Diese Arbeit wurde vorgelegt am
Lehr- und Forschungsgebiet Theorie der hybriden Systeme

Interaktive Visualisierung des Schlagschattens von Windparks

Interactive Visualization of Shadow Cast for Wind Farms

Bachelorarbeit
Informatik

September 2021

Vorgelegt von Presented by	Mostafa Elgayar Kühlwetterstraße 8 52072 Aachen Matrikelnummer: 390501 mostafa.elgayar@rwth-aachen.de
Erstprüfer First examiner	Prof. Dr. rer. nat. Erika Ábrahám Lehr- und Forschungsgebiet: Theorie der hybriden Systeme RWTH Aachen University
Zweitprüfer Second examiner	Prof. Dr. rer. nat. Horst Lichter Research Group Software Construction RWTH Aachen University
Betreuer Supervisor	Dr. rer. nat. Pascal Richter Lehr- und Forschungsgebiet: Theorie der hybriden Systeme RWTH Aachen University

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich diese Bachelorarbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Stellen meiner Arbeit, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, habe ich in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Dasselbe gilt sinngemäß für Tabellen und Abbildungen. Diese Arbeit hat in dieser oder einer ähnlichen Form noch nicht im Rahmen einer anderen Prüfung vorgelegen.

Aachen, im September 2021

Mostafa Elgayar

Contents

1 Introduction

It is no secret that climate change is a real issue that our generation is facing, and the consequences are experienced by many people around the globe, whether in the form of floods, earthquakes, storms, wildfires, global warming, etc... These drastic changes to our environment are due to the heavy industrialization behaviours that we've been practising in the last century which have increased in exponential levels, harming the planet in the process. These behaviours result in excessive production of greenhouse gases such as carbon dioxide. The main sources for these gases stem from the burning of fossil fuels such as coal, oil, and natural gas, which are what drives the industry goals of humans. The problem that comes with the increase of concentration of greenhouse gases in the earth's atmosphere is the trapping of heat-producing the greenhouse effect. This isn't an easy problem to deal with, especially with deforestation which makes it less possible to decrease such concentration of greenhouse gases from our atmosphere.

A large percentage of the produced greenhouse gases come from electricity which is what powers almost everything in our daily lives, and life can simply not be imagined without its existence these days. Electricity is produced with the burning of fossil fuels, however, in the last few decades, scientific progress has been made into discovering new, cleaner ways of producing electricity.

To reduce the reliance on fossil fuels, which not only are limited in quantity but also very polluting and is the reason behind the disasters, we had to shift towards renewable sources of energy and increase the reliance on them. Such sources are present daily, like the sun and wind, they are unlimited in their availability which means they are highly sustainable for the future generations to come as well as to the well-being of the planet. Such renewable energy projects are being constructed and planned everywhere around the world these days, with the growing awareness of their importance and necessity.

Although projects like solar energy plants and wind farms have a great positive effect on the climate and the environment, some obstacles stand in the way of implementing them smoothly. In this work, we look closely towards optimizing the planning of wind turbines and the problems that come along with them, and we try to develop countermeasures and strategies to solve them.

Wind energy has an increasing share of electricity generation in Germany. In 2020, wind energy had again the largest share in German electricity production with 131.9 TWh, ahead of nuclear energy and brown coal, see Figure 1. In 2020 wind energy had a total share of about 27% in the German electricity production, see Figure 2 [15].

The main issues arise from the communities and individuals living next to wind farms. There is a notable rise in the resistance of their constructions, and the reasons vary between concerns on the impact on the landscape, the removal of forests to free up space for construction, noise from the turbines, shadows, and the impact the turbines

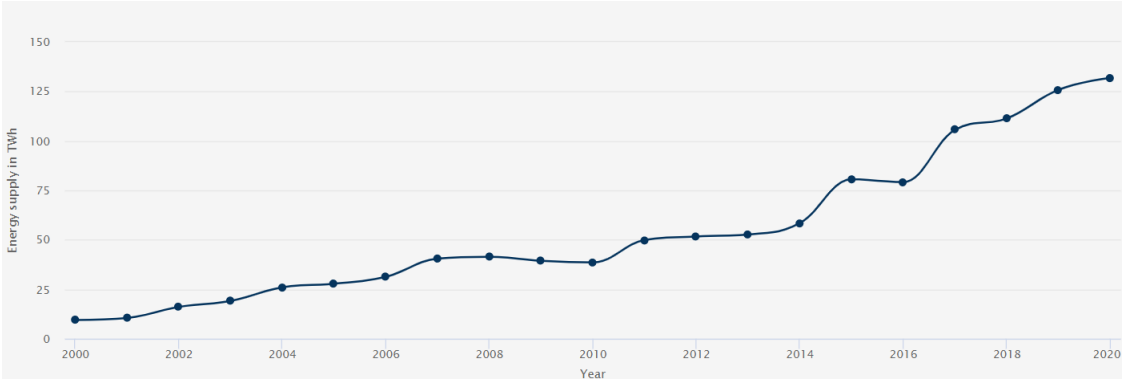

Figure 1: Gross electricity production from onshore and offshore wind energy in Germany [15].

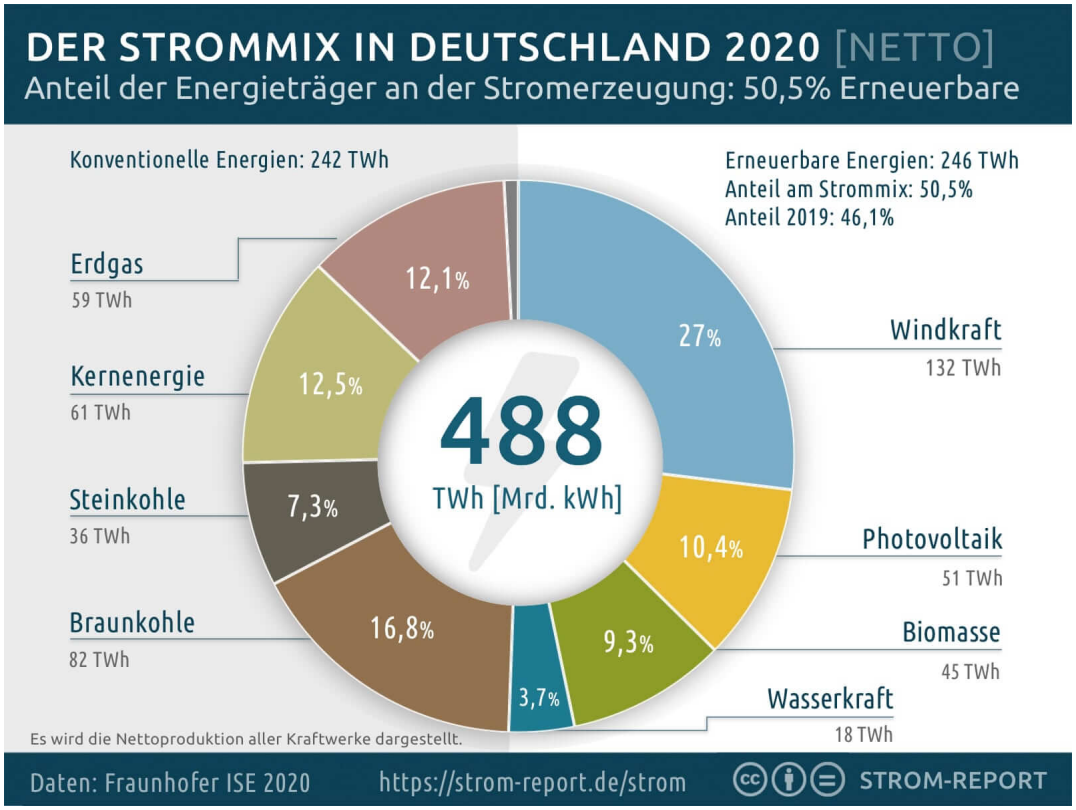

Figure 2: Electricity distribution in Germany [15].

could have on the wildlife, specifically endangered species that could be in the region. Therefore presenting transparent information and visualizations to clear the doubts of citizens is crucial to eliminate opposition in a constructive way that considers all stakeholders.

We, therefore, hypothesize that engaging citizens through a platform that shows them the construction plans beforehand, and prompting them to give their feedback, opinion on their preferable scenarios in the planning of the wind farms in terms of wind turbine positioning, for example, could increase the probability of acceptance as they would be essentially taking ownership in what will be built in their neighbourhood or city.

This will be done through visualizing the proposed positions of the turbines on a map, where they could compare it with their addresses, and see how they will be potentially personally impacted whether positively or negatively. This includes also visualizations of noise propagation, shadow casts, and virtual and augmented reality 3D models of the project. Moreover, informative and transparent statistics and numbers on the real environmental long term impact of the project construction would provide them with more motive to accept proceeding with the project, especially if shown in comparison to the current state without the wind farm's existence.

To achieve the representation of some of the technical aspects of wind farm planning and the influence they have on the neighbouring areas, we develop our algorithms that would run simulations for certain periods given certain inputs. In our case, we look closely at shadow casts of wind turbines and how we can use the outputs from these algorithms to interactively visualize them on the graphical interface of the app

Users will be able to navigate through the different possible scenarios for the optimized plans of the wind farms and give their opinion and feedback through dedicated channels designed to effectively engage them in voicing their concerns and honest observations. The results would then later be analyzed and presented to the relevant stakeholders such as investors and politicians as insights and assistance to their decision-making process.

2 State of the art

Wind farms consist of several wind turbines in proximity to each other with the goal of producing electricity in a sustainable and non-polluting way. They usually vary in size, ranging from small numbers to sometimes several hundreds covering tremendous areas. They are constructed either onshore or offshore, but for our work, we will focus on onshore as they pose inconviniences for the local citizens, and the goal is to develop solutions that would tackle these problems.

The main components of a horizontal-axis wind turbine that are visible from the ground are its tower, nacelle, and rotor, as can be seen in Figure 3. The blades capture the

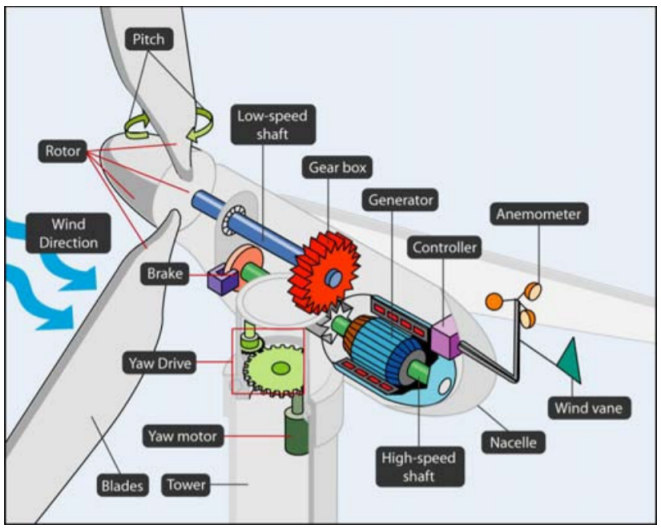

Figure 3: Wind turbine components [21].

kinetic energy in the wind and transform it into the rotational kinetic energy of the wind turbine [21].

In order to set up a wind farm, a lot of research and assessments have to be done, while considering factors like wind direction, turbine noise, environmental effects, safety, shadow impact, and others. In addition to the factors, another aspect that influences the plans is usually the standards and regulations imposed by the country where the wind farm will be constructed in, and these have to be followed as well. More information about existing standards can be found here [16].

The engineering planning of a wind farm generally includes critical decision-making regarding [5]:

1. the layout of the turbines in the wind farm
2. the number of wind turbines to be installed, and
3. the types of wind turbines to be installed.

Therefore, to assist with the planning and to yield efficient and effective productions, the usage of optimization algorithms are involved in the shape of simulations given different data points such as wind, environmental and geographical information. Our work is one piece of a bigger puzzle with the goal of achieving the most accurate representation of the planned wind farm as possible. The algorithms discussed are not the focus of this work, however their outputs are used to construct different interactive visualizations given the different wind farm configurations.

Current methodologies used to visualize wind farms exist in different levels of complexities. They also differ according to the target group. For example, visualization schemes to help engineers plan the logistics of the windfarm like proposed here [23].

There are also existing approaches that utilize virtual reality and augmented reality, as well as geographic information systems (GIS) that help in large construction construction sites [8] where it isn't very easy to navigate on-site.

Several projects were developed to visualize wind farms for educational and informative purposes which is similar to what we our developing, an example for that can be found here: [7]. These projects in many occasions involve virtual and augmented reality technologies, as well as making use of GPS functionalities to visualize the wind turbines in a way that is realistic to users. Research was also conducted to investigate the public's acceptance of wind farms using web-based simulations [2]. The paper concluded that using simulations is an effective way to inform the public about the ins and outs of the wind farm. It also suggested the effectiveness of using two-way flow of information between users and the stakeholders planning the project.

In our work, we further extend the current progress in visualizing simulations of wind farms using modern technologies, with an extra focus on user experience and the strategies that could be incorporated to enhance users' engagement and acceptance of projects. Our aim is to develop a software that is fully interactive in all its visualizations and behaviors in order to yield the highest engagement rates and gather as much feedback as possible to assist a smooth construction process of wind farms.

3 Modeling of shadow cast

One of the main concerns of citizens where wind farms are planned is the flickering shadow that results from the continuous rotation of the blades on the ground or structures. This means that for certain several areas, there will be constant alternating changes in light intensity. If a person is exposed to such frequent change, this can lead to health side effects such as headaches, nausea, disorientation and dizziness [22]. Therefore regulations have been set in place which limits the shadow cast of wind turbines to certain periods and amounts throughout a calendar year.

According to the "Instructions for determining and assessing the optical immissions of the wind turbines" released by the Federal Working Group on Emission Control: The shadow cast by wind turbines must not affect a residential building for more than 30 hours per year and 30 minutes a day. If this period is exceeded, the wind turbines must be switched off as long as their shadow falls on the immission point [14].

Therefore it would be in the best interest of everyone to calculate during the wind farm simulation and be aware of how much shadow would be accumulating over a year.

In Figure 4, a visualization of the flickering is visualized in a worst-case scenario which occurs in a butterfly-like pattern around each turbine.

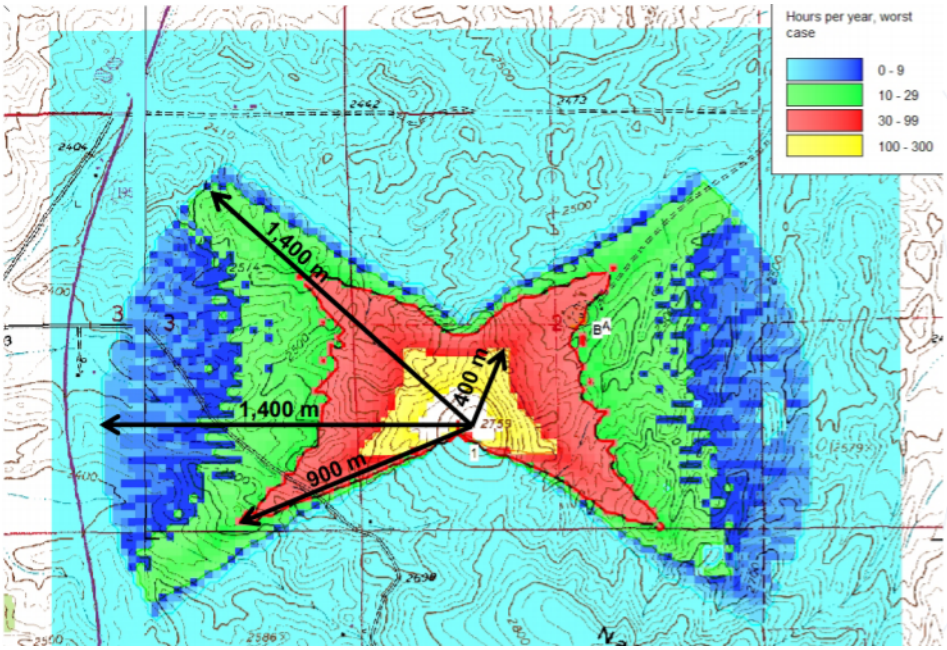

Figure 4: Flickering occurs in a butterfly-like pattern around each turbine with colors indicating the amount of shadow hours per year in a worst case scenario [22].

3.1 Shadow model

The goal of this model is to calculate the different shadow casts for the planned wind farm to have them interactively visualized in the app later on. This fulfils the goal of informing users on the potential areas where shadows would be cast. In contrast to traditional methods, where the shadow casts were inaccurately statically plotted, this approach provides a more precise demonstration that is driven by the power of data-driven simulations run by algorithms.

3.1.1 Assumptions

In the model, we assume an entirely clear sky throughout the year, without clouds, planes or other obstacles that would be blocking the sun. Additionally, we presume that the turbine would automatically reposition itself to remain facing the sun, meaning that each light ray would always hit the rotor orthogonally (in a top-down perspective). Furthermore, in the initial computation of cells with shadow, the topography is ignored and considered fully flat, which means all cells are assumed to be at height 0. Instead of simulating each continuous timestamp in a year, we simulate once every 10 minutes.

The rotor diameter and size of the used time frames are adjustable parameters to optimize for different goals. We also do not consider the shadow of the turbine pole since it is non-flickering and therefore irrelevant and won't interfere with the accuracy of the calculation.

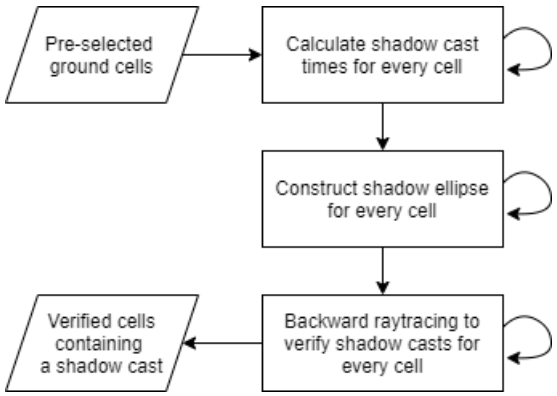

Figure 5: Flowchart of the algorithm that calculates and verifies shadow casts using backward raytracing.

Given these assumptions, we compute shadow cast values using the following dedicated algorithm that is also demonstrated by the flowchart in Figure 5.

3.1.2 Algorithm

To calculate shadow cast times for every cell, we iterate over every day of the year with some given time interval (for example, 10 minutes) and then calculate the projection of all the turbines given the sun position at this time of day and year.

We then use simple geometry to calculate the projection of the center point of the rotor. Afterwards, we calculate the length of both diameters of the shadow ellipse. One of them is the same as the rotor diameter since the turbine is positioned orthogonal to the sun at all times, and the other depends only on the sun elevation. Knowing the diameters, the center point and the azimuth of the sun, we can easily construct the shadow ellipse and all the cells inside it.

Raytracing is a technique for image synthesis: creating a 2D picture of a 3D world [11]. It is used for modeling light transport and is becoming more popular because of increases in computing power and because of its ability to cleanly deal with effects such as shadows and transparency [25].

The resulting cells are candidates, therefore we use raytracing on each of them to verify if the shadow is indeed cast there. More specifically, we compute the intersection of the projected sun vector starting from the center of the cell and the plane spanned by the rotor. If the intersection point is close enough to the rotor center, meaning the distance is less than half of the actual rotor diameter, we verify that the cell contains a shadow. After determining which candidates are actually shadowed and only if none of these cells contains a residential building, we save the 10 minutes for each of these cells.

A vector equation like

$$x = s + t \cdot d$$

can be used to calculate the intersection x between the projected light ray from the sun vector s with a direction d and t is the distance between x and s. Distance t is determined using the quadratic formula:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}. \tag{1}$$

where

$$b^2 - 4ac \tag{2}$$

is the discriminant of the equation and should yield a positive value should there be an intersection between the sun ray and the cell. Should the intersection point be close enough to the rotor center, it will be considered in the results of cells that contain a shadow cast.

The code below describes the iterative process of compiling the verified cells that contain shadows:

Algorithm 1 Detect which cells have a shadow cast

```
for each cell in candidates do
    intersection ← computeIntersection(sunVector, cell, plane)
    if intersection < rotorDiameter/2 then
        cellsContainingShadow.insert(cell)
    else
        skip
    end if
end for
```

3.1.3 Acceleration

Since checking for every cell without a prior selection is computationally infeasible and poses several limitations, the approach we take is to initially compile a list of candidates and then verify each of them. With this, a hybrid approach is enabled, as simulating without topography is very fast, and the computationally costly raytracing has to be done only for a selection of promising cells.

We settle on two parameters, allowing a trade-off between precision and speed. The two parameters are the used time frame (which we call time delta (e.g. 10 minutes as used in the description of the algorithm), and the used rotor diameter size, given as a multiplier of the actual size (which we call diameter multiplier, e.g. two as used in the description).

To illustrate, if we simulate each minute separately and use a rotor diameter of 20 times the actual length, we are very likely to find all relevant candidates and compute the actual shadow times with high precision (less than 1min fault per cell), but the runtime would be extremely long. On the other extreme, we can simulate windows of 3 hours and use the actual rotor diameter to generate the candidates. This leads to a rapid completion, but some shadowed cells might not show up in the candidate list, and the results are strongly inaccurate, as either 3 hours or no shadow is added in each iteration.

To improve the performance and hence allow for a more precise simulation, parallel processing is used. Each day of the year is distributed on the available processes (currently set to 4 due to resource limitations), and then each process simulates the assigned days independently and stores the result. Using two as the diameter multiplier and 10-minute windows, a runtime of roughly 6 hours on 4 cores of an i5 CPU is achieved.

3.2 Polygon calculation

Given shadow cast or noise propagation values for each cell, we compute borderlines of polygons for given threshold specifications. For example, an area where shadow cast duration is over 10 minutes a year, a surrounding area of 5 minutes or more, etc are computed by specifying all cells that lay on the borderline of each area.

To achieve this, each cell gets classified into a discrete class depending on the thresholds specified and the given shadow value. Next, borderline cells get identified by comparing each cell class to the classes of directly neighbouring cells. If these cells differ, the cell must be on a borderline. To avoid double borderlines, only two of the four neighbours of each cell are considered. A double borderline means that two different neighbours would lead to both being considered as borderline cells because of one another.

After identifying all borderline cells, each is stored with their closest threshold value to specify to which polygon each of them belong. As classifying each cell takes $O(n)$, comparing each cell to two of its neighbours takes $O(2n) = O(n)$ and storing the cells takes another $O(n)$ time, the total theoretical runtime of the algorithm is $O(n)$, where n is the number of cells in the grid.

In figures 6 and 7, the results of simulating the first 120 days of 2021 can be seen. In Figure 6, the intensity of the color reflects the accumulated minutes of shadow over the simulated timespan, where in Figure 7, each entry is a segment of the borderline of its respective polygon.

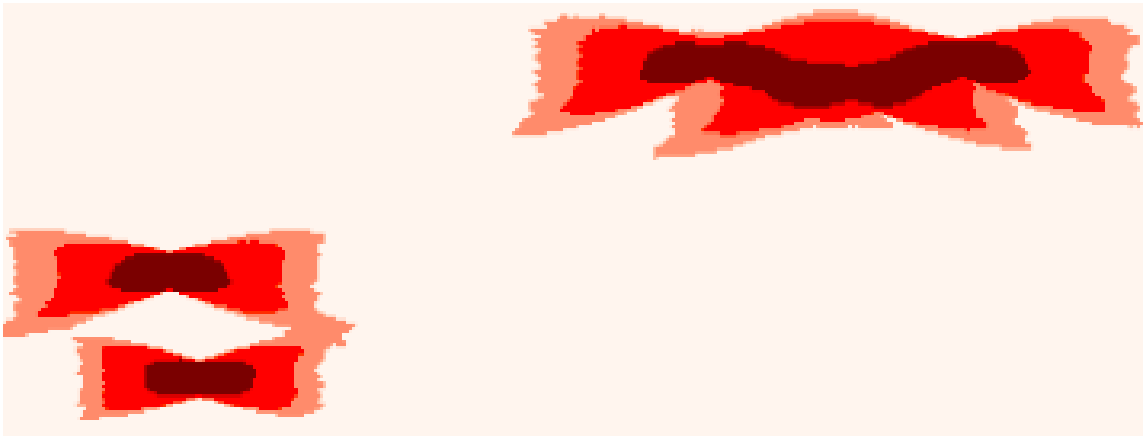

Figure 6: The calculated polygons, with colors indicating the number of accumulated shadow in hours. Darker color intensities indicate longer periods of exposure to shadow.

Figure 7: Borderline results of the simulation for the relative polygons demonstrated in 6, plotted on a world map.

4 User experience design for interactive visualizations

The last decades have seen great technological advancements and progress in mobile applications that are in some occasions used by a different variety of large target user groups. This has raised the necessity for delivering apps that are easy to use, intuitive, and self-explanatory. An app that manages to fulfill its business goal smoothly through designing a satisfying experience for its users, is said to be a successful app. In this section, we look into how we approached designing an attractive user experience from start to finish in order to guide users through the presented information, interactive visualizations and engagement opportunities. This was achieved by utilizing the process of prototyping, incorporating feedback from revisions and iterations and following user experience laws and standards.

Perfecting user experience involves taking into account different factors and always take them into consideration during the designing process. These include the different device types, operating system, screen size that all have to be handled, they add up to what is called the Mobile Equation [18].

Interactive visualization is increasingly embraced as a medium for recording, analyzing, and communicating data [28]. A great focus of this work is on the interactiveness of all what is visualized, because effective communication of information can only happen with effective design, and we believe interactive visualizations of data and knowledge contribute massively to this result.

4.1 Prototyping

Prototyping is a development approach used to improve planning and execution of software projects by developing executable software systems (prototypes) for experimental purposes. It is very suitable for gaining experience in new application areas and for supporting incremental or evolutionary software development [1]. In other words, it is a simulation of the final interaction between the user and the interface they will be using, and a simulation or envisioning of how a finished product will work. It allows product developers to test the usability and feasability of their proposed designs. Essentially, they are crucial to ensure the design concepts work as intended and determine if people are able to use the product, and detect usability issues before launch.

In today's software industry, achieving the right user experience (UX) design means acheiving effective communication of information to the users that belong to our target group. Prototyping plays a vital role in the process of creating a successful UX. This is due to the minimal effort that is required to create them, as they can be as simple as sketches on paper, and minimalistic aspects when convering that to a digital environment on a prototyping software. This gives the ability to focus on the design and experience and the details involved in them rather than the app content itself. With several iterations, coupled with evaluations and incorporated feedback from the potential end-users, a presentable and accurate product could eventually be developed

and launched.

Prototypes however don't have to look like the final product, they can have different levels of detail, different fidelty. The fidelty of a prototype refers to how it conveys the look and feel of the final product. Fidelty usually varies in areas like the **Visual Design**, **Content**, and **Interactivity**. The two extremes are low and high fidelty.

1. **Low Fidelty Prototypes**: They are often paper-based and do not allow user interaction. They are helpful in enabling early stage visualizations of different suggestions and solutions of proposed designs, and they help in the brainstorming process, as well as provoke innovation and improvement. When asking for feedback, users are more likely to feel more comfortable suggesting changes than when presented with a more sophisticated prototype.
2. **High Fidelty Prototypes**: They are often computer-based and usually allow for more realisitic user interactions such as clicking or using hand gestures in the case of apps. They are the closest representation of the final user interface. They are assumed to be more effective in collecting true human performance metrics and in demonstrating the actual products before initiating the actual implementation and technical development process.

For our prototype, a low fidelty was the starting point through several paper prototype sketches. Given it's cost and time effectiveness, it was a rapid way to gain initial insights and clarifications on how we want to implement things and helped define exactly what our goals were for the app. The second step was implementing a high fidelty prototype that is digital and involves more details in terms of visual design, content, and interactivity. For that, we used a prototyping tool called Figma, which is web-based and focuses specifically on user interface and UX design. The results of using Figma to prototype our initial thoughts and ideas for the app have been very enlightning and enabled several productive iterations that gradually developed several mockup screens of how the app would look like. Screenshots of the results can be seen in Figures 8 and 9.

A remarkable observation here would be that the end product that has been developed which will be presented in the upcoming sections matches the prototype results with a very high degree, which proves the importance of the activity of prototyping and how it can be utilized as an integral and time-efficient method before the technical implementation in order to reach design decisions.

4.2 Laws & practices

Throughout the design process and prototyping several UX laws have been followed in order to ensure the best results for the user interface that guarantee an optimal and intuitive UX, which is one of the main priorities of representing the visualization results. They have been implemented from the start on, during the prototyping phase which has been discussed in the previous section, and also in the actual frontend designing and

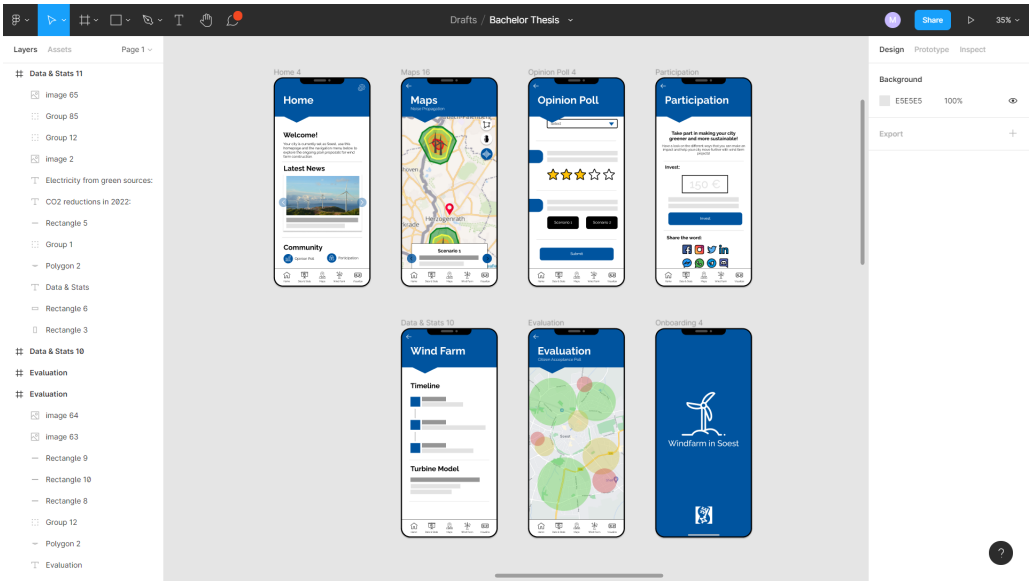

Figure 8: An overview of the app prototype from inside Figma.

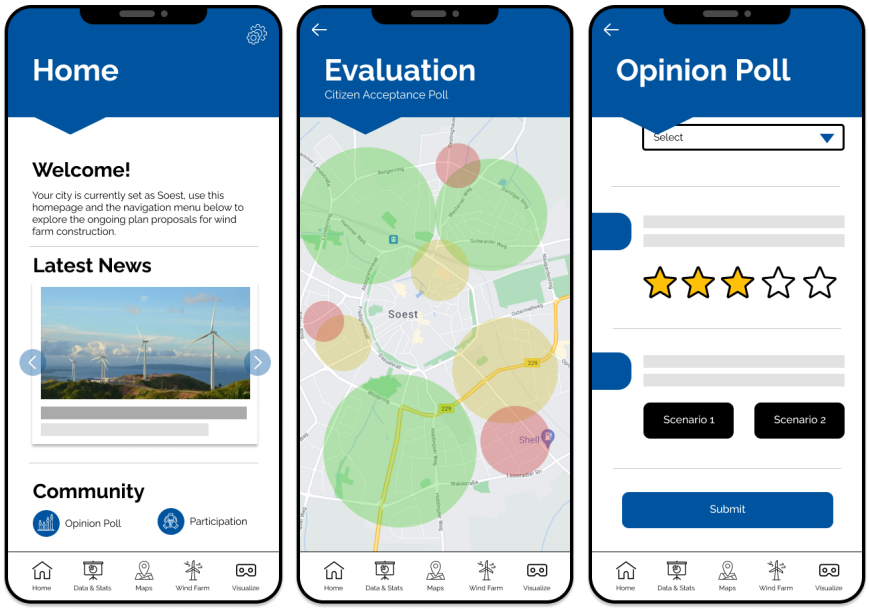

Figure 9: Multiple screenshots of the final app mockups.

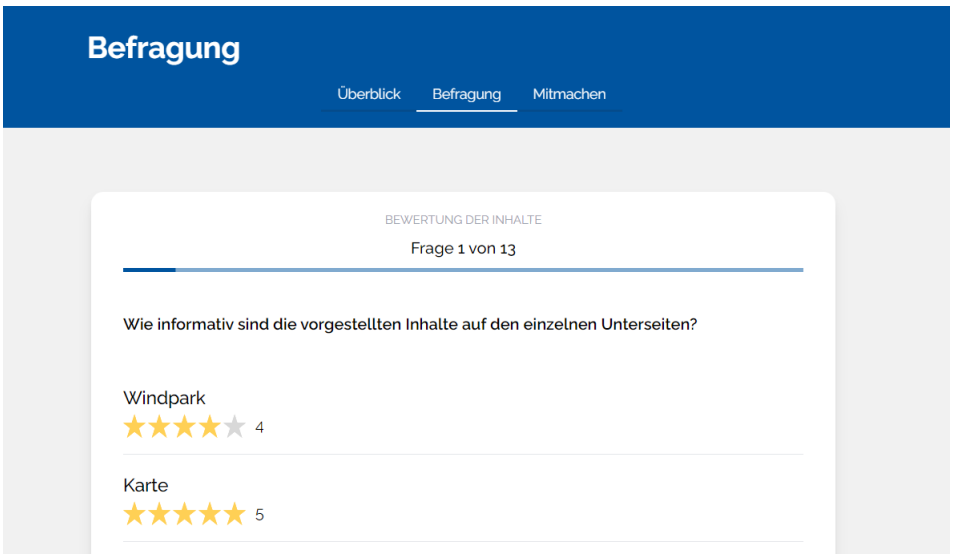

Figure 10: Screenshot with the opinion poll demonstrating the progress monitoring that provokes the Zeigarnik Effect.

coding. These laws achieve several UX goals such as usability, learnability, efficiency of use, memorability, efficiency of use and many others. For a product to be usable, these dimensions have to be considered [19].

Below are some of the most important laws that can be easily detected from a first impression on the app:

1. **Zeigarnik Effect**[1]: People tend to remember uncompleted or interrupted tasks better than completed tasks [27]. This has been mainly reflected in pages where the user's input was prompted, such as opinion polls, participation, and feedbacks. That is essential and vital to fulfill the app's purpose of engaging users with the planning of the wind park. Clear indications of progress in the opinion poll through a moving progress bar with every answered questions reinforces this effect by giving users an objective and motivation to finish the questions till the end, see Figure 10. In the participation page, certain questions and content are displayed ("discovered") the more they answer, which urges users to continue engaging, as according to the law this is how they would best remember the content they were interacting with. Bogoslavsky and Guthrie (1941) suggested that tension present during the solving of a problem increases the problem's memorability [3], and this can be reflected in the somehow challneging nature of the opinion poll where users have to make decisions on their preferred wind farm scenarios for example.
2. **Von Restorff Effect**[2]: Also known as The Isolation Effect, predicts that when

[1]Zeigarnik Effect - `https://lawsofux.com/zeigarnik-effect/`
[2]Von Restorff Effect - `https://lawsofux.com/von-restorff-effect/`

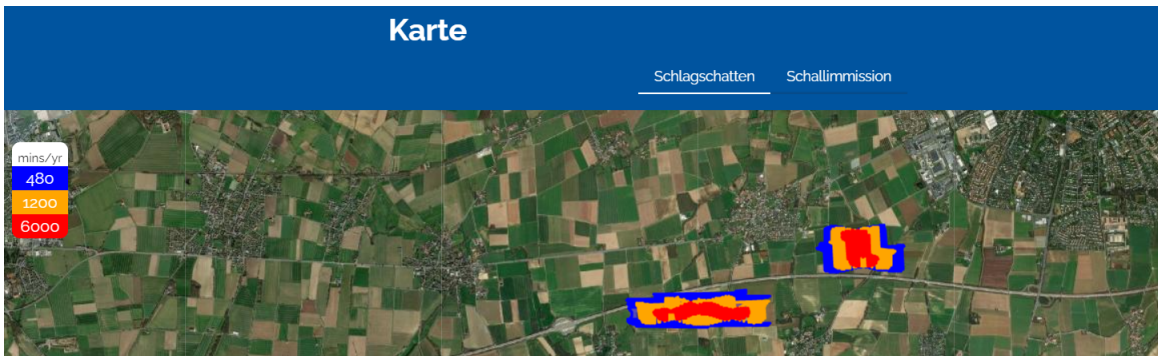

Figure 11: Screenshot with the shadow cast visualization on the map.

Figure 12: Screenshot with an example of how the app was structured to minimize the amount of content displayed simultaneously.

multiple similar objects are present, the one that differs from the rest is most likely to be remembered. This can be clearly detected for example in Map Displays where the shadow casts (see Figure 11) are distinctively marked on the map area with visual elements such as colored borders and transparent fillings to clearly diffrentiate a certain area from the remaining map, hence leaving a rememberable impression of how the map looks like with this certain piece of information that also relates to them since the user will be probably a resident in the zoomed area.

3. **Miller's Law**[3]: The average person can only keep 7 (plus or minus 2) items in their working memory. This law has been carefully followed throughout the entire development process and when structuring the content in different pages. Where each page has only a maximum of 6 elements, on average 3, and when needed pages were split into subtabs that are easily accessible from the header of the app interface, see Figure 12. Therefore giving the user the luxury of having enough elements in their working memory so that they have their full attention on what is important and what really matters.

4. **Law of Common Region**[4]: Elements tend to be perceived into groups if they are sharing an area with a clear defined boundary. This law can be noticed throughout the entire app, where all contents that are related to each other are grouped together in cards. The card component is the most frequently used component in the whole app, as consistency of displaying content that is related to each other in a certain area was crucial. This card has certain features such

[3]Miller's Law - `https://lawsofux.com/millers-law/`
[4]Law of Common Region - `https://lawsofux.com/law-of-common-region/`

Figure 13: Screenshot demonstrating the law of common region through the card component.

as following consistent inner and outer spacing as well as consistent styling like a background color that distinguishes it from the background color of the app, see Figure 13. Figuring heavily in grouping by common region is the fact that elements located within a contour or on a region tend to be captured by it in the sense of being seen as belonging to it-and possibly even attached to it-rather than as floating in front with no visible means of support [20].

5. **Jakob's Law**[5]: Users spend most of their time on other sites. This means that users prefer sites to work the same way as all other sites they already know. This law was particularly important to address since the target group of the app is a wide range of people from different ages and backgrounds. Therefore the implementations have been carefully designed to be intuitive and self-explanatory by following the modern and common standards. This means all icons used for example are ones that aren't new or necessarily different, they are icons that can be seen in almost all websites (For example: the save icon is the widely known floppy disk which can be seen in text editing programs and many other websites, and it automatically indicates saving the state of the current input). This can also be noticed in the way the navigation structure on the app works, where the bottom navigation is the main one since it's always there and accessible the way all apps these days have, see Figure 14. By leveraging mental models, we created a superior UX in which users can focus on the tasks in the app rather than learning new models to interact with the tasks.

6. **Fitt's Law**[6]: The time to acquire a target is a function of the distance to and

[5]Jakob's Law - `https://lawsofux.com/jakobs-law/`
[6]Fitt's Law - `https://lawsofux.com/fittss-law/`

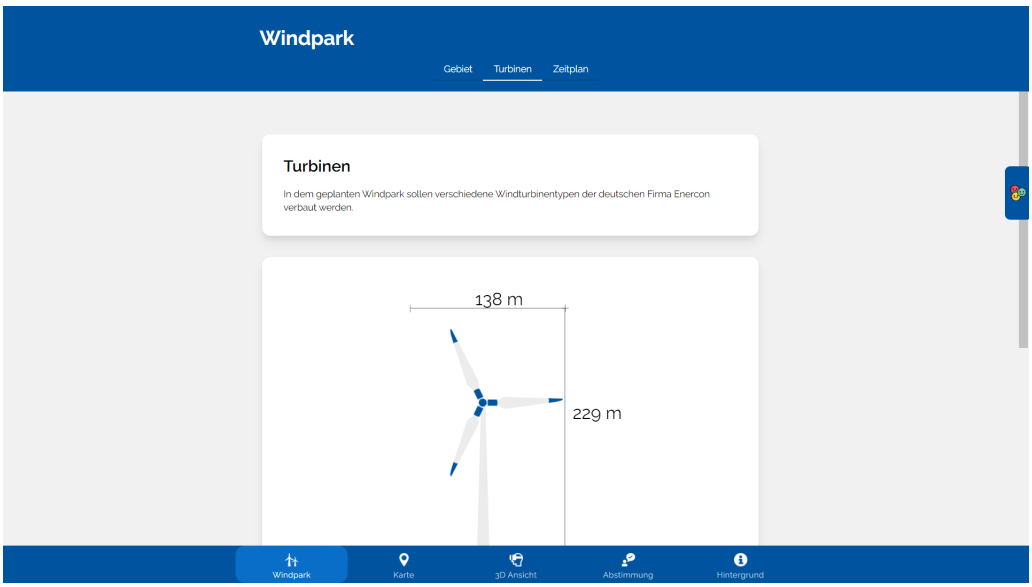

Figure 14: Screenshot demonstrating the familiar look of the app in comparison to other apps (Bottom navigation menu).

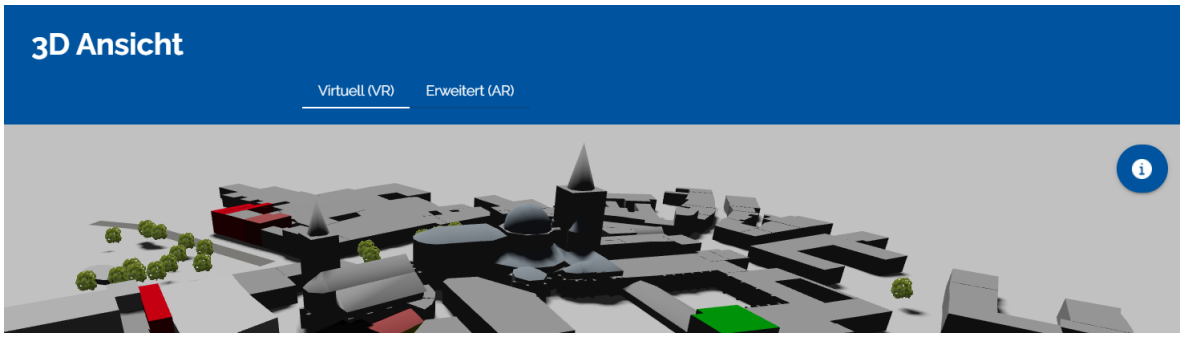

Figure 15: Screenshot demonstrating the strategic placement of buttons in order to comply with Fitt's Law.

size of the target [9]. In other words, our goal is to minimize the time to acquire targets such as clicking buttons, and to achieve that, targets should be large enough for users to accurately select them, and they should have spacing between them, should multiple ones be in proximity to one another. This can also be clearly noticed everywhere where buttons or points of interactions are there, where the size is adequate enough to be easily acquired and also have enough relaxed spacing from its surrounding to avoid confusion and to minimze the time of selection/touching/acquiring, see Figure 15.

5 Software construction

The main aim of this work on the technical level is to develop an interactive application that is sustainable and would support future integrations to it. Maintaining code quality and readability was one of the highest priorities during the entire project lifecycle. Documenting and ensuring a smooth handover to future developers was carefully thought of through every step of the process. Each technology that has been chosen was heavily researched and will be discussed throughout this paper, supported by arguments and justifications on why a named choice was preferred over another.

5.1 Goals

Users need to be considered early and often. Usability needed to be a part of every step of the design process [4], and to achieve this, pre-defined goals were set so that by the end of our milestone, we would have achieved a user-centered product that is of high quality. The main design objective of an effective web design should be to fulfill the different goals of the user [12] and for that they are also considered in every aspect of the goal setting process. Below, we discuss some of the main goals:

1. **Accessibility**: The target group for the result of this work is a wide array of users, that encompasses different backgrounds, ages, as well as different mediums of access. This means that we had to ensure that all results were accessible from all types of devices, this means a huge number of differing screen sizes that had to display the same transparent results regardless of the interface a user is using. With the increasing number of operating systems these days, as well as different screen sizes that range from mobile phones to large desktop screens, in addition to the different manufacturers and the variability in their display settings, a challenge was presented. The result was using state of the art modern technologies that handle these variables behind the scenes through single codebases and responsive frameworks.

2. **Maintainability**: Projects are usually handled and contributed to by different developers, which means that code maintainability was a high priority during the whole process. This meant using the best practices and latest standards to ensure smooth further development in the future and embrace the full scalability potential that this work presents. The different components and building blocks were built and designed in a way that guarantees easy understanding as well as promotes further additions and integrations without running into problems when the codebase grows. This was supported by impressive error handling implementations in every part of the work that prevents any unexpected crashes and predicts any potential issues. Another aspect to ensure maintainability is using technologies that have well-documented recourses and engaging, large and active communities.

3. **Compatibility**: As this work is merely a frontend visualization for the results of

complex algorithms and computations that existed initially from previous contributions, all components and frameworks used had to be compatible with the existing work that has been bundled into one app. This meant ensuring smooth and flawless communication and transactions between the different API endpoints, as well as a robust and organized structure of storing data to make sure it's compatible and reusable when passed on to a different component within the app.

4. **Simplicity**: As previously mentioned in Accessibility, the target group for this user encompasses a huge number of them that differ drastically. This presented the task of developing an easy-to-use app, that is intuitive, visually appealing and allows easy navigation back and forth between all pages, components and features without feeling the weight of doing so. The choices made to achieve such simplicity will be discussed more in further detail in the User Interface and User Experience design section of this paper. An integral aspect perhaps was to structure all presented content in a way that complies with the two-tap rule, where users can access any page from anywhere in a maximum of two taps.

5. **Reusability**: Large projects could get messy, but thanks to the choice of technologies that will be discussed soon, the concept of reusability have been heavily utilized which resulted in a highly efficient and dynamic workflow, rapid and high-quality results, and a clean and readable codebase. Reusing previously coded components whether custom developed or integrated from the open-source community assisted in minimizing code length, supporting the modularity of the work, and presenting the opportunity to focus on what matters instead of duplicating and repeating certain tasks.

5.2 Architecture

In this section, we discuss the architecture of the entire app and how the different flows of data behave and which services interact with each other, see Figure 16. We will discuss the whole process from scratch up until the eventually visualized results.

There are three main abstract levels: Frontend, Backend, Offline Calculations.

1. **Frontend**: This level is mainly responsible for representing everything to the end-user understandably and intuitively, all UI and UX efforts that will be discussed later on have been channelled into this level. This level interacts directly with the Backend level to read and store data. It is also worth noting that this level makes use of the hosting device's local storage where it stores data that could be relevant for the user and also submit them later on to the backend when the device is connected to the internet.

2. **Backend**: This level stands between the frontend and the offline computations/-calculations. It is the middleman that takes care of retrieval and storing of data,

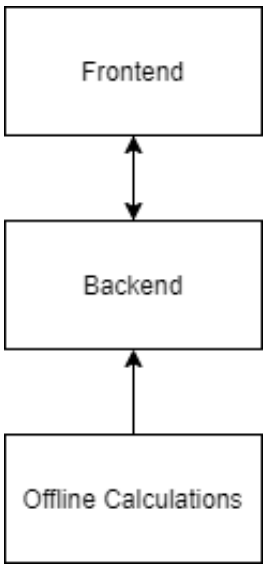

Figure 16: Diagram demonstrating the communication and data flow between the different abstract levels.

as well as managing access permissions, and providing the relevant endpoints needed to access the database it manages upon request.

3. **Offline calculations**: This level includes all the relevant algorithms (written in Python & C++) that perform all the calculations for the optimal positioning of wind turbines, turbulence, noise, shadow cast. The outcomes of these calculations are stored in JSON files that are later stored in the database managed by the backend, which is then accessed by the frontend to display the calculations meaningfully to end-users.

Below we will further explain each level in detail, how it's operating behind the scenes, which main tasks does it have, as well as the data, flows and the endpoints it communicates with.

Starting with the frontend, this level handles all the visual representations of the work. This is done through pre-built and custom-built reusable Vue components that are placed and grouped alongside the relevant data to display certain views for end-users. The frontend level also provides users with all the navigation capabilities to access the different pages of the app, this is handled using Vue's highly renowned routing library.

Aside from the styling and design of the app as a whole, the app mainly revolves around the data that are displayed within these design elements. And these data come from mainly two sources: a local JSON file and the Backend endpoints. The local JSON file includes all the hardcoded content such as titles, headers, texts, button texts, etc... The reasoning behind putting all texts in a single JSON file and not hardcoding them directly in source files is three things:

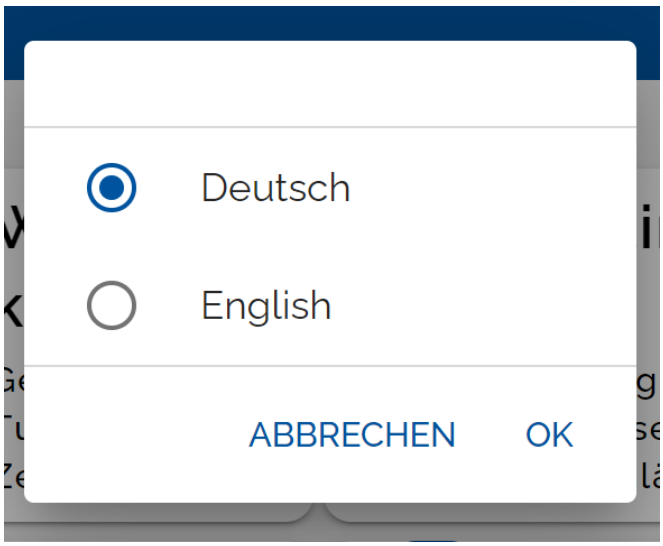

Figure 17: Popup for language selection.

1. Adds the possibility of localization. Meaning that to provide the app in other languages that can be switchable from within the app, see Figure 17. The only requirement would be to translate a single file, and this would dynamically take effect on all source files depending on the user's preferred language of choice.
2. Adds simplicity for non-technical users. As this app is mainly developed to inform its users, the essence of this app relies on what is displayed rather than the technological impressiveness behind the scenes. To perfect what is displayed, it is not necessary that the person writing the texts has a technical background, and a JSON file provides a more readable format that can be easily edited, without having to go through the many source code files and changing texts there and risking altering code that could result in bugs.
3. Adds dynamism and reusability. The app could be thought of as a large placeholder waiting to be filled with the relevant information. This doesn't only include texts, but also certain styling elements such as colours. Therefore, using a JSON file as a centralized entity to dynamically present the desired information adds the possibility to reuse the app for multiple cities, by just changing the texts, photos, and colours to match the theme of the city where the app will be mainly targeting its citizens.

Below is an excerpt from the main JSON file that was just briefly explained:

```
{
 "en": { ... },
 "de": {
         ...
   "headers": {
        "home": "Startseite",
```

```
        "windpark": "Windpark",
        "map": "Karte",
        "visualization": "3D Ansicht",
        "opinion": "Abstimmung",
        "background": "Hintergrund",
        "engagement": "Ihre Meinung",
        "opinionPoll": "Meinung zum Windpark-Projekt",
        "participate": "Buergerbeteiligung am Windpark-
          Projekt",
        "evaluation": "Auswertung"
    },
        ...
  }
}
```

The second source is from the backend REST API through the different endpoints it offers, these data are the essence of the visualizations provided on the app, they are the final results of the algorithms of the level of the offline calculations. The data is essentially retrieved once during booting up of the app and is stored locally using Vuex Store so that it's accessible even if there is no connection to the internet which is needed when accessing the Backend API's endpoints. Also for efficiency reasons and to save the data usage of users, a simple versioning system has been implemented that detects if there has been any change to the data on the backend and only retrieves them if there has been a change, otherwise, the existing JSON files will continue being used.

The JSON files retrieved include data that will be used in visualizations such as shadow cast, noise levels, optimal positioning of wind turbines, where the results of the calculations that will be map coordinates and numbers, will be visualized on maps with colours and indicators that visually make more sense to end-users than just plain numbers.

The second level in our architecture is the backend. It is mainly responsible for communicating with the front end and providing it with the data it requests. However, it also hosts the functionalities and capabilities needed for generating and exporting 3D scenes for the visual simulations such as the virtual reality and augmented reality components of the app.

The third level is the offline calculations, which performs heavy calculations on data provided by wind farm planners that could take a long while and wouldn't be very user friendly if performed on the spot. Therefore they are all performed offline, hence offline calculations, that are then submitted to the backend in the MongoDB database it is managing. These calculations include the estimations of noise propagation, optimized placement of wind turbines, potential area, etc...

5.3 Technologies used

Below, we give an overview of the technology stack that was utilized in the construction process of our software supported with arguments and justifications of why certain choices were made and the advantages they bring along.

5.3.1 Hybrid application (Ionic + Capacitor)

Hybrid applications are simply web applications that can be installed on a device, and with that winning the best of both worlds, the power of the web, and the native features of devices that run on operating systems like Android and iOS. They are essentially deployed in a native container that uses a mobile WebView object, and when used, this object displays web elements and content exactly as a website would normally do using web technologies such as HTML, CSS, and Javascript, see Figure 18. A remarkable aspect of hybrid applications is that they can gain access to the device's hardware features such as (accelerometer, camera, GPS, gyroscope, etc...), which will come in handy for the visualizations of our work. An example would be augmented reality where we make use of the hybrid app's features to use the device's camera and place 3D models of wind turbines in their surroundings, giving them a glimpse of how a wind farm project would look like when implemented in real life. Another aspect is the ability to easily integrate existing web components to the app like 3D visualizations that have been previously developed in three.js (also a web technology). Therefore using a technology that supports the web was a high priority when choosing the development stack.

For our work, we use Ionic[7] to implement a hybrid application. Ionic is an open-source software development kit for hybrid mobile app development, and can be used alongside frontend frameworks that facilitate user interface design such as Angular, React and Vue (more on that in the next section). It provides tools and services for developing apps that can run on mobile (iOS and Android), desktop, as well as progressive web apps that use the latest and most modern web development practices. The results of using Ionic can be distributed on native app stores and enable installing on devices through utilizing cross-platform frameworks like Cordova[8] or Capacitor[9].

In our case, we use Capacitor through plugins to gain full native access to device hardware features like GPS and Camera for example. Ionic simplifies app building and deployment to different platforms by wrapping around the Capacitor build tool.

The advantages of using hybrid applications include gaining the ability to function whether or not the device is connected to the internet while still making full use of web-based services, unlike normal web apps which need internet. They can operate on different platforms by developing and maintaining only one codebase, unlike

[7] Cross-Platform Mobile App Development: Ionic Framework - `https://ionicframework.com/`
[8] Apache Cordova - `https://cordova.apache.org/`
[9] Capacitor - build cross platform apps with the web - `https://capacitorjs.com/`

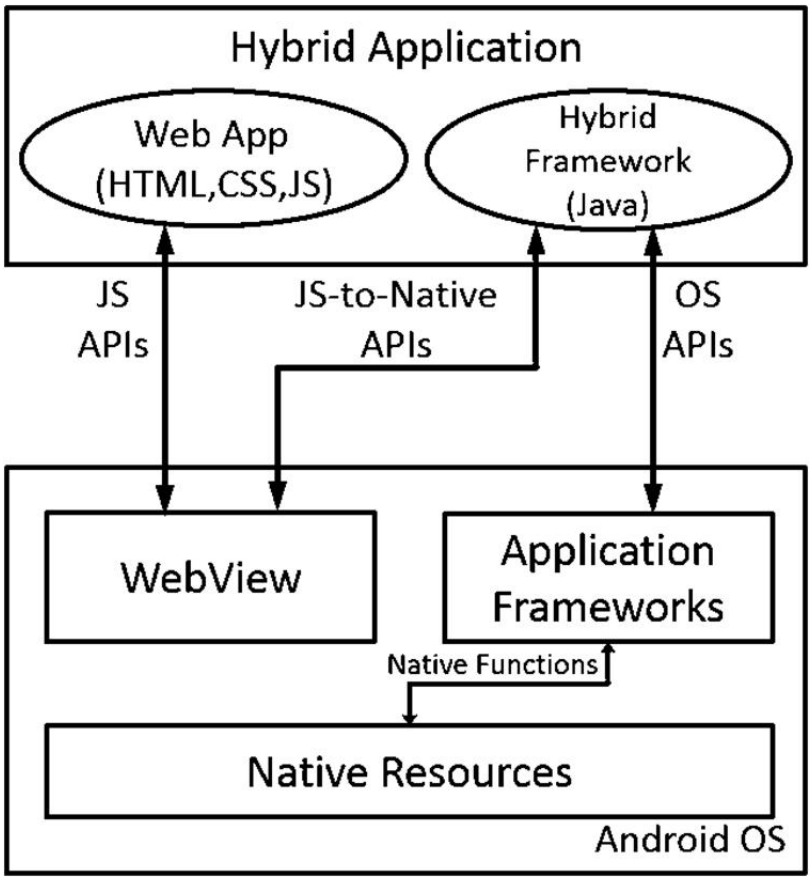

Figure 18: Hybrid application architecture [17].

traditional mobile app development where separate codebases are needed for every operating system. Ionic is therefore platform independant [13]. This complies very well with the limited recourses we had for this work in terms of human recourses and time constraints. This leads to faster development and building times compared to native app development. This also leads to drastic cost reductions compared to building two apps for two different platforms. It is also easier to launch patches and updates which aligns very well with our requirements where many updates to the wind farn scenarios will need to be released as part of the engagement strategy with the users of the app. Additionally, hybrid apps enable the use of the device's local storage, which will be discussed in more detail in a future section and how this helps in gaining database functionalities of a web app while being offline.

The disadvantages include possible variations in the app on two different platforms or devices due to leaning to a certain platform during development and test, this could result in potential unhandled bugs that could arise on a different platform that wasn't tested, as well as appearance variations. A measure to counter this disadvantage is to increase the testing of the application on a wide range of devices to ensure proper and smooth operation. Additionally, since hybrid apps add an extra layer between the source code and the target platform, they may perform slightly slower than native or web versions of the same app where the source code interacts directly with the platform.

5.3.2 Frontend framework (VueJS)

Frontend development involves the parts of the application that the users see, in contrast to the backend where it is more about the behind the scenes functionalities. As with any framework in software engineering, the concept behind frontend frameworks is essentially bundled code that other developers have written and can be included in applications to help accelerate and facilitate the development process.

Using frontend frameworks eases the workflow for developers, speeds up development and improves the robustness and stability of the application being built. Building entire applications with just Javascript, also called VanillaJS, can be very tedious and recourse inefficient because there will be a lot of repetitiveness and duplication. Frontend frameworks present the power of reusability and dynamism, as well as vigorous data management in data-intensive and data-driven apps, which is the case for our work.

For our work, we use VueJS[10], which is an open-source lightweight frontend javascript framework for building user interfaces and single-page applications. Perhaps the most remarkable feature of Vue and frontend frameworks, in general, are components, which extend HTML elements to encapsulate code and make them reusable in different instances and render them as much as needed with dynamic content, see Figure 19. An

[10]Vue.js - The Progressive JavaScript Framework - `https://vuejs.org`

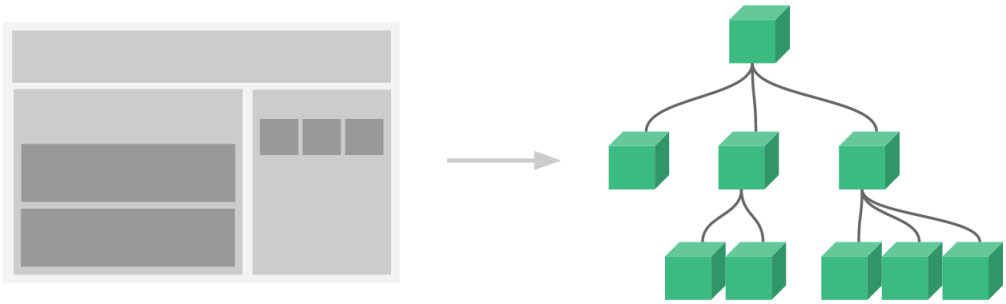

Figure 19: Apps in VueJS consist of different components within each other.

example to explain this concept is a social media's news feed where there are a quasi-infinite number of posts, all posts have the same features and design but changing contents depending on the poster, date of posting, interactors, etc... The component in this case is an empty post that has placeholders for the different data, and the component is duplicated and rendered (filled in) with the respective data needed to be displayed for a certain user. Such data are passed interchangeably between components using Vue's props and rendered dynamically in the component's HTML declarations with JSX which is a syntax extension to Javascript that enables HTML co-existence with Javascript code. This brings us to Vue's HTML-based template syntax that allows binding the rendered DOM (a programming interface for web documents) to the component's data, which makes use of Vue's virtual DOM render functions. A virtual Document Object Model allows Vue to render components in its memory before updating the browser, which enhances and boosts performance, as well as enables dynamic changing of data on web pages.

Notable alternatives to VueJS are Angular[11] and React[12]. VueJS according to the number of stars on Github has the highest growing popularity among developers which is a big plus when using frameworks, as there is always ongoing development and improvements to the framework's architecture and performance. In addition to the large community that present great support through contributions of reusable components for example. Vue is also highly customizable in comparison to its peers and allows combining UI and behaviour of components from within a script. Unlike its peers, there is no need to be forced to follow the framework's way of doing things, which provides high flexibility and agility.

[11] Angular - `https://angular.io/`
[12] React - A JavaScript library for building user interfaces - `https://reactjs.org/`

5.3.3 Styling libraries (Tailwind)

One integral part of eye-catching apps is styling. To implement modern designs that meet the users' expectations, modern styling techniques have to be followed as well. For that, CSS styling libraries have been utilized to achieve efficiency in the frontend development process. Such libraries offer pre-defined classes that are reusable and help maintain consistency throughout the code. They also help in sustaining clean code that is readable and easily adjusted in the future. For our work, we used Tailwind[13], which is a utility-first CSS framework packed with classes that can be composed to build any design directly in the HTML files with little need to resorting to CSS files.

This not only results in extremely rapid progress but also a more minimal code structure. Another advantage of Tailwind is the naming conventions that are unified and pre-defined as part of the documentation which enables easy access to styles that need certain changes. One more advantage of the concept of utility classes that are being utilized by this library is that it helps to work within the constraints of a system instead of littering stylesheets with arbitrary values that are hardcoded. This makes it easy to be consistent with spacing, typography, shadows, and everything else that makes up a well-engineered design system.

For example, the **shadow-lg** class from tailwind defined a certain shadow styling that has been used all over the app to maintain a consistent shadow for all elements with this single keyword.

Tailwind also is very high in performance where it makes use of CSS Purging to automatically remove all unused CSS when building for production, which means that the final CSS bundle is the smallest it could be.

Another aspect of using styling libraries, in general, is the utilities they bring when it comes to responsive design, which in our case was a high priority since the entire single codebase should be made accessible through all different types of screens. Responsive design helps enrich the user experience through creating apps that can adapt layout and content to viewing contexts across a spectrum of digital devices [10].

Traditionally, developers would have to wrestle with numerous complex media queries in CSS files, but with Tailwind, this can be done directly from within the HTML, which was very helpful during the development process as everything in terms of page structure was visible and helped have a better overview when making screen responsiveness decisions.

Last but not least, tailwind has a rapidly growing community that is driven by the idea of modern and flexible frontend development, which helps keep this library updated and supported. This also makes it more likely for creative UI components to emerge from open source contributors which will help accelerate development even more through usability.

[13]Tailwind CSS - Rapidly build modern websites without ever leaving your HTML - `https://tailwindcss.com/`

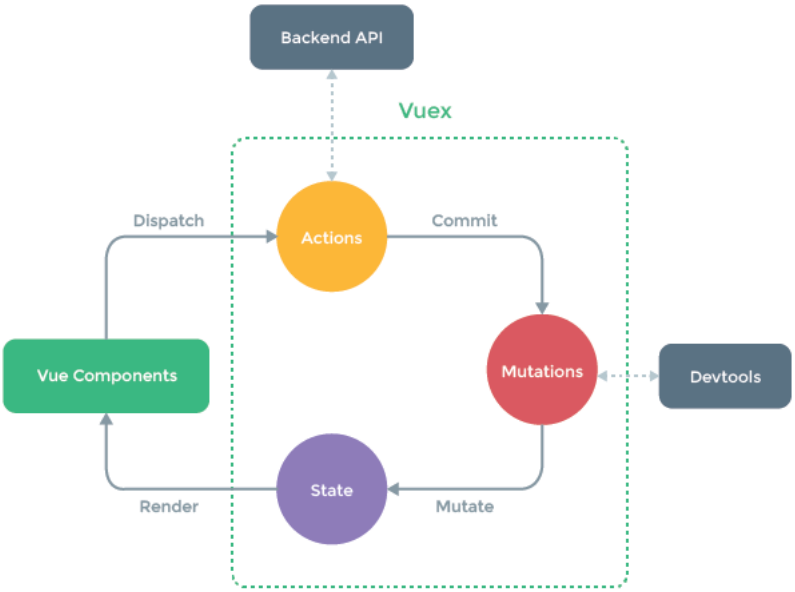

Figure 20: Vuex features integrated into an existing VueJS app with components and a backend API that provides it with data.

5.3.4 Data & state management (Vuex Store)

The app relies heavily on data and all visualizations are essentially defined by the data computed by the offline calculations and served by the backend. And to provide users with a robust and fast app performance, we needed to set up a storage to refrain from requesting the data everytime. This would also allow the app to function even in the absence of internet.

For that, we use Vuex[14], which is a state management pattern and library for Vue.js applications. It serves as a centralized store for all the components in the applications, with rules ensuring that the state can only be mutated in a predictable fashion, see Figure 20. Vuex also comes with a devtool extension that provides advanced features for debugging and developing.

Using such library allows us to save the state of the data and user's inputs during their usage of the app, which with the growing complexity of data-driven apps proves as a reliable method of retrieving data, correctly mutating them, and having them easily accessible from different components.

[14]What is Vuex? - `https://vuex.vuejs.org/`

Figure 21: Splash screen with a loading progress bar.

6 App solution

In this section we will take look at the final result of the implementations with screenshots from multiple pages of the app. We will go through the main pages and describe the functionalities implemented. Each figure displayed will consist of two versions, the mobile version and the desktop version because as mentioned before, responsiveness was put in mind from the beginning on, and we would like to show how this is reflected on the end result.

All over the app pages, visualizations and graphics take the larger share of what is being displayed, and textual information is always kept to a minimum when there is an interactive visual that can replace it, and in many occasions can be retrieved from information buttons that are placed consistently when necessary on the top right of the screen. The idea that pictures tell a more compelling story than words is a long-standing tradition [24], and it is no secret that increases users' active engagement and curiosity to learn more.

6.1 Initial points of interaction

The following screens are the users' first impression of the app, therefore simplicity and minimalistic delivery of content was ensured.

6.1.1 Splash screen

The startup of the app is through this screen, as demonstrated in 21, is where the loading of all relevant components happen, as well as any retrieval of new updates from the backend. This is done by comparing the user's local current version with the version on the backend, and downloading only if there is an update.

Figure 22: Onboarding for first time users as an introduction for the app.

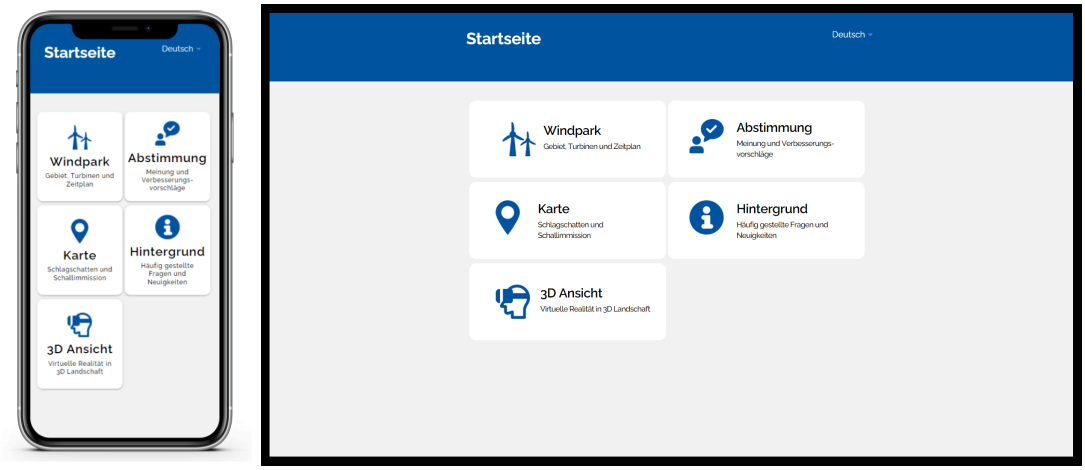

Figure 23: Dashboard home screen as the starting point to navigate the app.

6.1.2 Onboarding

This component is displayed only for first time users where they are briefly introduced to what the app has to offer, a flag is set once they proceed so that it's not displayed again. See Figure 22.

6.1.3 Home screen

This page (Figure 23 displays the main sections of the app, with different cards labeled with the respective page name and relevant icon. This page can only be accessed once in a single session, and then navigation can be performed from the traditional bottom bar. The reasoning behind going for this approach is to give users an initial impression of the possible routes on the app from a magnified prespective as these sections are the parimary ones.

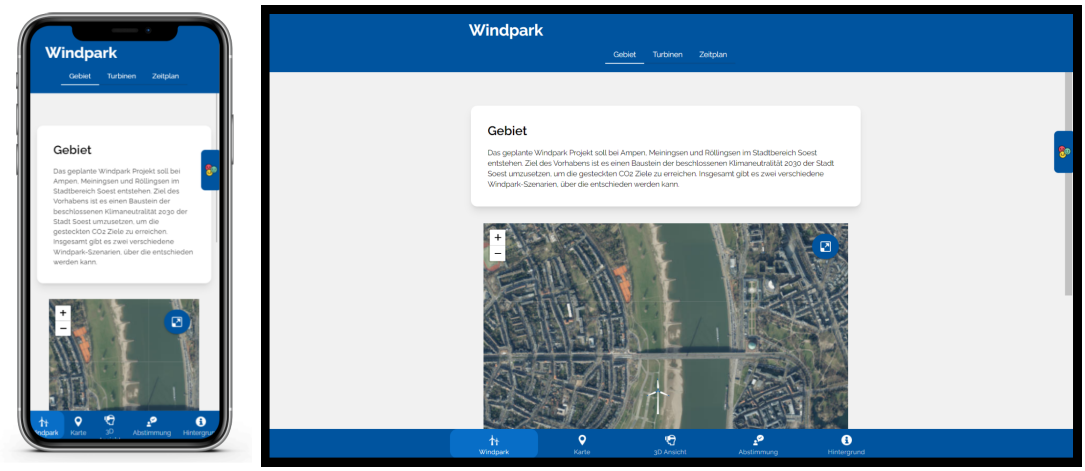

Figure 24: Wind farm area page.

6.2 Wind farm

In this category, relevant information about the overall wind park plans are displayed.

6.2.1 Area

In this page (Figure 24), the optimal positions provided by the offline calculation results retrieved from the backend are visualized on a map by placement of markers in the respective cooridnates. Users also have the possibility to switch between the different proposed scenarios for their positioning.

6.2.2 Turbine

This page (Figure 26) provides information about the used turbine models, and users can know more about the specifications and attributes of the turbines.

6.2.3 Timeline

Here (Figure 26), users are given the chance to monitor the latest updates that concerns the planning and construction. This page is kept up to date through the JSON file explained in Section 5.2.

6.3 Map Visualizations

In this category, different visualizations on maps are implemented, these include visualizations of shadow casts and noise propagations. Research and a study conducted on a group of participants has showed that interactive maps play well with good user

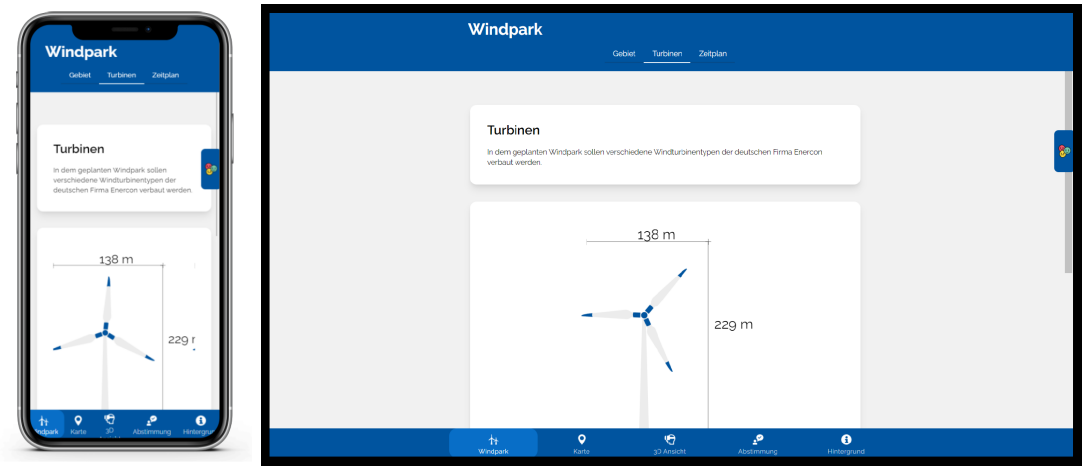

Figure 25: Turbine information page.

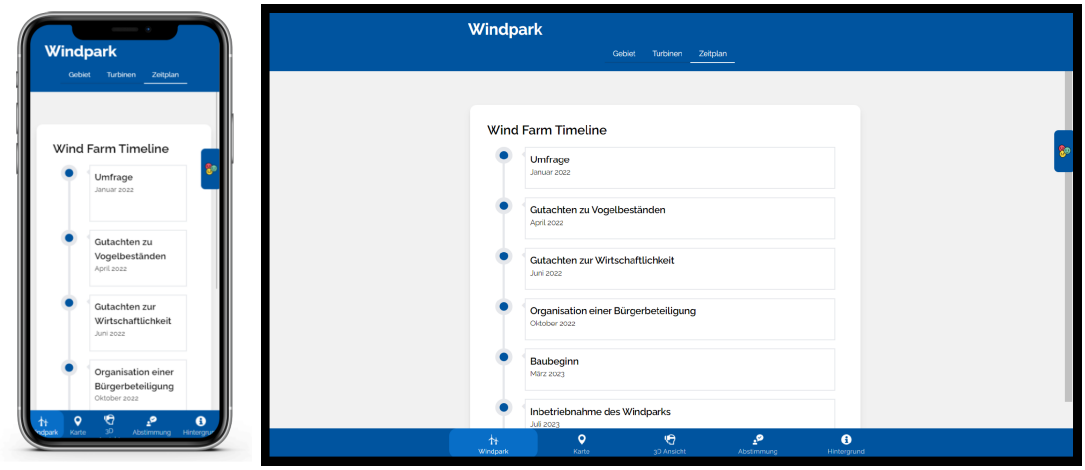

Figure 26: Wind farm timeline progress and updates page.

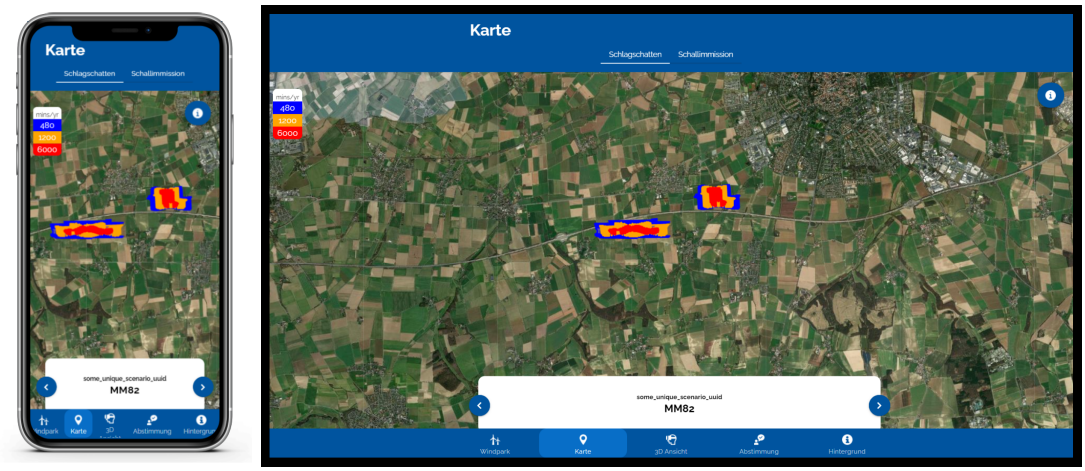

Figure 27: Shadow casts on map visualization.

experience and has an overall positive influence on it [6]. This shows the importance of visualizing elements on maps in an interactive and user-friendly way, and that is done through giving the possibility to switch between different wind farm scenarios instantaneously.

6.3.1 Shadow Cast

Through this page (Figure 27), shadow casts are visualized using polygons calculated from the simulations performed in the offline calculations level. With a legend component, the different shadow durations in minutes per year are defined, and polygons are filled respectively with the relevant color. Users here have the capability to switch between the different possible scenarios that would result in different shadow cast outputs. Informing them with the different scenarios enables them to engage by selecting their preferred scheme later on in the opinion poll. Users can also learn more about what the page has to offer by clicking/tapping on the info icon on the top right. Users with little domain knowledge will need to have the system (our app) explain what it is doing and what the different options mean [26].

6.4 3D Views

In this category, modern virtual reality and augmented reality visualizations are made possible for users. This is essential in order to give them the closest demonstration possible to a real life implementation of the planned wind park project.

6.4.1 Virtual Reality

In this page 28, users can see an animated visualization of their town and see how the wind park will look like in a virtual environment. This page is served through an

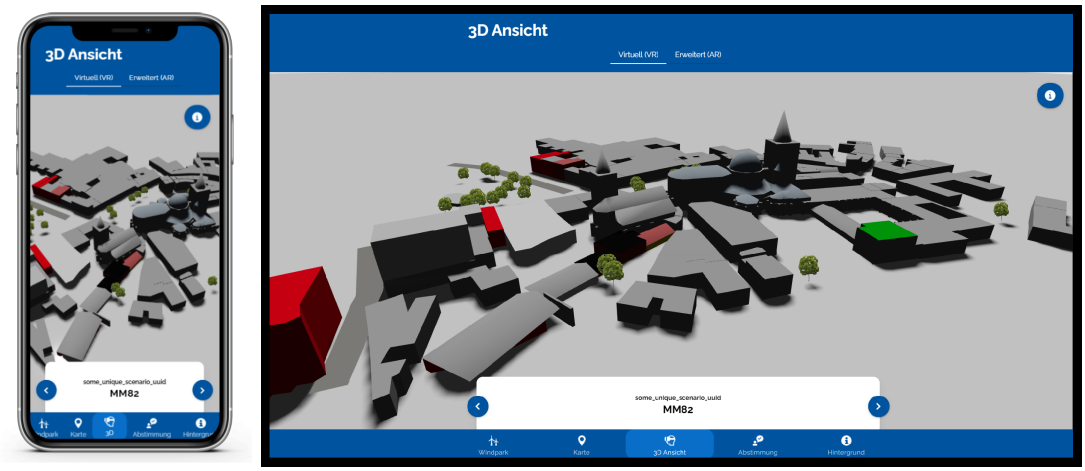

Figure 28: Virtual reality visualization.

iFrame that is hosted by the Backend.

6.5 Poll

This category provides user the opportunity to directly engage with the planning and participate with their feedback and even make financial contributions.

6.5.1 Opinion poll

In this page (Figure 29), a set of pre-defined questions are displayed for users to inquire about their feedback and insights of what they have witnessed throughout the app. The opinion poll can be accessed at all times as long as the deadline has not been reached. It also enables conditional questions, where some questions depend on the answers of previous ones, this enables more targeted questions and provides users with a desired interactivity. In the end, users are asked for sharing their location for the sole of purpose of determining if they are located in the city where the wind park is planned, they are then discarded immediately. This is done in order to only record responses of the citizens and not outsiders, with that we prevent manipulation of feedback results and only get reliable engagement information. The results are stored to be analyzed when the deadline is reached, and provide evaluations to the relevant stakeholders.

6.6 Background Information

In this category, general information such as news, statistics and frequently asked questions are compiled to provide users with all the relevant knowledge they need to make decisions and formulate their opinions accurately.

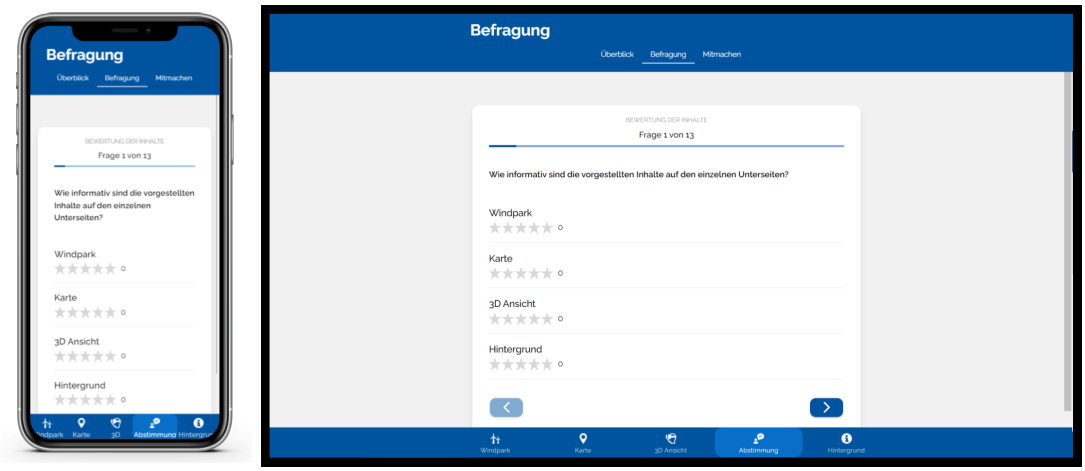

Figure 29: Opinion poll page for user engagement and collecting feedback.

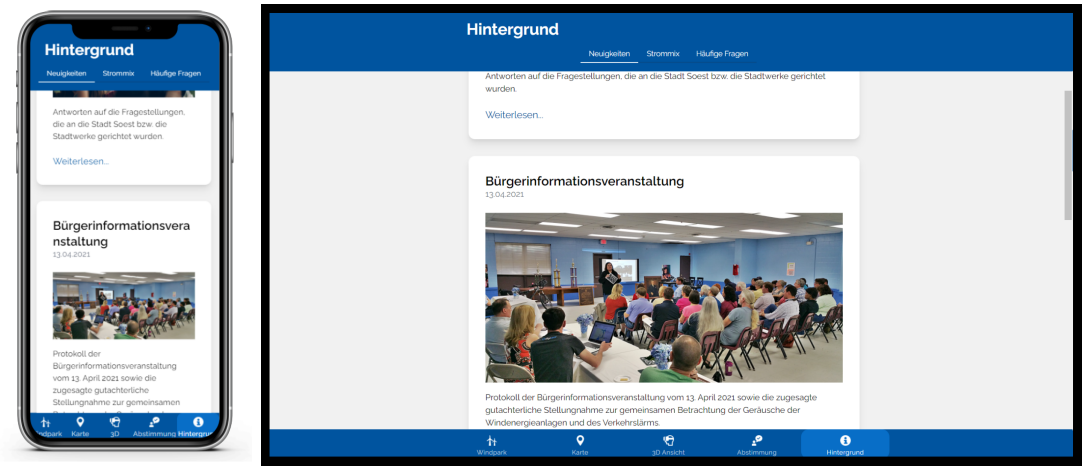

Figure 30: Latest wind farm news page.

6.6.1 News

In this page (Figure 30), the latest news are posted, and are redirected to the respective full article on clicking read more.

6.6.2 Strommix

Here 31, data about electricity sources distribution and CO2 emissions are collected and visualized in an informative manner that gives users an idea of how the wind park will have an impact on the current situation.

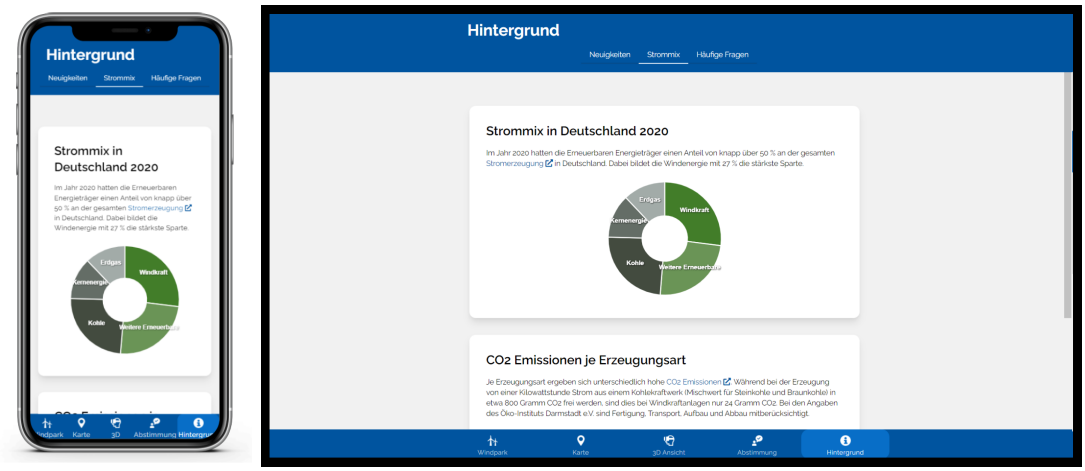

Figure 31: Electricity and CO2 emissions information page.

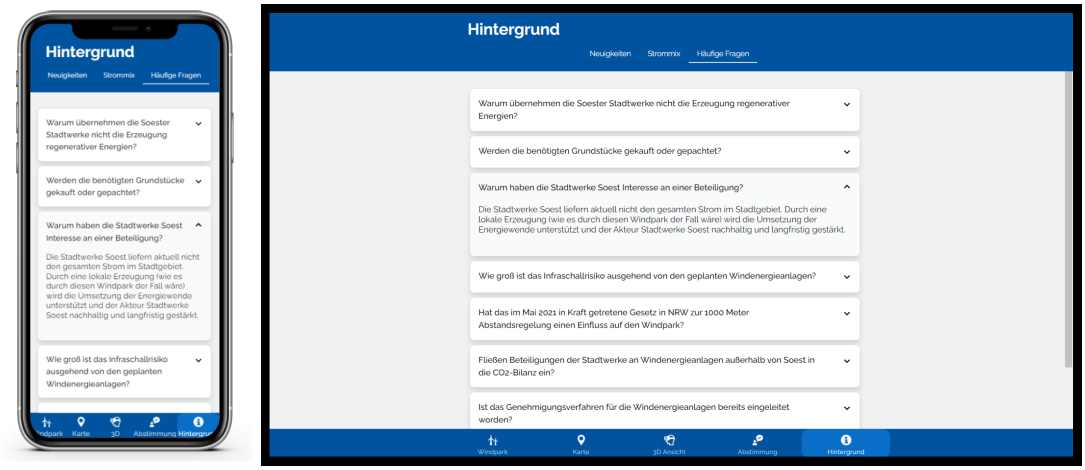

Figure 32: Frequently asked questions page.

6.6.3 FAQs

This page, demonstrated in Figure 32, lists some of the frequently asked questions collected from different users as they might also come to the users' minds while using the app.

7 Conclusion

The goal of this work is to develop an app that informs users with wind park plans in their city through interactive visualizations of the aspects that raise their concerns. This is done by utilizing different technologies and components that serve different purposes in delivering data resulted from dedicated algorithms in a presentable manner, be that animated visualizations in the form of virtual and augmented reality or 2D

interactive map visualizations that demonstrate how shadow casts would look like in real life and give users the opportunity to compare it to their addresses and assess the situation from their prespective. Giving the users the interactive possibility of switching between different scenarios of the construction plans will prompt them to engage in making a preference, which we believe was a strong and powerful option that comes as a beneficial aspect for our goals.

After achieving the informing goal, the engaging and participating goal follows, where we were keen on involving users as much as possible through a two-way communication facilitated using the opinion polls that prompt them to share their feedback and insights on what they liked and disliked, what their favorite scenarios are, etc... These information can then be analyzed and assessed to enable the relevant stakeholders to make informed decision that involve the citizens and not just a one-sided prespective.

The accessibility and simplicity nature of the app reduces the information gap between users and project planners, since users who are citizens of the city where the planning is to take place, are not necessarily from a technical background. That's why every aspect of the app was designed and developed to suit the widest possible range of target groups, taking into consideration the different channels they will be accessing the app from. As wind energy production is a relatively new concept for many, the app would hopefully impact citizens into accepting the concept of wind park constructions, and ease the way for the energy transition goals and help fight climate change.

7.1 Future work

This work has so much potential for additional integrations and still has a lot of room for improvement. As the entire development process was done with goals like maintainability, scalability, and reusability in mind, the app can be extended easily and smoothly without the risk of breaking existing components. This work carves the path for creativity and innovation to fulfill the bigger of goal of transparently informing users about wind park plans through interactive visualizations using modern and cutting-edge technologies. The following are some proposed next steps which could be implemented to refine the app's overall performance and capabilities as well as introduce new behaviors and features to it:

1. Improve overall overall responsiveness behavior by testing on more different devices and operating systems.
2. Refine interactiveness in sections where user engagement is desired to keep users in furthering their exploring of the app.
3. Incorporate more UX laws and ensure their overall implementation throughout the app.

4. Implement a better structure for the JSON file discussed in Section 5.2, as currently it's a single file with all infromation in it.
5. Implement more interactiveness in 3D visualizations such as virtual and augmented reality pages as they are currently a one-purpose page with limited options for actions to perform there.
6. Visualize more of the results of offline calculations to give users more information in an understandable manner.
7. Display more complex and interactive visualizations of wind turbine models so that users have a better understanding of how they function regardless of their background.
8. Provide app in multiple languages.

References

[1] Dirk Baumer, W Bischofberger, Horst Lichter, and Heinz Zullighoven. User interface prototyping-concepts, tools, and experience. In *Proceedings of IEEE 18th International Conference on Software Engineering*, pages 532–541. IEEE, 1996.

[2] Robert Berry, Gary Higgs, Richard Fry, and Mitch Langford. Web-based gis approaches to enhance public participation in wind farm planning. *Transactions in GIS*, 15(2):147–172, 2011.

[3] GW Boguslavsky and ER Guthrie. The recall of completed and interrupted activities: an investigation of zeigarnik's experiment. *Psychol. Bull*, 38:575–576, 1941.

[4] Tom Brinck, Darren Gergle, and Scott D Wood. *Usability for the web: Designing web sites that work*. Elsevier, 2001.

[5] Souma Chowdhury, Jie Zhang, Achille Messac, and Luciano Castillo. Optimizing the arrangement and the selection of turbines for wind farms subject to varying wind conditions. *Renewable Energy*, 52:273–282, 2013.

[6] Auriol Degbelo, Jan Kruse, and Max Pfeiffer. Interactive maps, productivity and user experience: A user study in the e-mobility domain. *Transactions in GIS*, 23 (6):1352–1373, 2019.

[7] Gerald Dekker, Qiuhao Zhang, John Moreland, and Chenn Zhou. Marwind: mobile augmented reality wind farm visualization. In *Proceedings of the International Conference on Modeling, Simulation and Visualization Methods (MSV)*, page 1. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2013.

[8] Anna Eteläaho, Teemu Kumpumäki, Jari Turunen, Tarmo Lipping, and Anne Nummela. Wind-farm visualization in western finland. *Journal of Architectural and Planning Research*, pages 91–106, 2015.

[9] Paul M Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology*, 47(6):381, 1954.

[10] Brett S Gardner. Responsive web design: Enriching the user experience. *Sigma Journal: Inside the Digital Ecosystem*, 11(1):13–19, 2011.

[11] Andrew S Glassner. *An introduction to ray tracing*. Morgan Kaufmann, 1989.

[12] Chu Hiang Goh and Nurul Hanim Romainoor. User goals, behaviours and attitudes: Developing web user personas of art and design students. *Art and Design Review*, 7(1):1–9, 2018.

[13] Aarush Gupta and Abdul Gaffar H. Hybrid application development using ionic framework & angularjs. *International Journal of Innovative Research in Computer Science & Technology (IJIRCST)*, 4(2), 2016.

[14] Bund/Länder-Arbeitsgemeinschaft Immissionsschutz. Hinweise zur ermittlung und beurteilung der optischen immissionen von windkraftanlagen. `https://www.lai-immissionsschutz.de/documents/wka_schattenwurfhinweise_stand_23_1588595757.01`, 2019.

[15] Fraunhofer ISE. German wind energy in numbers. `https://www.wind-energie.de/english/statistics/statistics-germany/`, 2020.

[16] Clemens Jauch, Julija Matevosyan, Thomas Ackermann, and Sigrid Bolik. International comparison of requirements for connection of wind turbines to power systems. *Wind energy*, 8(3):295–306, 2005.

[17] Jian Mao, Hanjun Ma, Yue Chen, Yaoqi Jia, and Zhenkai Liang. Automatic permission inference for hybrid mobile apps. *Journal of High Speed Networks*, 22 (1):55–64, Feb 2016. ISSN 18758940, 09266801. doi: 10.3233/JHS-160538.

[18] Adrian Mendoza. *Mobile user experience: patterns to make sense of it all*. Newnes, 2013.

[19] Jakob Nielsen. *Usability engineering*. Morgan Kaufmann, 1994.

[20] Stephen E Palmer. Common region: A new principle of perceptual grouping. *Cognitive psychology*, 24(3):436–447, 1992.

[21] Lucy Y Pao and Kathryn E Johnson. A tutorial on the dynamics and control of wind turbines and wind farms. In *2009 American Control Conference*, pages 2076–2089. IEEE, 2009.

[22] Tom Priestley. An introduction to shadow flicker and its analysis. `https://windexchange.energy.gov/files/pdfs/workshops/2011/webinar_shadow_flicker_priestley.pdf`, 2011.

[23] A Retik. Visualization for decision making in construction planning. *WIT Transactions on Information and Communication Technologies*, 5, 1970.

[24] Kari Sandouka. Interactive visualizations: A literature review. *MWAIS 2019 proceedings*, 8, 2019.

[25] Peter Shirley and R Keith Morley. *Realistic ray tracing*. AK Peters, Ltd., 2008.

[26] Chauncey Wilson. *User experience re-mastered: your guide to getting the right design*. Morgan Kaufmann, 2009.

[27] Bluma Zeigarnik and Kurt Lewin. *Das Behalten erledigter und unerledigter Handlungen*. Verlag nicht ermittelbar, 1927.

[28] Jonathan Zong, Dhiraj Barnwal, Rupayan Neogy, and Arvind Satyanarayan. Lyra 2: Designing interactive visualizations by demonstration. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):304–314, 2020.