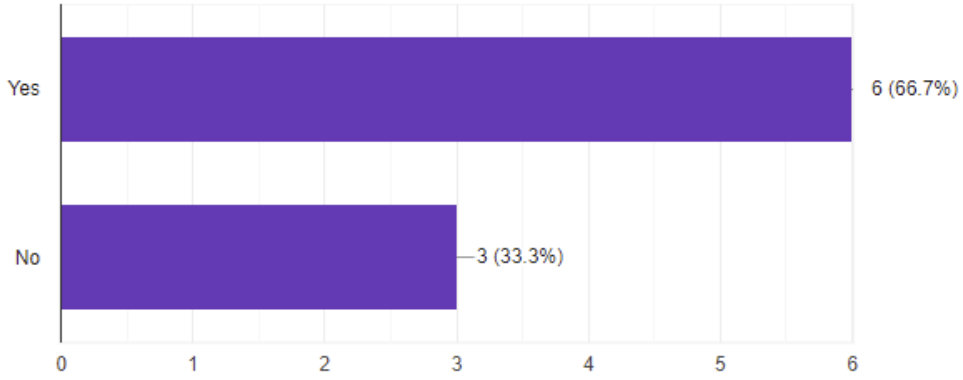
How useful was the Forecast graph?

9 responses



Danish Green Energy production: Were you, as a user, surprised about how much each carbon-neutral sector produced?

9 responses



How useful was the Green energy production graph?

9 responses

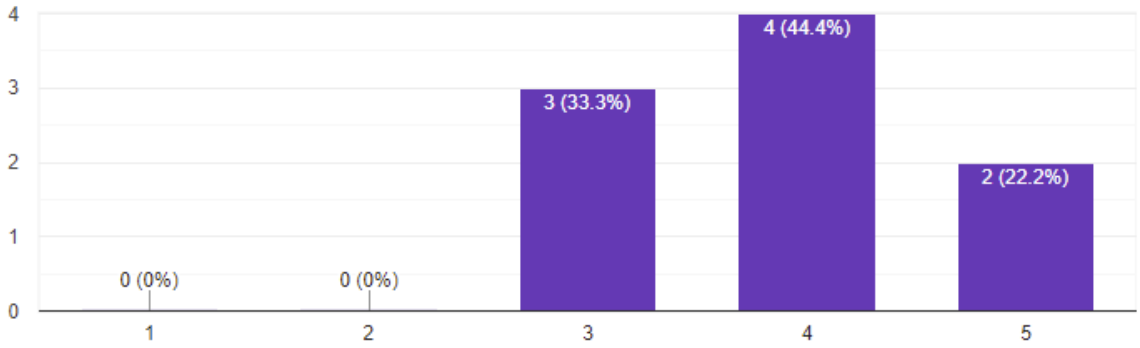
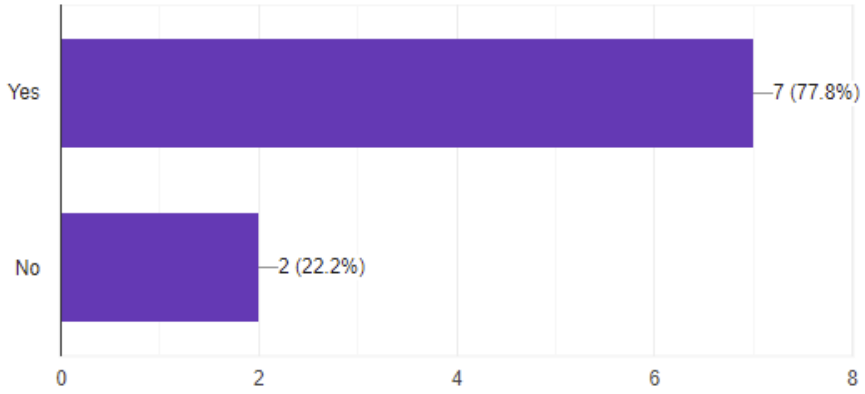


Figure 64: User test survey responses part 6

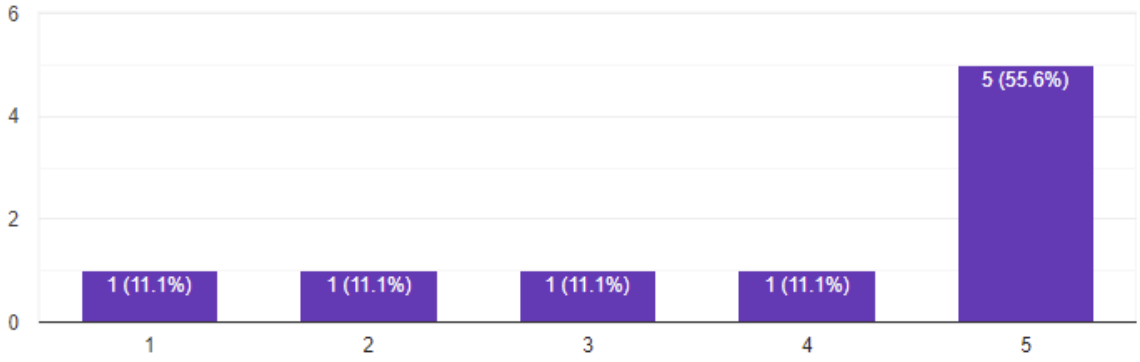
My Home Energy Consumption: Did you acquire any insights into your electricity usage by looking at this graph?

9 responses



How useful was the Home energy Consumption graph?

9 responses



Which method of receiving information do you prefer?

9 responses

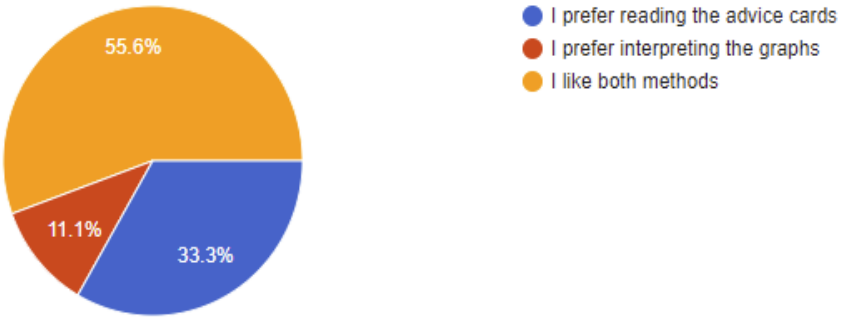


Figure 65: User test survey responses part 7