UltraWeather

Andrew Newbill, Joshua Chamberlain

Abstract

This application uses the OpenWeatherAPI to collect weather data for the current time as well as the forecast for the week for a specific city in a state and displays to users in an easy to read manner. The target audience is students who need to know the weather conditions to get to work or class on time. UltraWeather is able to call the API and get the current weather and forecasting weather for up to a week. It also has switchable themes that can be changed at any time.

1. Introduction

Our goal is to deliver a product that is user friendly and well designed so that users 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 isn't. This is crucial for low end systems where multitasking with a web-browser open slows everything to a crawl. We are targe power-users who would like to use as few resources as possible to track the weather. It is often the case with weather apps that they use far more resources than necessary.

Another issue we tackle with this project is privacy. Weather applications by companies 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, companies know exactly where you live and where you go based off only the information in this application. We store information client-side to give the user more security in how their data is stored. Closed ecosystems 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 is open-source. Even if our implementation is imperfect or less secure, at least aren't 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 theme system for this project. A theme in the context of our program will be a change of icons that represent the current weather for a given day. The user should also be familiar with the imperial system, as our program will provide the requested data in this form.

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 was getting our C# WPF application to interact with the OpenWeather API. Once we could the API and receive data the next major challenge was organizing this data into something that is easy for the user to view and interpret. After that we had to overcome the task of implementing ”themes” that allow the user to organize more than one location's data. We overcame these challenges by first reading the XML data from the API into variables that can be placed at various locations in the window. Getting the data organized into something that is pleasant to view and organize was by far the biggest challenge of this project. The free tier of OpenWeather limits us to a certain amount of calls, and does not allow us access to data beyond current weather and forecast.

Use Case ID	Use Case Name	Primary Actor	Complexity	Priority
1	Search for weather information for a location	Application user	Med	1
2	Entering a location that is not found	Application user	Low	2
3	Changing the theme	Application user	Med	3

TABLE 1. ULTRAWEATHER USE CASE TABLE

2. Scope

We understand that in 2022, most weather apps have the same features such as temperature and humidity. The main way we can make our app different is by focusing more on the aesthetics and graphics of the app. We plan to do this by implementing themes that allow the user to change the look and feel of their application at any time. We also want to implement a radar system that can help users identify weather systems that may be approaching as well as implementing severe weather alerts as a stretch goals.

2.1. Requirements

We decided on the requirements by talking in class about functional 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.
- All weather information will be displayed to the user in the Window.
- The User will be able to switch between the two modes: Cartoon and Emoji.
- The User will be able to search for a location.
- The User will be prompted with a message box if the city entered does not exist.

2.1.2. Non-Functional.

- Privacy - The user's search locations will be protected and not be given to any third parties besides OpenWeather API.
- Reliability - UltraWeather will work 100
- Capacity - UltraWeather will contain the capacity will be able to contain information about the current weather and a week after. Also the API calls are limited to 1,000,000 calls per month.
- The system will connect to the OpenWeatherAPI.

2.2. Use Cases

Use Case Number: 1
Use Case Name: Searching for current weather and forecast for a location
Description: Upon opening the application, the user will have the option to enter a city and state to get the weather for that location.

1) If the user wishes the application to find the weather for a location they can first enter the name of the city.
2) After entering the name of the city, the user can select which state the city is located in.
3) Once they have entered both the city and the state, the user can press the Go button to retrieve the information.

Termination Outcome: The application will display the current weather as well as the forecast for the week for that location.

Use Case Number: 2
Use Case Name: Entering a location that is not found
Description: A message box appears that lets the user know that the city was not found. No information will change in the window.

1) The user enters the name of the city for which they wish to get weather information.
2) The user selects the state of the city they wish to gather weather information for.

Termination Outcome: A message box appears that let's the user know the city was not found. None of the weather data displayed in the window changes.

Use Case Number: 3

Use Case Name: Changing the theme

Description: The user will have the option to change the theme of the application. This will not change the functionality.

1) The user wishes to change their theme. They select a theme from the list of available themes.

Termination Outcome: The icons change for the application to reflect the theme that the user has selected.

2.3. Interface Mockups

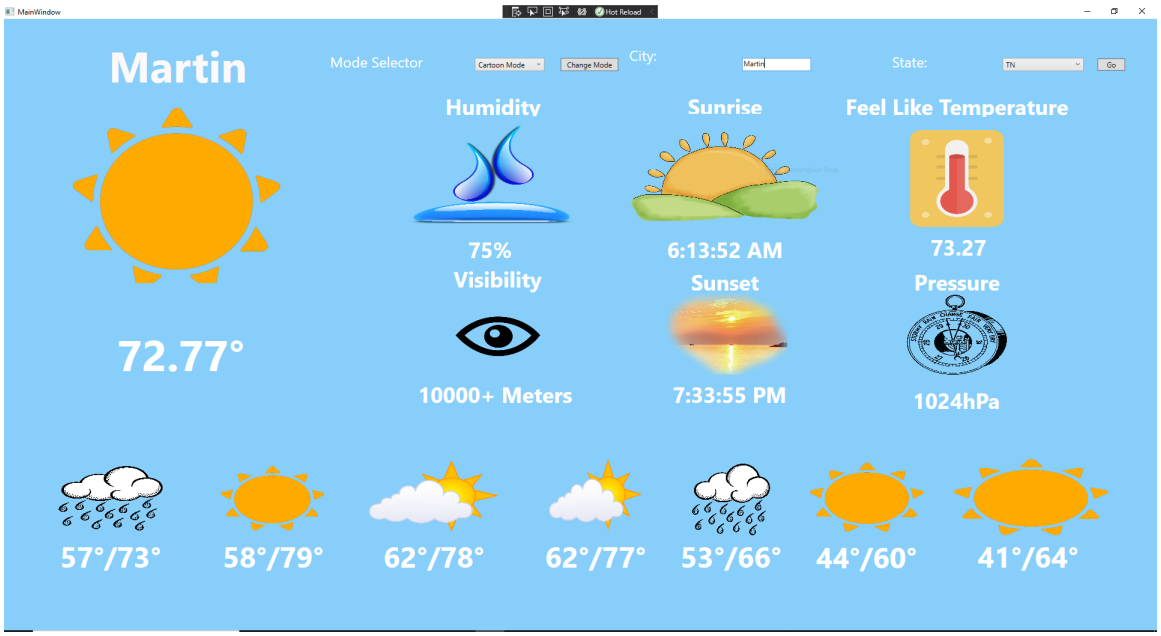

Figure 1. Illustration of Use Case 1. Here the user has entered a city and state and retrieved the weather information for said location.

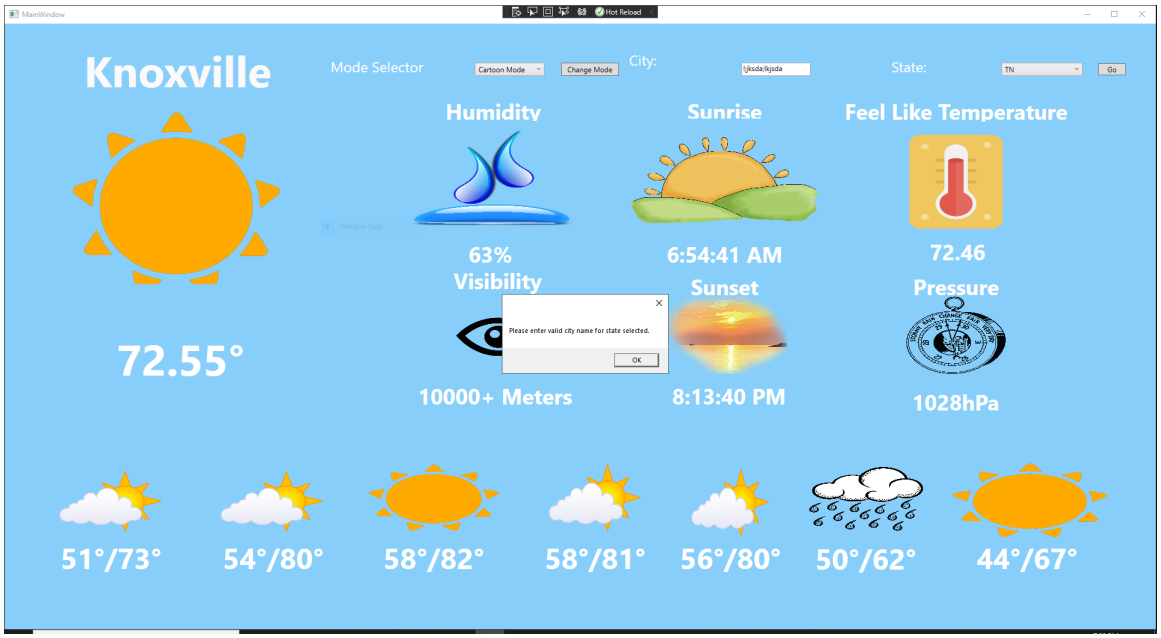

Figure 2. Illustration of Use Case 2. Here the has entered a city that could not be found in their chosen state.

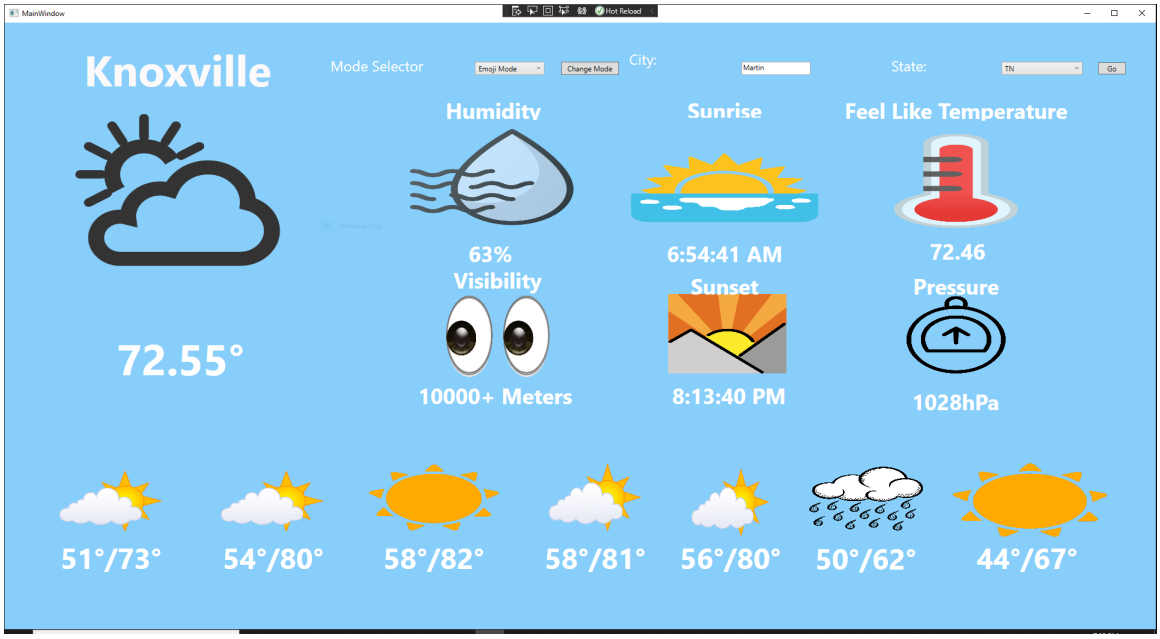

Figure 3. Illustration of Use Case 3. Here the user has changed the theme.

3. Project Structure

The program is structured around the API Container. The Go button acts as a facade for the complex API call that happens once the user enters their city and selects their state. The program uses part of the visitor pattern for themes. All of the data is contained in a temporary City struct that will be destroyed once the data has been given to the relevant fields. This ensures a minimum usage of resources. The MainWindow class contains the functions for every button and list in our program window, which it uses mainly to access the API Container's call api functions.

3.1. UML Outline

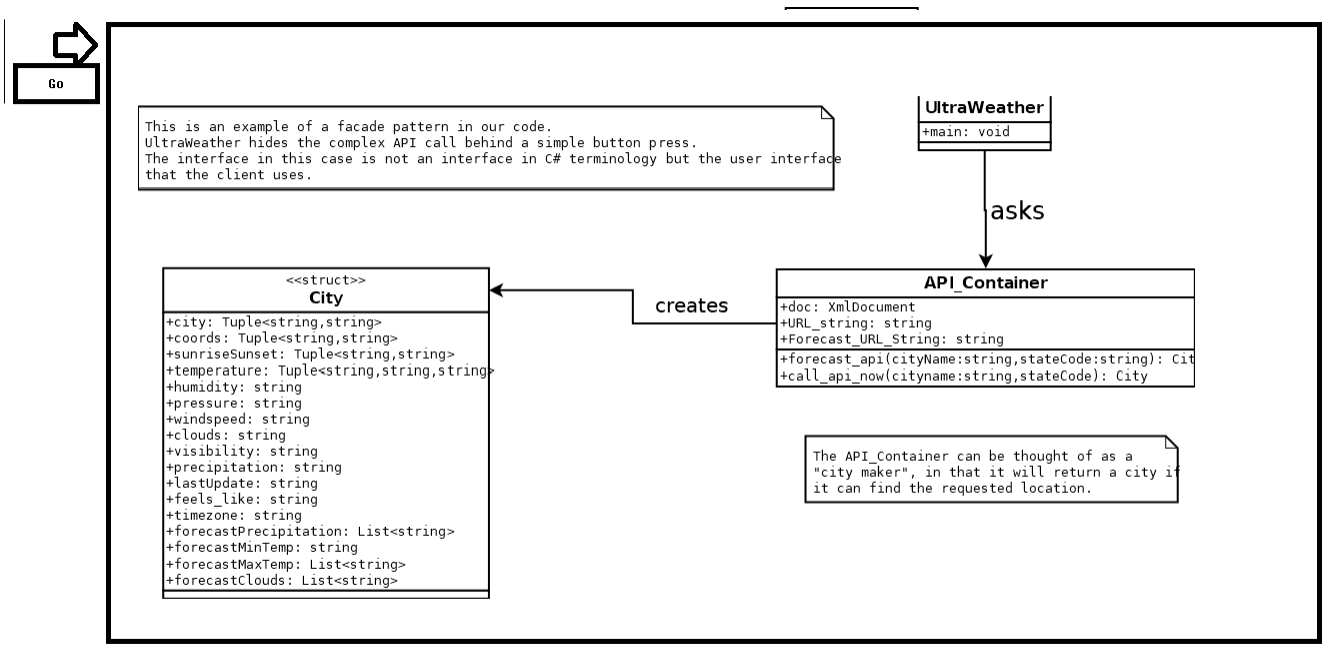

Figure 4. This is the facade pattern used in UltraWeather. The button is the facade for the complex API call that happens behind the scenes when the user presses the go button.

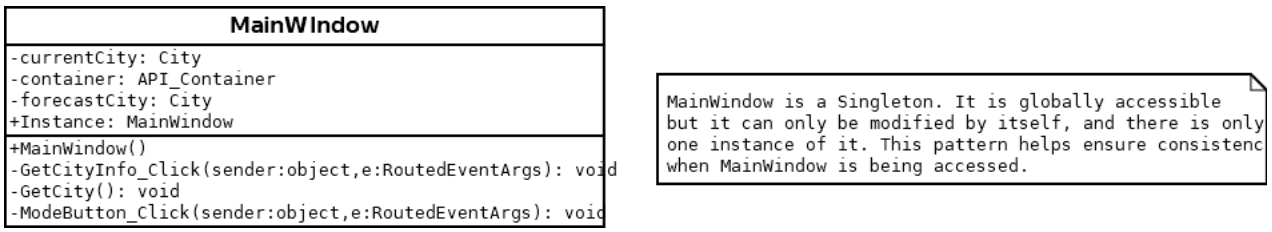

Figure 5. This is the singleton pattern used in UltraWeather. The main program window is singleton, which means that it is globally accessible and there exists only one window.

4. 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.

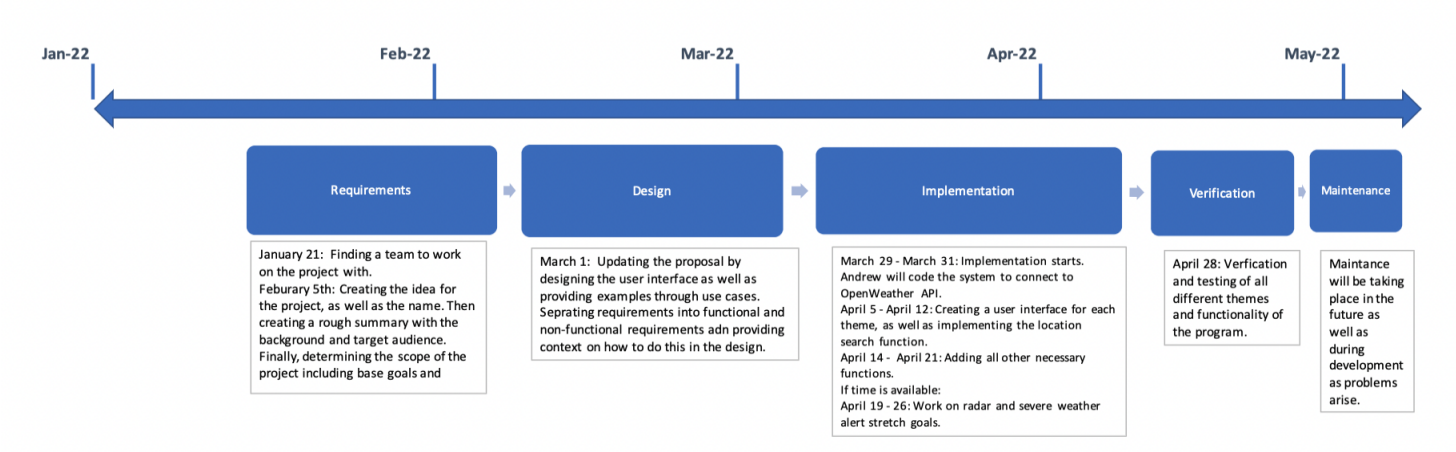

Figure 6. Full Project Timeline

- 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 a ForecastPicChangerand functionality for switching modes.
- 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.
- Maintenance will be taking place in the future as well as during development as problems arise.

4.1. Design Patterns Used

We used a facade pattern and a singleton pattern. The facade pattern comes with the user interface. The user interface is set up in a way that the user can provide information to the system, but they cannot touch anything on the backend. The singleton design pattern was used to make the MainWindow only be initialized once.

5. Results

The layout of UltraWeather is very information intensive as intended. It displays multiple weather attributes, such as humidity, visibility, and the 7 day forecast in two different themes. Unfortunately, the stretch goals of radar and the severe weather alerts were not implemented due to time constraints.

5.1. Future Work

We seek to implement the stretch goals such as implementing a radar and improve the efficiency of the program even further as our knowledge of computer science grows. The prioritization of the stretch goals goes implementing the radar, adding severe weather alerts, and adding new themes. Afterwards, we plan to put it on our resumes and forget about the specific details, but keep the knowledge we gained from the process of finishing the project.