UltraWeather

Andrew Newbill, Joshua Chamberlain, Aaron Alden

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. The target audience are driving adults and 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 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 targeting 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 plan to 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 plan to 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 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 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 will be getting our 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 to view and interpret. After that we will have to overcome the task of implementing "themes" that allow the user to organize more than one location's data. We plan to overcome the 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 will be 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	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

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 want to take an information intensive approach giving the user all necessary information to plan out their day based on the weather. For our stretch goals, we want to implement severe weather alerts and implement a radar. The radar would be static and depict the radar at the current time only.

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.
- 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 by deleting the search locations after it is finished.
- 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: 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 City into the textbox and select a state from the combobox. Then they will press a button and the data will appear for that city.
2) The application will present the current weather forecast.
3) The user can search for new locations if desired.

Termination Outcome: The user can now interact with all features of the application regarding this location.

2.3. Interface Mockups

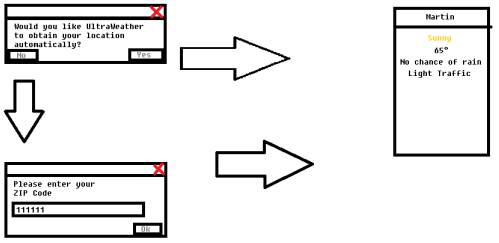


Figure 1. Illustration of Use Case 1. Here the user is choosing whether or not to automatically obtain their location.

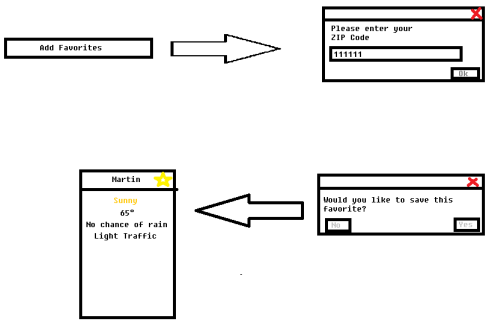


Figure 2. Illustration of Use Case 2. Here the user is adding a favorite to their favorites folder.

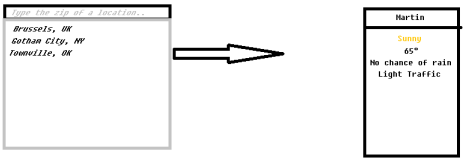


Figure 3. Illustration of Use Case 3. Here the user is searching for a location.

3. Project Timeline

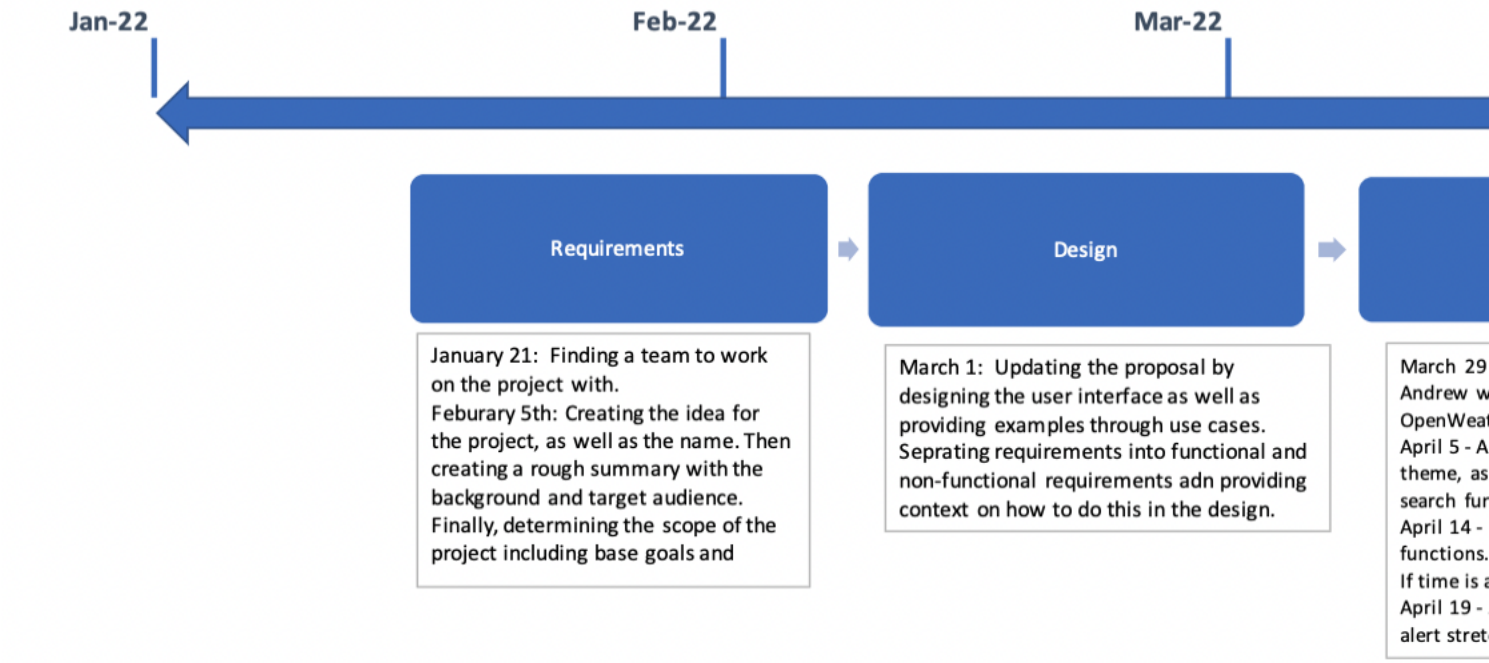


Figure 4. Full Project Timeline

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

3.1. UML Outline

4. Project Structure

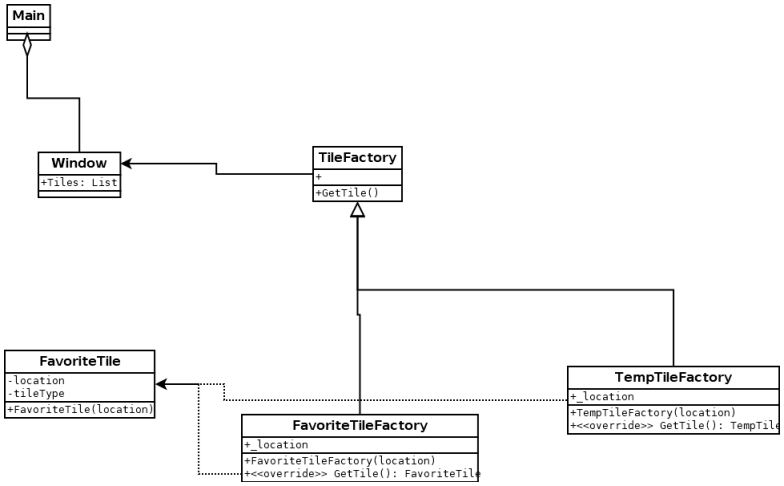
We haven't decided on all of design patterns yet but we feel that the factory method would be good for the tile system we plan to implement. Each factory will be responsible for producing different types of tile, depending on what the users wants. We also need to come up with some more tile types to make the project more useful and compelling to use.

4.1. Design Patterns Used

We used a facade pattern and a vistor 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 visitor pattern is used when checking the mode/theme of the program. The program visits each data point (the two themes in this case) and does different actions based on which one it is.

5. Results

The UltraWeather collects data from OpenWeather API and displays it to the screen in a user friendly way. UltraWeather allows the person to type in a city and checks whether the city is valid. It provides an error MessageBox when it is not valid. The pictures will change when the weather is different, and the forecast will be as accurate as the API.



5.1. Future Work

We seek to implement the stretch goals and improve the efficiency of the program even further as our knowledge of computer science grows. Afterwards, we plan to put it on our resumes and forget about the specific details, but keep the knowledge we gained from teh process of finishing the project.

References

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