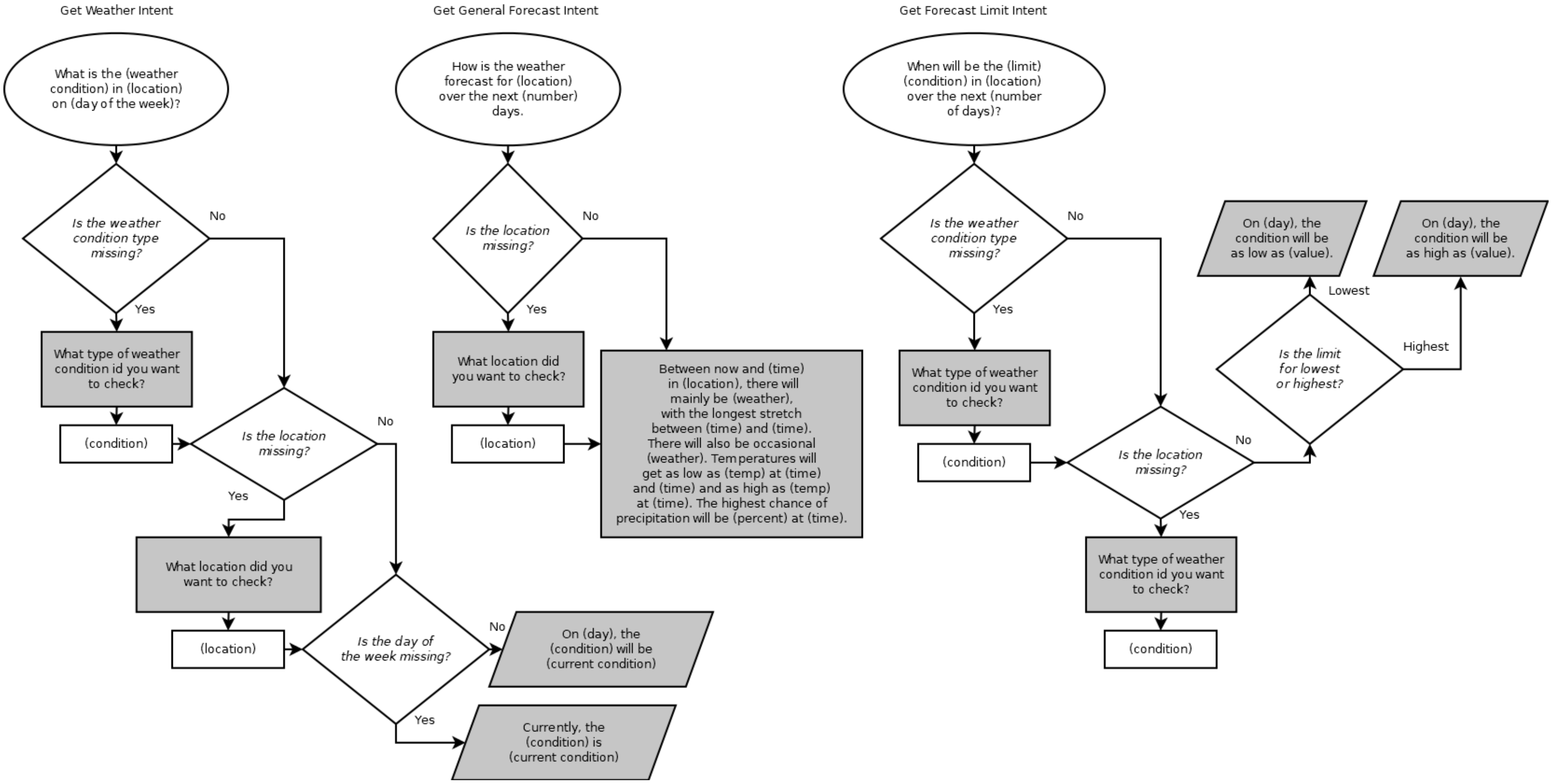
Max Oakes
Final Project

Step 1: Application Description

I will be building a weather forecast assistant using Dialogflow. I wanted to build an app that I could realistically use. The app will utilize Openweather API, and will allow the user to:

- Get a current weather condition at any global location
- Get the general weather trend over the next 5 days for a location
- Get the highest or lowest weather condition (temperature, wind, visibility, etc) over the next 5 days for a location

The app will allow for checking of international locations for specific weather conditions. It should allow for one to easily see when the lowest or highest of a weather condition will be.



Get Weather Intent
What is the (weather condition) in (location) on (day of the week)?
Is the weather condition type missing?
No
Yes
What type of weather condition id you want to check?
(condition)
Is the location missing?
No
Yes
What location did you want to check?
(location)
Is the day of the week missing?
No
On (day), the (condition) will be (current condition)
Yes
Currently, the (condition) is (current condition)
Get General Forecast Intent
How is the weather forecast for (location) over the next (number) days.
Is the location missing?
No
Yes
What location did you want to check?
(location)
Between now and (time) in (location), there will mainly be (weather), with the longest stretch between (time) and (time). There will also be occasional (weather). Temperatures will get as low as (temp) at (time) and (time) and as high as (temp) at (time). The highest chance of precipitation will be (percent) at (time).
Get Forecast Limit Intent
When will be the (limit) (condition) in (location) over the next (number of days)?
Is the weather condition type missing?
No
Yes
What type of weather condition id you want to check?
(condition)
Is the location missing?
No
Yes
What type of weather condition id you want to check?
(condition)
Is the limit for lowest or highest?
Lowest
Highest
On (day), the condition will be as low as (value).
On (day), the condition will be as high as (value).

(Will be uploaded separately in case it is too small)

Step 2: Interaction Model

Intents:

- GetWeatherIntent
    1. How is the current {Condition} in {Location}?
    2. How is the {Condition} currently in {Location}?
    3. How is the {Condition} in {Location} right now?
    4. How is the {Condition} in {Location}?
    5. How is {Condition} in {Location} right now?
    6. What is the current {Condition} in {Location}?
    7. What is the {Condition} in {Location}?
    8. What is the current {Condition} for {Location}?
    9. What is the {Condition} in {Location} right now?
    10. What is the {Condition} in {Location} currently?
    11. What is the {Condition} like in {Location} right now?
    12. What is the {Condition} like in {Location}?
    13. What is the active {Condition} in {Location}?
    14. What’s the current {Condition} in {Location}?
    15. What’s the {Condition} in {Location}?
    16. What are the current {Condition} in {Location}?
    17. What are the {Condition} in {Location}?
    18. What are the {Condition} in {Location} right now?
    19. In {Location}, how is the {Condition}?
    20. In {Location}, how are the {Condition}?
    21. In {Location}, what is the weather?
    22. What will the {Condition} be in {Location} {DayTime} at {Time}?
    23. How will the {Condition} be in {Location} {DayTime} at {Time}?
    24. {DayTime} at {Time}, how will the {Condition} be in {Location}?
    25. {DayTime} at {Time}, what will the {Condition} be in {Location}?
    26. How is the {Condition} in {Location} {DayTime} at {Time}?
    27. How is the {Condition} {DayTime} at {Time} in {Location}?
    28. What is the {Condition} {DayTime} at {Time} in {Location}?
    29. What is the {Condition} in {Location} {DayTime} at {Time}?
    30. What will the {Condition} be in {Location} {DayTime} at {Time}?
    31. What will be the {Condition} in {Location} {DayTime} at {Time}?
    32. How {Condition} will it get this week in {Location}?
- GetGeneralForecastIntent
    1. What is the forecast for {Location}?
    2. What is {Location}’s forecast?
    3. What is the weather forecast for {Location}?
    4. What is the forecast for the next {DayCount} days for {Location}?
    5. What is the {DayCount} day forecast for {Location}?
    6. What is the {DayCount} day weather forecast for {Location}?

    7. What is the weather forecast for the next few days for {Location}?
    8. What is the weather forecast for the next {DayCount} days for {Location}?
    9. What is the forecast over the next few days for {Location}?
    10. What is the forecast over the next {DayCount} days for {Location}?
    11. What will the weather be like in the next few days for {Location}?
    12. What will the weather forecast be like in the next few days for {Location}?
    13. What will the weather be like over the next {DayCount} days?
    14. What will the weather be like this week for {Location}?
    15. What will the weather be this week for {Location}?
    16. What will the {DayCount} day forecast be for {Location}?
    17. What will the forecast be for {Location}?
    18. What’s the weather forecast for {Location}?
    19. What’s the forecast for {Location}?
    20. What’s the forecast like for {Location}?
    21. How is the weather forecast for {Location}?
    22. How is {Location}’s forecast?
    23. How will the weather forecast be for {Location}?
    24. How is the forecast for {Location}?
    25. How is the weather forecast for {Location}?
    26. How is the forecast for the next few days for {Location}?
    27. How is the forecast over the next {DayCount} days for {Location}?
    28. How is the weather forecast over the next few days for {Location}?
- GetForecastLimitIntent
    1. What will be the {Limit} {Condition} in {Location} over the next few days?
    2. What will be the {Limit} {Condition} in {Location} over the next couple days?
    3. What will be the {Limit} {Condition} in {Location} over the next {DayCount} days?
    4. What will be the {Limit} {Condition} in {Location}?
    5. How {Limit} will the {Condition} in {Location} be over the next few days?
    6. How {Limit} will the {Condition} in {Location} be over the next couple days?
    7. How {Limit} will the {Condition} in {Location} be over the next {DayCount} days?
    8. How {Limit} will the {Condition} in {Location} reach over the next few days?
    9. How {Limit} will the {Condition} in {Location} reach over the next couple days?
    10. How {Limit} will the {Condition} in {Location} reach over the next {DayCount} days?
    11. What day will have the {Limit} {Condition} in {Location} this week?
    12. What day will have the {Limit} {Condition} in {Location}?
    13. When will be the {Limit} {Condition} in {Location}?
    14. When will be the {Limit} {Condition} in {Location} this week?
    15. When will the {Condition} be {Limit} in {Location}?
    16. When will the {Condition} be {Limit} in {Location} this week?
    17. Over the next {DayCount} days, when will be the {Limit} {Condition} in {Location}?
    18. Over the next few days, when will be the {Limit} {Condition} in {Location}?
    19. Over the couple days, when will be the {Limit} {Condition} in {Location}?

    20. When is the {Condition} expected to be {Limit} in {Location}?
    21. When is the {Limit} {Condition} this week in {Location}?
    22. When is the {Limit} {Condition} in {Location} over the next few days?
    23. When is the {Limit} {Condition} in {Location} over the next {DayCount} days?
    24. When will the {Condition} be the {Limit} in {Location}?
    25. When will the {Condition} be the {Limit} in {Location} this week?
    26. On what day will {Condition} be the {Limit} in {Location}?

Slots:

1. Location: @sys.location
2. Condition: Custom
    a. Overview, Snow, Pressure, Rain, Visability, Cloud cover, Wind, Temperature, Weather
3. Limit: Custom
    a. Highest, Lowest, High, Low, Worst, Best, Least, Most, Bad, Good
4. DayCount: Numbers
5. DayTime: @sys.date-time

Step 3: Fulfillment Logic

Get Weather Intent

- Get weather condition, time and location from the user
    - If no time is specified, use current time
    - If no weather condition or location is specified, ask the user to state missing required items
- Perform API call to get weather at the established date/time and location
- Get the weather condition value from that API response
- Return a string stating the weather condition value at the specified time and location

Get Forecast Intent

- Get the location, and optionally the number of days from the user
    - If no location is stated, ask the user to state it
- Perform API call to get the forecast for the user-specified location, up the number of days that the user said, or up to the limit of the API call if no number is specified
- From the API response, count the number of occurrences of each type of weather event
    - Get the two most common types of weather
- Get the duration of each stretch of weather event
    - Find the longest stretch of the most common weather type
- Find the date and time of the lowest and highest temperature, and highest chance of precipitation
- Output to the user in a sentence format:
    - The two most common types of weather
    - The longest stretch of the most common weather
    - The high and low temperatures and their times
    - The highest chance of precipitation and its time

Get Limit Intent

- Get the location, limit, weather condition and number of days from the user
    - If the location, limit, weather condition are not specified, ask the user gain
    - If number of days is not specified, use the maximum amount of time that the API response returns
- Perform the API call to get the forecast for the location
- Iterate over the API response’s weather data up the number of days specified
    - Find the (min or max) of that condition (eg, temperature, visibility, chance of rain) over that time
- Tell the user when that lowest or highest condition will occur at the specified location

Step 4: User Testing

	Control	User 1		User 2	
	Time to End	Completed?	Time to End	Completed?	Time to End
Find the current temperature for Boston.	12s/6s (Type/Speak)	Y	6.8s (speak)	Y	10.3s (speak)
Find the current cloud cover for London, England tomorrow at about 5pm.	12s/7s (Type/Speak)	Y	38.6s (speak, 3 attempts)	N	60s+
Find when the lowest visibility will be in Santiago, Chile this week.	14s/8s (Type/Speak)	Y	21.12 (speaking, restate loc)	Y	9.1s (speak)
Find the highest temperature in New Delhi over the next two days.	12s/7s (Type/Speak)	Y	9.2s (speak)	Y	9.87s (speak)
Get the 5-day forecast for Seattle.	9s/5s (Type/Speak)	Y	15.5s (speaking, two attempts)	Y	6.8s

Answers:

- What is the current temperature in Boston?
    - The temperature in Boston, Massachusetts is 69.1 degrees with a low and high of 61.1 and 76 degrees with humidity at 70%.
- What will be the cloud cover in London, England tomorrow at 5pm?
    - The cloud coverage in London, England will be 80% at 7:00 PM on Sunday.
- When will the lowest visibility be in Santiago, Chile this week?
    - In Santiago, Cartago Province at 3:00 PM on Wednesday, the lowest visibility will be 2358m.
- What will be the highest temperature in New Delhi over the next two days?
    - In New Delhi, Delhi at 2:30 PM on Monday, the highest temperature will be 111.8 degrees.
- What is the 5-day forecast for Seattle?
    - Between now and 5:00 PM on Thursday in Seattle, Washington, there will mainly be overcast clouds, with the longest stretch between 2:00 AM on Wednesday and 2:00 AM on Thursday. There will also be occasional rain. Temperatures will get as low as 44.6 degrees at 5:00 AM on Tuesday and as high as 73 degrees at 2:00 PM on Wednesday. The highest chance of precipitation will be 100% at 11:00 PM on Saturday.

User 1

	Questions	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1	I knew what I was able to ask and how to ask it.					X
2	The forecast's response was too long.		X			
3	The forecast's response was meaningful and useful.					X
4	When asking for current weather, I get enough information.					X
5	When asking for a minimum or maximum weather condition, I get all of the information that I wanted.				X	
6	I felt confident that the information I was receiving was correct.				X	
7	The date and time that I stated was poorly recognized.			X		
8	The phrasing of the responses seemed awkward or artificial.	X				
9	The format and data that I received was readable and easy to understand.					X
10	It was sometimes hard for the application to understand the location I was actually referring to.				X	
11	The application is generally useful.					X
a	What did you like about the application?	easy to use				
b	Are there any aspects of the application that have room for improvement?	Recognition of more locations				
c	What other weather-related features would be useful or interesting?	NA				

User 2

	Questions	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
1	I knew what I was able to ask and how to ask it.				X	
2	The forecast’s response was too long.				X	
3	The forecast’s response was meaningful and useful.				X	
4	When asking for current weather, I get enough information.				X	
5	When asking for a minimum or maximum weather condition, I get all of the information that I wanted.			X		
6	I felt confident that the information I was receiving was correct.				X	
7	The date and time that I stated was poorly recognized.				X	
8	The phrasing of the responses seemed awkward or artificial.			X		
9	The format and data that I received was readable and easy to understand.				X	
10	It was sometimes hard for the application to understand the location I was actually referring to.				X	
11	The application is generally useful.				X	
a	What did you like about the application?	seemed user friendly				
b	Are there any aspects of the application that have room for improvement?	Ability to respond based using question including, "Thursday" or "about 2 days". App got confused when I it knew today's date.				
c	What other weather-related features would be useful or interesting?	Lead question SB: Farhenheit or Celcius?				

Step 5: Reflection

General Impressions

I found aspects of this assignment harder, and some aspects that were easier than the previous Dialogflow assignment. It was easier because the use case was more straightforward, and there was a fairly large variety of intents that I could come up with. This is the second Dialogflow agent that I created, and not the first time I have used external APIs. However, it was also harder because the intents that I did end up creating were somewhat complex, so there was simply more logic to figure out and more to code compared to previous assignments.

There were aspects of the project that turned out well, and some that failed to live up to expectations or desired usefulness.

What turned out well was the usage of the APIs and the usage of user-specified queries. When I used APIs in a previous course, getting them to work successfully on a website was difficult. However, when using Node.js and Axios, the APIs worked flawlessly and required no additional back-end configuration. As for working with Javascript Date objects, I was able to convert everything to UTC, then at the very end, convert to the local time of the location that was queried. Using Date objects worked smoothly, likely because I had a lot of practice working with them in the previous Dialogflow assignment.

What also turned out well was getting as specific as possible when using a specified location from the user (assuming that Dialogflow interprets the location correctly). I set up the API calls to gather as much geolocation information as possible (missing states and/or countries if the user said only a city, for example). Then, in the final response to the user with the weather information, I would include that full geolocation data (city, state and/or country) as a way to confirm to the user where the weather is being checked.

This is possible because the OpenWeather API has a geocoding API endpoint that returns location information based on a query. If the API user queries “London” without any country or state, then the geocoding API response will contain information with all locations with the name “London”, and full naming and location data about all of them. To fully utilize this feature, I would call this API endpoint, select the location that matches the best based on the user input, then get the full information of that location. The output statement to the user will have a clear location name, and be as specific as the user makes it. I am fairly proud about creating this feature.

However, this is sometimes overshadowed by Dialogflow’s inability to consistently interpret the user queries. Dialogflow’s built-in location entity is an object that has fields for city, state, country, ZIP code, and other specific location types. An example would be if a Dialogflow user makes a query to get the weather for “London”; the location’s city field will be filled in, and everything else would be blank. But if the user says “London, Ohio”, the city and state fields would (theoretically be) filled in with the correct information, and my fulfillment code would use that location data to get the weather information for London, Ohio.

But in actuality, Dialogflow sometimes fails to provide correct location data to my fulfillment code, despite the state and/or country clearly being understood by the speech-to-text functionality. An example of this failure would be when a user tries to get weather information for some lesser known

London like “London, Ohio”, “London, Kentucky”, or “London, Ontario”; all of these are real locations that are all able to be queried with the OpenWeather API. Dialogflow would sometimes provide my fulfillment code with a location object with only “London” filled in and no state or country. My fulfillment code would then query for “London” and return London, UK as that is the top result. My fulfillment code would then return weather information for the wrong London, thus providing incorrect weather information to the user, all because the full location was not understood by Dialogflow. Because of this, it is disheartening to know that I created a way to get a location as specific as possible, but that specific information is simply not passed to my fulfillment code. This failing on Dialogflow’s part was also encountered during user testing.

If I were to do this project again, I would look into creating my own location entity, or completely revamping the sample phrases, as this may have played a part in the poor location recognition.

Another aspect of the project that didn’t turn out as well as I had hoped was when using the microphone. While this is largely outside of my ability to solve, I did feel like it had an impact on usability. About half of the time, when speaking any query, Dialogflow would automatically stop voice capture and start processing, even if there was only a half-second pause in speech from the user. It is because of this that I mainly stuck with typing my queries.

Completion Time

I estimate that it took about 50 hours to complete this project, from conceptual design to final demo. Creating the fulfillment was certainly the part that took the longest. Within the fulfillment, the forecast intent was likely the intent that took the longest. This involved getting the data, and trying to figure out what to do with it. Initially, I had a vague idea of “I will just show a forecast for a location”, but I was unsure how to present that data. I eventually came up with what type of information to report to the user, but then had to find a way to calculate a summary of the data. This intent had the longest code, as well as the longest design time.

The intent that took the smallest amount of time was the weather condition limits intent. This was the last intent that I completed, and a primary part of the code was actually already done within the forecast code for getting the highs and lows and the highest chance of precipitation. In order to adapt that code to the “Get Weather Limits” intent, it was just a matter of changing which weather condition was being evaluated for a minimum or maximum.

Something that I worked on for a surprising amount of time was testing the completed code by making random queries with international locations and different combinations of dates and weather conditions. I spent the entirety of one afternoon randomly looking around the globe for locations to find bugs and test the API accuracy. Through doing this I found that the API is not 100% accurate compared to other sources of weather data. For example with temperatures, it typically falls within 5 degrees fahrenheit of other weather data, and rarely does it match exactly the temperature of other weather sources.

Testing: During and After Development

For testing during development, I learned from my stresses of the previous Dialogflow assignment. Previously, I did all development in the in-browser fulfillment code editor, and found that deployment of that code took a long time, so testing after every tweak was annoying. This time, I created the intent functions in the in-browser code editor, but each of them had only a single function call to an external function. For those external functions, I did all development in an IDE with Node.js. When I was done with those functions in the IDE, I would simply copy and paste them into the in-browser fulfillment window and then deploy.

Because the Dialogflow testing window showed the parameter entity objects that are obtained after each user query, this allowed for a way to quickly test. Within the IDE, I set up arrays of sample parameter entity objects (such as location objects, dates, or weather conditions). After I had completed a major feature, I would run a nested loop and try every combination of those sample parameters. Using this method, I was able to essentially perform dozens of user queries in a few seconds, and I was able to find and fix several bugs in a very short period of time. If I had done all testing exclusively in the browser one query at a time, development would have taken far, far longer.

Getting weather conditions from different locations around the world was another factor in finding a number of bugs. If some weather condition was not applicable for an area, like if it was not raining, then the weather API response would simply not include the rain object field. This made undefined-variable-related errors very common. When testing a variety of locations, that allowed for easy catching of these errors.

For testing the application with other users, I created a page on my personal website that has an embedded Dialogflow window, as well as the 5 sample tasks and a link to a Google Form for evaluation.

Overall, the results and feedback of the user tests provided validation to how I felt with the application. Also, valid suggestions were made that I will address in the future works section.

Both the first and second user thought that the application was easy to use. However, opinions on what could be improved on were different; the first user thought that the recognition of locations could use more work, and the second user thought that better interpretation of date and times. The first user tried to use statements like “What is the hottest temperature in the world” and “what is the temperature in Saudi Arabia”. Both of those statements are not possible with the API that was used, so there is not a lot that can be done about it. The second user tried many times to get a 1-day or 2-day forecast for a specific location, but failed most times to have the day duration recognized. The second user also tried queries like “What is Tuesday’s forecast…”

Any negative feedback that was received was only confirmation of what I already knew about the application. I sometimes had trouble getting the speech recognition to get the forecast of a duration other than what the maximum of what the API returned, despite knowing the exact syntax that is needed for that query. I also acknowledge that location recognition needs more work.

There was also feedback about fahrenheit versus celsius. It was suggested that an option to get the information in celsius should be added. I had pondered in the early stages of development adding a way to get information in metric or imperial units, but thought it would not be needed. Realistically, this could be added to the end of any sample utterance like “What is the temperature in London in celsius” or “What is the rainfall amount in London in metric units”.

It is also worth noting that I had text-to-speech enabled for the application. For one user, when the forecast was being read back, they asked how to cancel the playback, as the speech kept on going. In their survey, they did agree that the forecast response was a bit too long. The length of the forecast response is something that is hard to optimize.

If I had more time for testing, I would obviously ask more people. This would give a wider variety of ways to phrase queries, get more diverse locations, and get more opinions and feature suggestions.

Future Work

After development and user input, I have identified several aspects that could either be improved on, or overhauled and several features that would make good additions to the application.

One part of my application that I feel needs improvement is the general forecast response that is given to the user. Trying to find the sweet spot between response length and response usefulness is difficult, particularly since it is automatic based on an array of weather instances. If a human had to interpret a forecast, the wording, length and specificity would be highly tailored exactly to the forecast. When a simple algorithm outputs a forecast in sentence format, it might be too verbose or not have enough detail, or be too repetitive. One user thought that the forecast length was fine, the other thought it was too long. Based on my own observation and some user input, the forecast statement would require a lot more research and design time to perfect. Given more time, improvement of the forecast statement would likely be a primary focus.

A possible addition to the application would be to allow forecasts for a specific date. A user had tried to do this during testing and I must admit, this option did not occur to me during design or development. Allowing for a forecast for a specific day would be a very nice addition, and would likely be a primary focus next to the reworking of the forecast's return statement.

The same user also mentioned the use of celsius, as noted in the user testing section. It would be relatively easy to add an optional few words at the end of the sample phrases to allow the user to choose which unit they want their data returned in. Allowing different units would be a nice quality-of-life addition.

Other possible addons to this project would be other weather-adjacent conditions, such as pollen count or air pollution. OpenWeather has an air pollution API, so this one could easily be added as a weather condition option.

If future testing were to be done for this project, I would look into how weather queries are spoken. I feel like I covered all possible ways to say the intents of this project, but I would be curious to see if there are more.

I would also look into if a custom location entity is required. Of all of the entities used in this project, the location is the one that fails to be interpreted correctly the most. At the same time, it is also the most versatile, since it can distinguish between cities, states and countries, so it would likely be difficult to develop a new one that works better.

Step 6: Demo and Presentation

Demo Link:
- https://maxoakes.github.io/dialogflow.html

Presentation:
- https://media.pdx.edu/media/t/1_d4ye0e9j