Accessible Weather Forecast Visualization

Abigail Albuquerque, Kaley Du, Justin Weintraub, Aria Yan

1 Overview and Motivation

Our project aims to create an accessible weather visualization, improving on current techniques used by popular weather apps, using live API calls and innovative visual and sound-based techniques.

As a part of the design, there is a map on the first page where the user is able to select the location they'd like to see the weather for. The second page contains data about said location's weather conditions. Our goals consisted of using the live API to create the weather visualization, and using accessibility frameworks to make sure the tool was accessible to all users. We also wanted to add features like animating the visualizations and adding sound-based effects, so our app had unique and interactive ways displaying data.

We differentiate ourselves from other weather applications due to our focus on accessibility with regards to the Chartability framework, which is a newer accessibility framework. Our experience will hopefully allow disabled and blind or low-vision users navigate weather conditions, while still providing a good and digestible experience to nondisabled users.

2 Introduction

An estimated 15% of the world's population experiences some form of disability. They face barriers in regular life, including access to information and communication technology [5]. Weather is a very commonly visualized type of data and weather apps are some of the most commonly used across the globe. The data is usually visualized in the form of global maps, charts, heatplots, etc [7].

We aim to address this issue by evaluating existing weather apps and developing a more accessible weather visualization tool. Drawing from recent research, our goal is to create a platform that offers comprehensive weather information while being inclusive to users with disabilities.

Our primary focus is on informing a user regarding the weather across locations in the US . We aim to provide information on various aspects such as temperature, weather conditions, wind speed, etc. This will be achieved through a combination of accessible visual and sound-based techniques, offering users a multi-sensory experience.

Additionally, we will learn to work with live API calls, and learn more about how to create accessible visualizations and combine audio and visual components for the same. We hope to bridge the gap between weather visualization and accessibility.

Analyzing apps using Chartability's framework, we found that the most common problem with accessibility tools was that finding information regarding the weather simply through accessibility tools like VoiceOver functionality was extremely tedious. Further, despite being tech-savvy, we found it very hard to navigate the functionality. We realized that a lot of visual components of the apps are being read unnecessarily as well. Through our efforts, we hope to improve this design and make finding information simpler and less tedious.

Chartability's tests are split under the following categories: perceivable, operable, understandable, robust, compromising, assistive, and flexible. For perceiving, we need to make elements visually distinct and easy to see. For operability, we have to make sure our app is usable through screen readers, by low vision people, and having the apps operable by methods such as text to speech and keyboard inputs. For understandability, we need to make sure there's a lot of informative explanations. We need to ensure our app runs smoothly for the user for robustness and assistiveness. For the compromising category, we need to consider adding a table for the data in addition to the sound/background based visualizations. And for flexibility, we have to make sure our app works for

different users and for different screen sizes.

3 Related Work

The paper "Designing Accessible Visualizations: The Case of Designing a Weather Map for Blind Users" by Dustin Carroll, Suranjan Chakraborty and Jonathan Lazar [1] is closely related to ours, having also researched into the topic of accessible weather maps. This project was based on the AISP framework (a framework that aligns using auditory feedback to assist blind users), providing updates to use a sense of touch for the user. Like us, this paper tries to make a map easily navigable to the visually impaired user, prevents statistics on the map, and gives audio queues on weather conditions. We hope to enhance the findings of this study by improving on it's audio capabilities through thurough screen reader explanations and audio queues for weather that create a story over time. We also implemented the Chartibility framework for our testing to see how the accessibility measures up in a different field.

The paper "Visualization in Meteorology—A Survey of Techniques and Tools for Data Analysis Tasks" [6] by Rautenhaus et al. provides an overview of existing visualization techniques. It discusses various areas of most interest and need. It also touches upon interactive visualization and new methods which we could consider incorporating in our visualization. This will provide us with a guide for best practices when it comes to visualizing weather and serve as a starting point for understanding various other challenges.

The paper "How accessible is my visualization? Evaluating visualization accessibility with Chartability" by Frank Elavsky, Cynthia Bennett, Dominik Moritz [2] provides an accessibilty evaluation framework known as chartability. This is a framework we can use to evaluate the accessibility of our weather app. This framework includes 7 main principles: Percievable, Operable, Understandable and Robust, from the preexisting POUR principles, in addition to Compromising, Assistive, and Flexible. Among these principles are 10 critical heuristics and 35 non-critical heuristics. Chartability's workbook, which describes how to perform tests in detail, is used to perform Chartability's tests. We will try to evaluate our visualization on this framework so that all of the critical requirements are met. As a newer framework that is based off of research of existing guidelines, it helps us find and fix any failures we may have in the accessibility of our visualization.

The paper "Accessible Visualization: Design Space, Opportunities, and Challenges" by N. W. Kim, S. C. Joyner, A. Riegelhuth, Y. Kim [3] includes an analysis of papers published in the last 20 years before this paper was published regarding accessibility in visualization. The authors mapped a design space for accessibility that includes: user group, literacy task, chart type, interaction, information granularity, sensory modality, assistive technology. They also discuss current knowledge gaps, opportunities, and challenges. This will be a good overview that helps us get an empirical view of what is going on in the accessiblity visualization space. It's framework will be useful as Chartability mentions that it should not be used in place of a compliance audit but alongside it.

The paper "Development and Evaluation of Sonified Weather Maps for Blind Users" by Weir et al.[8] describes methods to sonify weather information for blind users. It draws from the Web Content Accessibility Guidelines (WCAG) from the Web Accessibility Initiative. This will be useful as an existing example of using sound to make weather visualizations accessible. Though it may be entirely auditory, we can take elements from it and apply it to the Web Content Accessibility Guidelines along with the Chartability framework in our own project.

The paper "Accessible Visualization: Design Space, Opportunities, and Challenges" by Kim et al. [4] provides a keyworded database for accessible visualization papers for the blind, visually impaired, and low vision people. It also analyzed said paper for their topics to see what worked and what methodology was used. Lastly, the paper wishes to propose methods to better help blind and low vision users. We used their proposed methods for enhancing accessibility as guidelines for our project among our other metrics.

4 Questions

Here are some of the questions we tried to answer through this project: How can we create a fully accessible weather visualization for the visually impaired? How can we best integrate real time API data generation into an accessible weather visualization? How can we tell a story through data visualizations? How can we create concise summaries for our visualizations? We explore these questions below.

1. How can we create a fully accessible weather visualization to the visually impaired?

We decided to audit our accessibility using the Web Content Accessibility Guidelines (WCAG) along with the Chartability framework. Originally, we were planning on creating a visualization that was mostly

sound-based, so the user can tell the weather for a location from audio cues. We realized we needed to be thorough in describing all the data and visualization, which presented us with a challenge as to how we'd do so. We addressed this problem by incorporating both visual and audio elements and following WCAG guidelines, creating a table of information, making screen reader accessible information and making sure to include appropriate labels and auditory descriptions for each component and graph on the application. For the map, we realized a significant challenge was getting a visually-impaired user to select a location. Our first thought was to allow the user to type a search query in the map to get the location. That led us to think about another question: How do we guide a visually impaired user towards the intended features? That led us to having the app send audio messages to the user at the start to guide them towards the search box for the map. We faced a challenge with having a screen reader find the search bar, so we wanted to make it as intuitive and accessible as possible.

2. How can we best integrate real time API data generation into an accessible weather visualization?

One of the most important aspects of our app is its real time functionality. However, with regards to accessibility, it is easier as a developer when the data is static, so any alternative text can be written and does not need change as the app does not change. However, with dynamic data, we had to learn how to work through creating alternative text that was dynamic as well. We used summarization through an LLM for the same.

3. How can we tell a story through data visualizations?

At first, we thought of implementing a completely visual application, one that showed a story through the background and was more artisitic in nature. After feedback from the professor, we switched to a display consisting of multiple graphs, and that led us to questioning how to best do this graphically. We wanted to keep the animation aspect, so we decided to try to integrate animations into the graphs. We also kept up the integration with sound, so we could tell a story through multiple methods.

4. How can we create concise summaries for our visualizations?

Given all of the information from the weather API, we needed a way to provide summaries to those with low-vision or blind users. We decided to use the ChatGPT API to accomplish this. We looked into how to implement this API and how to send prompts to generate the appropriate data. There were some issues with us not being able to use the ChatGPT API key on a public repository, so we unfortunately had to have a more limited, demo version of this feature.

5. How do we make the visualization less tedious to navigate for blind and low vision users?

Given the options of only tabbing through the website or using a screen reader, there are certain barriers that can make a visualization difficult to navigate through with these methods. Part of our primary goal was to address and fix issues that make such navigation difficult. We had to look into how to do screen reader testing and testing tabbing without mouse or other input. Providing sufficient and clear descriptions using aria labels, and using audio cues to communicate changes are some other requirements of heuristics listed in the Chartability workbook, which we had to go through systemically.

5 Data

We use data from multiple sources for our visualizaitons. For the first page of the app, the data for the map component and search box is from the Google Maps API and the Google Places API. In using these APIs we had to work through challenges such as restricting the map to only the United States, making the map clickable and making it draggable, adding a search bar, etc. When searching for places, the data gotten is an object that we have to parse to get the lat/lng/name, which was all relatively easy.

5.1 National Weather Service API

Our main data source for the weather data is from the National Weather Service API. The National Weather Service API is based upon JSON-LD to promote machine data discovery.

5.1.1 /points

The first endpoint we used is the /points endpoint. We used this to find the grid or zone a given coordinate point is in on the national weather service. This is important because the National Weather Service splits up the United States into small polygons, each one which has a corresponding set of data and weather observatory. Since each polygon is slightly different, we can't calculate which polygon a given coordinate point is in ourselves, which is what this API is good for. We can then get forecast box and also the zone that the coordinate points are in.

Here's an example of the properties given to us from the endpoint, where we can use the forecastHourly URL and forecastZone URL to get other data.

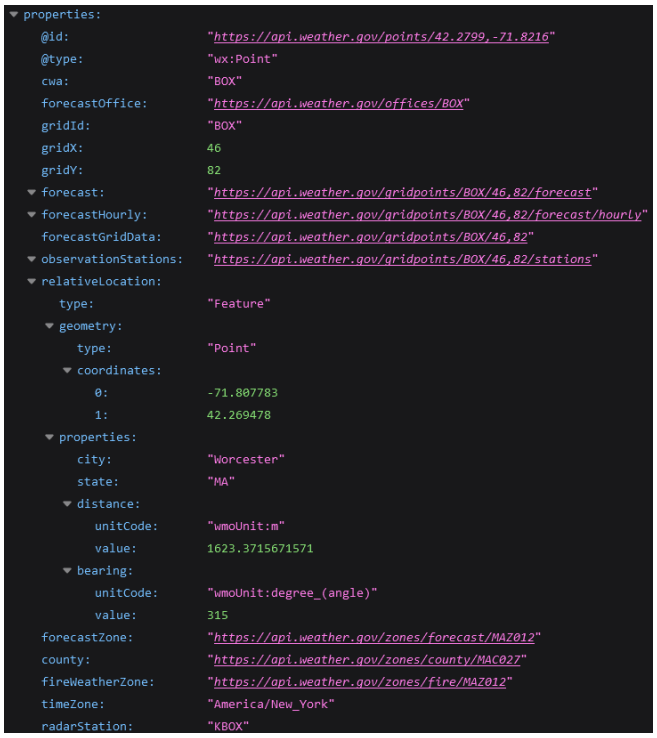

Figure 1: Properties given from /points endpoint

5.1.2 /gridpoints/BOX/-/forecast/hourly

The second endpoint we used is the hourly forecast for a given gridpoint. We get this endpoint from the /points endpoint. This gives us hourly forecasts, or periods, for a week. Each hour, we get information on the start time, end time, whether it's daytime or not, temperature, probability of precipitation, dewpoint, relative humidity, wind speed, wind direction, a short forecast, and an image. This is our main data source. Out of these different properties, the ones we ended up using in the app is start time, temperature, probability of precipitation, wind speed, wind direction, short forecast, and the image.

Here's an example of one period of data from the API:

And here's an example of the image of the short forecast given to us from the weather API:

5.1.3 /alerts/active?zone=

The final endpoint we used from the National Weather Service is the /alerts endpoint. We filter through all the alerts for active ones and find the corresponding zone for our coordinate point. This corresponding zone is given to us from the /points endpoint. The /alerts endpoint gives shows all the weather alerts for a given area, along with all impacted zones, a long description, and instructions for what to do. Since we didn't want to bombard the user with the long description, we decided to just show the name of the alert and the instruction for the user. This is what the result for one weather alert looks like:


```
{
    "number": 1,
    "name": "",
    "startTime": "2024-04-28T14:00:00-04:00",
    "endTime": "2024-04-28T15:00:00-04:00",
    "isDaytime": true,
    "temperature": 63,
    "temperatureUnit": "F",
    "temperatureTrend": null,
    "probabilityOfPrecipitation": {
        "unitCode": "wmoUnit:percent",
        "value": 3
    },
    "dewpoint": {
        "unitCode": "wmoUnit:degC",
        "value": 9.4444444444444446
    },
    "relativeHumidity": {
        "unitCode": "wmoUnit:percent",
        "value": 60
    },
    "windSpeed": "9 mph",
    "windDirection": "SW",
    "icon": "https://api.weather.gov/icons/land/day/bkn,3?size=small",
    "shortForecast": "Partly Sunny",
    "detailedForecast": ""
},
```


Figure 2: One period of data

Figure 3: Image of Short Forecast

Figure 4: Weather Alert example

5.2 ChatGPT API

Since the data we are using is dynamic, we used ChatGPT's API to generate data summaries for the generic weather summary and for describing graph trends. We used GPT-3.5 Turbo since it was the most cost effective model. We evaluated the accuracy of the text results manually and didn't find any

error in the summaries given by ChatGPT.

5.2.1 General Summary

In order to create the general summary, we first created key information from the data we got from the National Weather Service API, then fed that into the ChatGPT API. For each day, we calculated the highest and lowest temperature of the day, the average windspeed, the most common wind direction, and the highest chance of rain. Each prompt that we gave ChatGPT had the request prompt and then the data. Here's an example of the data we gave ChatGPT:

> "April 28: Chance Showers And Thunderstorms with highs of 82 and lows of 67. The highest chance of rain is 51 with an average windspeed of 15 in the direction of S. April 29: Showers And Thunderstorms with highs of 83 and lows of 65. The highest chance of rain is 98 with an average windspeed of 7.916666666666667 in the direction of S. April 30: Chance Showers And Thunderstorms with highs of 83 and lows of 64. The highest chance of rain is 47 with an average windspeed of 8.26086956521739 in the direction of S."

We played around with several different prompts. We first started with "Summarize this to a paragraph," which resulted in this following paragraph.

> "From April 28 to April 30, the weather forecast includes a mix of showers, thunderstorms, and partly cloudy conditions. On April 28, expect a chance of showers and thunderstorms with highs around 79 and lows near 67. The highest chance of rain is 74%, accompanied by an average windspeed of 15 mph from the south. Moving on to April 29, showers and thunderstorms are likely with highs reaching 81 and lows dropping to 64. The chance of rain increases to 93%, with a lighter average windspeed of 5.625 mph from the south. By April 30, the weather will be partly cloudy with highs of 83 and lows of 63. The chance of rain decreases to 34%, and the windspeed averages around 5.87 mph from the southeast."

We then tried this prompt, because Chartability advises all text to be at a 9th grade reading level or below. "Convert this to a grammatically correct 9th grade reading level paragraph"

> "On April 28th, there is a chance of showers and thunderstorms with a high temperature of 82 degrees and a low of 67 degrees. The likelihood of rain is 51%, and the average wind speed is 15 mph coming from the south. Moving on to April 29th, there will be showers and thunderstorms with a high of 83 degrees and a low of 65 degrees. The chance of rain is highest at 98%, and the average wind speed is 7.92 mph from the south. Lastly, on April 30th, there is a chance of showers and thunderstorms with a high of 83 degrees and a low of 64 degrees. The probability of rain is 47%, and the average wind speed is 8.26 mph from the south."

However, after some user testing, we decided to prioritize the short length of the paragraph. We used the prompt 'Summarize this to a concise paragraph' and we found that the reading level was below 9th grade anyways.

> "Expect a mix of showers and thunderstorms on April 28 with a high of 79 and low of 67. Rain chances peak at 74% with average winds of 15 mph from the south. On April 29, more showers and thunderstorms are likely with a high of 81 and low of 64. Rain chances increase to 93% with lighter winds at 5.6 mph from the south. By April 30, conditions improve to partly cloudy skies with a high of 83 and low of 63. Rain chances drop to 34% with gentle southeast winds at 5.9 mph."

We also conducted similar testing for the prompts to describe the graph information. We wanted to make sure that the summary described the trend of data. Here's an example of the prompt we sent ChatGPT.

> Describe the temperature movement in concise sentences. Time: ["2024-04-28T22:00:00.000Z","2024-04-28T23:00:00.000Z","2024-04-29T00:00:00.000Z","2024-04-29T01:00:00.000Z","2024-04-29T02:00:00.000Z","2024-04-29T03:00:00.000Z","2024-04-29T04:00:00.000Z","2024-04-29T05:00:00.000Z","2024-04-29T06:00:00.000Z","2024-04-29T07:00:00.000Z","2024-04-29T08:00:00.000Z","2024-04-29T09:00:00.000Z","2024-04-

29T10:00:00.000Z","2024-04-29T11:00:00.000Z","2024-04-29T12:00:00.000Z","2024-04-29T13:00:00.000Z","2024-04-29T14:00:00.000Z","2024-04-29T15:00:00.000Z","2024-04-29T16:00:00.000Z","2024-04-29T17:00:00.000Z","2024-04-29T18:00:00.000Z","2024-04-29T19:00:00.000Z","2024-04-29T20:00:00.000Z","2024-04-29T21:00:00.000Z"] Temperature: [79, 78, 74, 71, 70, 69, 68, 67, 67, 66, 67, 65, 65, 65, 64, 66, 67, 71, 74, 75, 77, 79, 80, 81]'

ChatGPT was able to summarize each graph into one or two sentences.

> "The temperature is gradually decreasing from 79°F at 10:00 PM to 64°F at 1:00 PM the next day. It then starts to rise again, reaching 81°F by 9:00 PM."

6 Exploratory Data Analysis

Since our data was dynamically sourced from an API, and not from a static dataset, there was not much exploratory data analysis that needed to be conducted to determine parts of the data that needed cleaning or to discover trends in the data. However, we had to explore the return values from our API calls to determine how we could use the data returned to create meaningful weather visualizations. Our first step was to test the API calls to determine the format of the information that was returned. We received a JSON file, which we then had to parse through to structure the data into a format that would be useful to build our visualizations.

7 Design Evolution

Initially, our design for the application was primarily on one page. At the top of the page, we placed a map of the US, on which the user will be able to select a location in the US. A toggle button was shown asking users if they are visually impaired, which toggled the accessibility settings on the page such as having high-contrast and enabling automatic screen reading/summarization tools.

A page displaying specific weather information was on the bottom and included a back button, a box showing the weather details, a weather alert box, an artistic visualization of the weather combined with

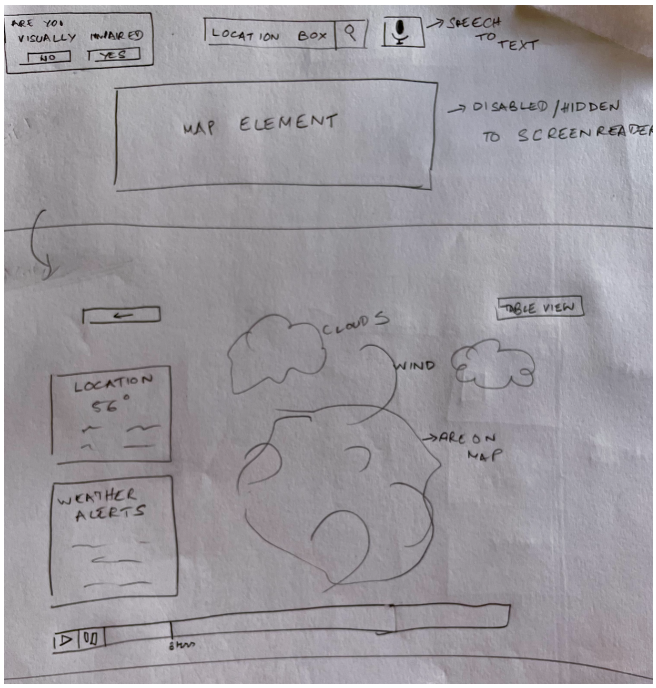

Figure 5: Early design of our application.

sound effects, and a time/animation slider at the bottom of the page that allows the user to play/pause the animation of the weather for the next three days. We also planned to provide a tabular format of the data.

We took inspiration from other weather application designs that have different square 'components' on their pages showing various information to users. For example, the Microsoft MSN weather application as seen in 6. In later iterations of our design, we got rid of the toggle button as we decided we wanted the app to be accessible to everyone without additional steps like having to switch an accessible mode on and off. We also refined what we wanted on the weather

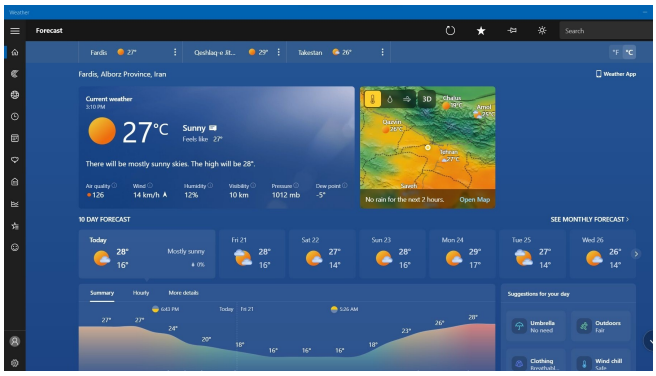

Figure 6: Inspiration from other weather applications like Microsoft MSN's weather.

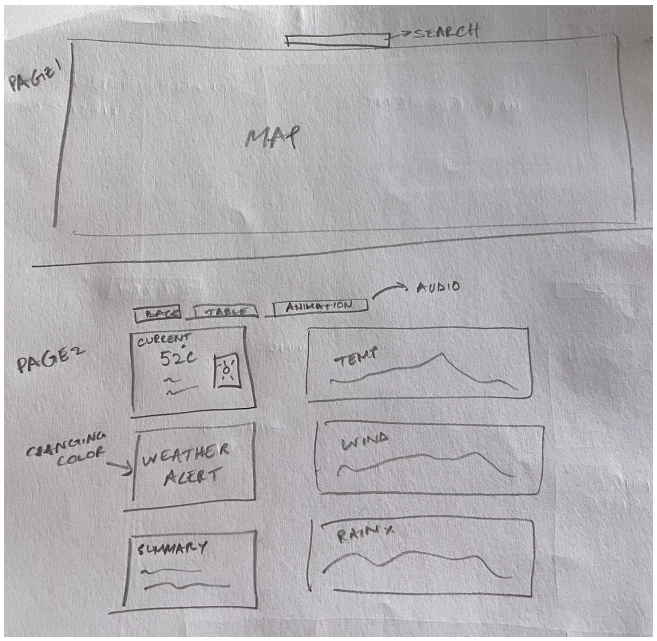

Figure 7: Final paper design of our application. Some stylistic changes were made during the development process.

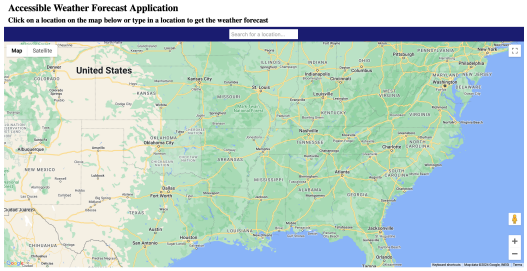

Figure 8: The first page of our application showing the map component.

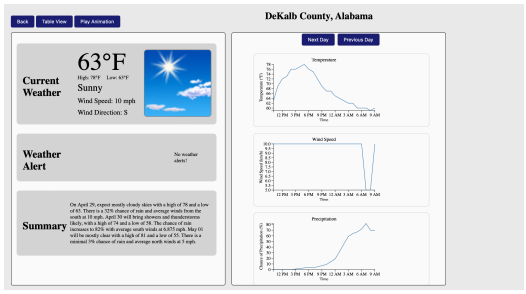

Figure 9: The second page of our application.

information page. We got rid of the slider and instead had three line graphs vertically stacked showing the temperature, wind speed and precipitation throughout the day.

We ended up dividing the map and weather information into two pages. This was so the map could be larger for easier selection of location. We also added a search bar so the user could just type in a location as well. You can see one of our final paper designs in 7.

Aspects we kept were speech-to-text capabilities and the summarization capability with chatGPT. After a location is selected, a new page will display the weather in the area for approximately the next 3 days. The thought behind this design change was to allow for user accessibility without the tediousness that comes with accessibility with most websites.

8 Implementation

We used several different tools to build our accessible weather application. The backbone of the tool including the pages, buttons and accessibility features were built using JavaScript and Node.js. We used the Accessible Rich Internet Applications (ARIA) features within JavaScript to make our applications tabable and accessible. The map feature was added into the application using Google's map API. We used d3.js to build the line charts as well as implement the animation features. The accessibility feature of summarizing the charts and other weather information was implemented using openAI's API. We used various screen readers to test our application through the development process including NVDA, a chrome extension simply called "Screen Reader".

The main intent for our visualization was to create an easily accessible way to view weather for both visual and non-visual users. The key interactive elements are the map feature, allowing the user to choose their location to view the weather which can be seen in Figure 8.

Additionally, all the information seen in Figure 9 is dynamic and shows the status of weather in the user selected location. The user is also able to click on the play animation button on the top of the page to show the animated line chart that has a sound component, depicting weather conditions through sound. Unfortunately, a lot of our features are on the backend, and relate to accessibility within data visualizations. Thus, we are not able to display them visually here. These features include enabling tabbing through elements, creating summaries of our visuals through the use of LLMs to be read out by a screen reader, and addition of sound-based elements depicting weather condition to the animation.

9 Evaluation

In this section, we highlight the evaluation of multiple aspects of our application, the most important of which are related to the application's accessibility features.

9.1 Lighthouse Audit, WCAG Guidelines and Chartability

We used the Web Content Accessibility Guidelines (WCAG) 2 guidelines as an accessibility audit and addressed failures with the Chartability framework.

For the WCAG guidelines, the website had to meet 4 main categories: Percievable, Operable, Understandable and Robust.

We also used the Lighthouse Chrome extension to audit issues. One error this caught was that the html tag did not have a language attribute. Some manual checks were recommended, and ones that were relevant included:

- Interactive controls are keyboard focusable
- Interactive elements indicate their purpose and state
- The page has a logical tab order
- Visual order on the page follows DOM order
- User focus is not accidentally trapped in a region
- The user's focus is directed to new content added to the page
- HTML5 landmark elements are used to improve navigation
- Custom controls have associated labels
- Custom controls have ARIA roles

We were able to fulfill most these as we already included the purpose and state as well as labels and ARIA roles for many interactive elements and ordered them in a way that makes sense.

To fulfill these recommendations, we had to conduct keyboard probing and screen reader testing. We found through keyboard probing that it would skip over some parts of the website that contained information, so the screen reader would not be able to read it out. We corrected this by adding a tag to indicate these parts should be tabbed to.

After conducting evaluation with WCAG, the following changes needed to be made:

We had to check contrast and fix any inaccessibility issues related to low contrast. On the starting page, the color of the padding around the search bar was changed as it was too low contrast. In addition, we changed the sound accompanying animations to be equal to or less than 3 seconds to fulfil 1.4.2 Audio Control so that the sound does not impede the screen reader.

After conducting evaluation with Chartability, we made changes corresponding to the following heuristics:

- Perceivable Heuristic: Small text size (critical) The text on the graphs on the weather page was too small. We changed the font size to be over 12px.
- Operable Heuristic: No interaction cues or instructions (critical) Originally, we did not have as much instructions on how to use the app. We were going to add an about page, but decided against it as it may reduce accessibility to have to navigate to a different page in order to get such instructions. Having an information icon and popup would also involve other accessibility considerations. We added a title and short summary to the top of the landing page that gives instructions on how to proceed from the landing page. We also added tags to the html throughout the landing page and the weather page that included extra explanation that would be read out by the screen reader.
- Understandable Heuristic: No explanation for purpose or for how to read (critical) We needed to add a bit more detail into how to read the line graphs. We added tags that included audio descriptions for each of the graphs.

We meet most A and AA level requirements for WCAG, and all critical heuristics for Chartability. We also tested tabbing through a website with no mouse input and tested with a screen reader.

9.2 User Testing

9.2.1 Procedure

We conducted some user testing with two students. These students had not used the website before. We used the NVDA screen reader during testing on a laptop and asked them to complete a series of tasks while they had their eyes closed. We asked them to get weather information for a location of their choice. Then, we asked them to find weather trend information on the weather app. We then asked them to find weather information for another location. Finally, we

asked them to give feedback on the app and the information they learned.

9.2.2 Website Feedback

Generally, both testers were able to get general weather information about a location of their choosing. They liked the weather information we provided. We observed some confusion while they were listening to the controls. They reported being overloaded with information at the start of both pages and having difficulty keeping track of their options and controls. After navigating to the weather page, one tester reported wanting the weather summary to play before receiving all the information about potential buttons. To learn about weather trends, one tester preferred the weather graph summaries for its conciseness, while the other preferred the data table. Both testers reported having difficulty hearing the change in audio levels for the graph animations, so we made that more dramatic and scaled the volume relatively. When one tester navigated to the data table, they were confused by what the "mode" buttons did, so we changed the description of those buttons to hopefully be more clear.

Both testers also reported wanting the option to skip the audio, especially because the summaries were too long and tedious at times. The screen reader we used had the ability to pause the audio, but not to skip. They were also confused by the order of the button tabs. This was because, during user testing, we had the weather information first in the tab order before the buttons for other modes, so they had difficulty finding the other buttons.

9.2.3 Other Notes

While both tester were used to using laptops, they were not as familiar with the keyboard. Thus, they both had trouble typing with their eyes closed and finding the "Tab" button quickly. We believe a user who is actually vision-impaired would not have these difficulties using the keyboard. One tester recommended that we add "real feel" data. While the National Weather Service API did not provide this data, we could add this in the future by calculating the temperature and windspeed, among other things.

9.3 What Can We Do to Improve?

Our application is in a relatively complete state, so improvements would come in the form of additional features and styling.

In terms of additional features, we have a few to mention. We want to potentially expand the map to show features such as heat, be usable in areas not just in the US, and make searching easier. In our conceptual stage, we were considering adding a heat map to better describe to the user the current situation before a location was selected. Expanding outside the US would be difficult due to not being able to use our current weather API, but the more locations the better. At the moment, searching is just a one and done thing, limiting exploration. We'd like to also improve searching for blind folk, making it more obvious when they're focused on an editable textbox in the search field. A random location feature could also be cool on the map.

Outside of the map, we were considering some other features on the weather page. For one, an option to expand the day range would be nice to tell more data to the user. We'd also like to potentially implement our original idea of background that tells a story. Stylistically, we'd like to improve the graphs and the text through implementing color to better emphasize information.

10 Conclusion

Our project showed us both the simplicity and complexity associated with integrating accessibility features within visualization applications. While some things were easy, such as tab indexing, other aspects were hard, such as implementing filtering and summarization to the dynamic data. We hope that our work in this app can showcase how much information can be shown to both the visually impaired and the sightful, and showcase the power of the telling a visualization with multiple senses.

References

[1] Dustin Carroll, Suranjan Chakraborty, and Jonathan Lazar. Designing accessible visualizations: The case of designing a weather map for blind users. volume 8009, pages 436–445, 07 2013.

[2] F. Elavsky, C. Bennett, and D. Moritz. How accessible is my visualization? evaluating visualization accessibility with chartability. *Computer Graphics Forum*, 41(3):57–70, 2022.

[3] N. W. Kim, S. C. Joyner, and Y. Kim A. Riegelhuth. Accessible visualization: Design space, opportunities, and challenges. *Computer Graphics Forum*, 40(3):173–188, 2021.

[4] N. W. Kim, S. C. Joyner, A. Riegelhuth, and Y. Kim. Accessible visualization: Design space, opportunities, and challenges. *Computer Graphics Forum*, 40(3):173–188, 2021.

[5] World Bank & World Health Organization. World report on disability: Main report, 2011.

[6] Marc Rautenhaus, Michael Böttinger, Stephan Siemen, Robert Hoffman, Robert M. Kirby, Mahsa Mirzargar, Niklas Röber, and Rüdiger Westermann. Visualization in meteorology—a survey of techniques and tools for data analysis tasks. *IEEE Transactions on Visualization and Computer Graphics*, 24(12):3268–3296, 2018.

[7] K. Riley, D. Ebert, C. Hansen, and J. Levit. Visually accurate multi-field weather visualization. In *IEEE Visualization, 2003. VIS 2003.*, pages 279–286, 2003.

[8] R. Weir, B. Sizemore, H. Henderson, S. Chakraborty, and J. Lazar. Development and evaluation of sonified weather maps for blind users. In Patrick Langdon, John Clarkson, Peter Robinson, Jonathan Lazar, and Ann Heylighen, editors, *Designing Inclusive Systems*, pages 75–84, London, 2012. Springer London.