Deliverable for UltraWeather
for UTM CSCI 352

Andrew Newbill, Joshua Chamberlain, Aaron Alden

Abstract

This computer application uses a weather API to collect certain weather data such as temperature, humidity, etc. to predict the weather forecast throughout the day. The target audience are driving adults who need to know the weather conditions to get to work on time. The application will also have emergency notifications, such as tornado warnings. The progress on the project is that we have selected a specific API to collect weather data.

1. Introduction

We planned this project as a way to practice our C# and design pattern skills. Our hope is that in doing so we can deliver a product that is user friendly and well designed so that users will have an easy way to check the weather from their desktop. One advantage of this is that web browsers are often resource hungry and a small WPF app like ours will not be. This is crucial for low end systems where multitasking with a web-browser open slows everything to a crawl. We are really targeting power-users who would like to use as little resources as possible to track the weather for as many areas as they choose. It is often the case with weather apps that they quickly become cumbersome when used for more than a few cities/regions.

Another issue we plan to tackle with this project is privacy. Weather applications by companies such as Apple almost certainly store your information on a server. Since most people keep not only the city they live in but the ones that they travel to pinned in these apps, Apple knows exactly where you live and where you go based off only in the information in this application. By storing this information client-side(and we plan to encrypt it) we give the user more security in how their data is stored. Closed ecosystems such as Apple's intentionally do not reveal exactly how they handle user information and this is not acceptable to everyone. We intend to be entirely transparent about how we handle your data since the project will be open-sourced once it is delivered. Even if our implementation ends up being imperfect or less secure than we hoped, at least we won't be hiding it.

1.1. Background

1.1.1. Terms to Know. The reader should be familiar with the general terms used to refer to various weather patterns. We plan to use a tiling system for this project. A tile in the context of our program will be a drag-able object that contains information on the weather for specific city or town. You will be able to organize these tiles with folders, which are simply containers for tiles.

1.1.2. Personal Connection. We came up with the idea for this project because most weather apps are integrated into operating systems and as such they have limited functionality for users who may travel between large areas or are simply interested in following the weather on a large scale. One strong inspiration was the recently unpredictable weather in Tennessee. It would be good for users working on the desktop to be able to check the weather as they work with a simple and fast app that does not run as a system service.

1.2. Impacts

We hope that our project we impact the safety of its users in a positive way. A user could be working at their computer, blissfully unaware of a coming storm. We hope that our user would happen to check the app and decide not to leave their home on that day. This common situation could also impact the safety of the general public since more people on the road during a dangerous storm leads to a higher chance of accidents. It could happen that a social get-together that was planned gets canceled once one user sees coming poor weather and notifies their friends. We also hope that working on this project impacts us positively by furthering our experience at proposing projects and working as a group.

1.3. Challenges

The first major challenge will be getting a C# WPF application to interact with the OpenWeather API. Once we can call the api and receive data the next major challenge will be organizing this data into something that is easy for the user

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Set location	Application user	Med	1
2	Add Favorites	Application user	Low	2
3	Search Location's weather	Application user	Med	3

TABLE 1. ULTRAWEATHER 1ST DRAFT USE CASE TABLE

to view and interpret. After that we will have to overcome the task of implementing 'tiles' that allow the user to organize more than one location's data. We plan to overcome the challenges by first reading the XML data into variables that can be married to fields in the tiles. Getting the data organized into something that is pleasant to view and organize will be by far the biggest challenge of this project. The free tier of OpenWeather does have certain limitations

2. Scope

We understand that in 2022, most weather apps have the same features we come to expect. The main way we can make our app different is by focusing more on the aesthetics and graphics of the app. We want to take a minimalist, tile approach to our app. We also want to implement our radar. Whether we will have it dynamic (a looping map) or static is still to be decided. For our stretch goals, we want to implement severe weather alerts and also "pre-alerts". For the pre-alerts, the idea is to give users a heads-up alert up to a week before severe weather occurs. For example, if meteorologists are predicting a nasty storm front next week, the user should be alerted that there is the possibility for severe weather. Sometimes, just a tornado warning pushed to your phone as the storm is happening is too late to seek shelter.

2.1. Requirements

The requirements were acquired with all of us talking in class about the function and non-functional requirements as well as further communication through discord about both types of requirements.

2.1.1. Functional.

- The User should have the ability to pick what location they want to get the weather from – allows the User to get particular information for their use.
- The User will be able to see all weather information provided by OpenWeatherAPI.
- The User will be alerted when the national weather service has alerts for their location and their favorited locations.
- The system will connect to the OpenWeatherAPI.
- The User will be able to put locations into folders.
- The system will be able to store the locations previously searched by the user.
- The User will be able to search for a location and/or zip code.

2.1.2. Non-Functional.

- Privacy - The user's current location will be protected and not given to any third parties. The user will have an option to not allow the app to track their location.
- Reliability - The app will work with a proper internet connection.
- Capacity - The app will have the same capacity as the storage on the computer i.e. The user will be able to have as many locations as their phone or computer can fit.

2.2. Use Cases

Use Case Number:	1
Use Case Name:	Set location
Description:	Upon opening the application, the user will have the option to set their current location, i.e. "Use current location". The user will then be presented with the weather forecast for the provided location.
1)	If the user wishes the application to find their location, they will click on the alert that the application is trying to access their location.
2)	If the user wishes to set their home location manually, they can enter their zip code into a "Set your location" menu option.
Termination Outcome:	The user now has access to their "home" location's weather. Making it readily available whenever they access the application.

Use Case Number:	2
Use Case Name:	Add favorites
Description:	The user can store favorite places to readily see what the weather is for these saved locations. They will be collected all in one place.
1)	The user will navigate to “Add favorites” section.
2)	The user will type in the zip code for desired location.
3)	The user will have the option to save as a favorite to be readily available to view.
Termination Outcome:	A collection of saved favorite locations will be stored in the application.

Use Case Number:	3
Use Case Name:	Search Location’s weather
Description:	The user will have the option to search for a location’s weather by location (zip code). The application will then show them the current weather forecast for the searched location.
1)	The user will type in the desired zip code in the “Search” bar.
2)	The application will present the current weather forecast.
3)	The user can search for new locations in the search bar if desired.
Termination Outcome:	The user can now interact with all features of the application regarding this location.

2.3. Interface Mockups

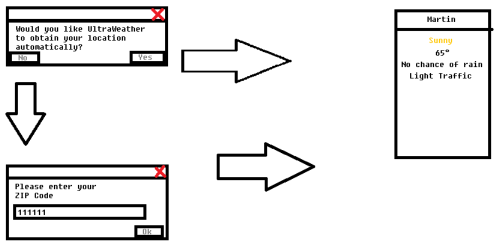

Would you like UltraWeather to obtain your location automatically?
No
Yes
Please enter your ZIP Code
111111
Ok
Martin
Sunny
65°
No chance of rain
Light Traffic

Figure 1. Illustration of Use Case 1. Here the user is choosing whether or not to automatically obtain their location.

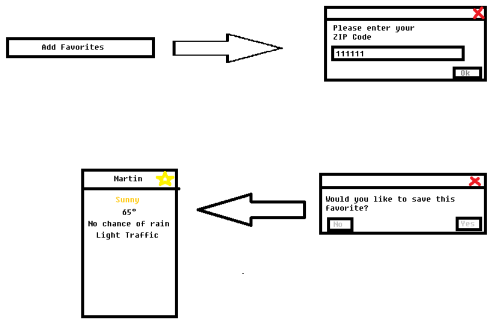


Figure 2. Illustration of Use Case 2. Here the user is adding a favorite to their favorites folder.

3. Project Timeline

Milestones: Feburary 5th: The requirements stage has been discussed and decided. March 1st: All designs are finished, shown through our use cases. March 29 - 31: Implementation starts with Andrew coding the system that connects to the API. April 5 - 12: Implementing the user interface designs shown through the use cases and implementing the location search function. April 14 - 21: Implementing other functions such as the tiling system and GPS optimal route searching functions If time is available: April 19 - 26: Work on radar stretch goal. April 28: Verfication and testing of all different

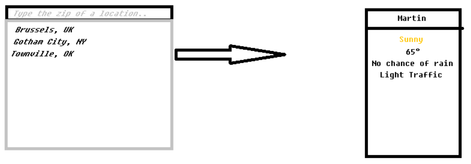
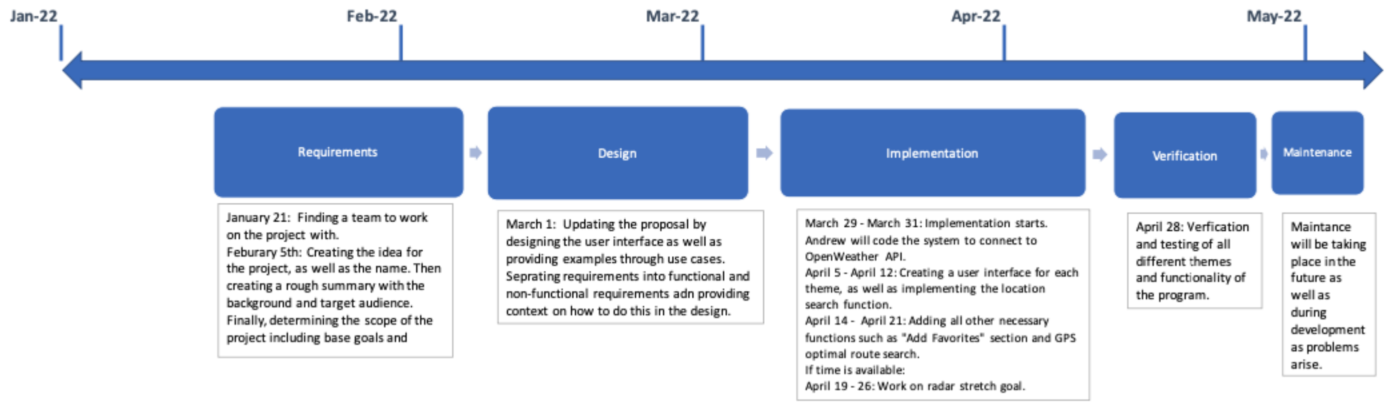


Figure 3. Illustration of Use Case 3. Here the user is searching for a location.



January 21: Finding a team to work on the project with.
Feburary 5th: Creating the idea for the project, as well as the name. Then creating a rough summary with the background and target audience.
Finally, determining the scope of the project including base goals and

March 1: Updating the proposal by designing the user interface as well as providing examples through use cases.
Seprating requirements into functional and non-functional requirements adn providing context on how to do this in the design.

March 29 - March 31: Implementation starts.
Andrew will code the system to connect to OpenWeather API.
April 5 - April 12: Creating a user interface for each theme, as well as implementing the location search function.
April 14 - April 21: Adding all other necessary functions such as "Add Favorites" section and GPS optimal route search.
If time is available:
April 19 - 26: Work on radar stretch goal.

April 28: Verfication and testing of all different themes and functionality of the program.

Maintance will be taking place in the future as well as during development as problems arise.

Figure 4. Full Project Timeline

themes and functionality of the program. Maintance will be taking place in the future as well as during development as problems arise.

4. Project Structure

At first, this will be a little empty (it will need to be filled in by the time you turn in your final report). This is your chance to discuss all of your design decisions (consider this the README's big brother).

4.1. UML Outline

Show the full structure of your program. Make sure to keep on updating this section as your project evolves (you often start out with one plan, but end up modifying things as you move along). As a note, while Dia fails miserably at generating pdfs (probably my fault), I have had much success with png files. Make sure to wrap your images in a `figure` environment, and to reference with the `ref` command. For example, see Figure 4.

4.2. Design Patterns Used

Make sure to actually use at least 2 design patterns from this class. This is not normally part of such documentation, but largely just specific to this class – I want to see you use the patterns!

5. Results

This section will start out a little vague, but it should grow as your project evolves. With each deliverable you hand in, give me a final summary of where your project stands. By the end, this should be a reflective section discussing how many of your original goals you managed to attain/how many desired use cases you implemented/how many extra features you added.

Figure 5. Your figures should be in the *figure* environment, and have captions. Should also be of diagrams pertaining to your project, not random internet kittens

5.1. Future Work

Where are you going next with your project? For early deliverables, what are your next steps? (HINT: you will typically want to look back at your timeline and evaluate: did you meet your expected goals? Are you ahead of schedule? Did you decide to shift gears and implement a new feature?) By the end, what do you plan on doing with this project? Will you try to sell it? Set it on fire? Link to it on your resume and forget it exists?

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.