Weather Data Integration Pipeline and Dashboard Technical Report

Rohit Sunilkumar Hooda, Rimsha Kayastha, Tri Watanasuparp

November 25, 2024

1 Introduction

Many people start their day by checking the weather forecast to plan activities such as commuting, exercising, or scheduling events. Given how essential accurate weather information is to daily life, it's important to have reliable and comprehensive tools at our disposal. This project seeks to fill that need by integrating data from multiple sources and offering a detailed weather dashboard that provides both current forecasts, for future preparations, and historical data, to be aware of the ongoing weather trends. Additionally, it incorporates interactive, user-friendly visualizations that empower users to analyze trends, draw relevant conclusions, and make informed decisions based on weather patterns. This combination of real-time updates and historical context aims to enhance users' ability to anticipate weather conditions and better plan their activities. This project focuses on Massachusetts, a US state, in particular for work on a more detailed, smaller scale.

2 Literature Review

2.1 Arcgis Dashboard

The Arcgis weather dashboard is a powerful tool designed to display real-time weather data with an emphasis on USA weather alerts, including storms, heat waves, and hurricane watches. Central to the dashboard is an interactive map, as you can see in Figure 1, that users can easily navigate—dragging, zooming in, or zooming out—to access weather information specific to any location. This interactivity provides an accessible way for users to obtain localized data, making it especially valuable for monitoring conditions and staying informed about potential hazards in targeted areas.

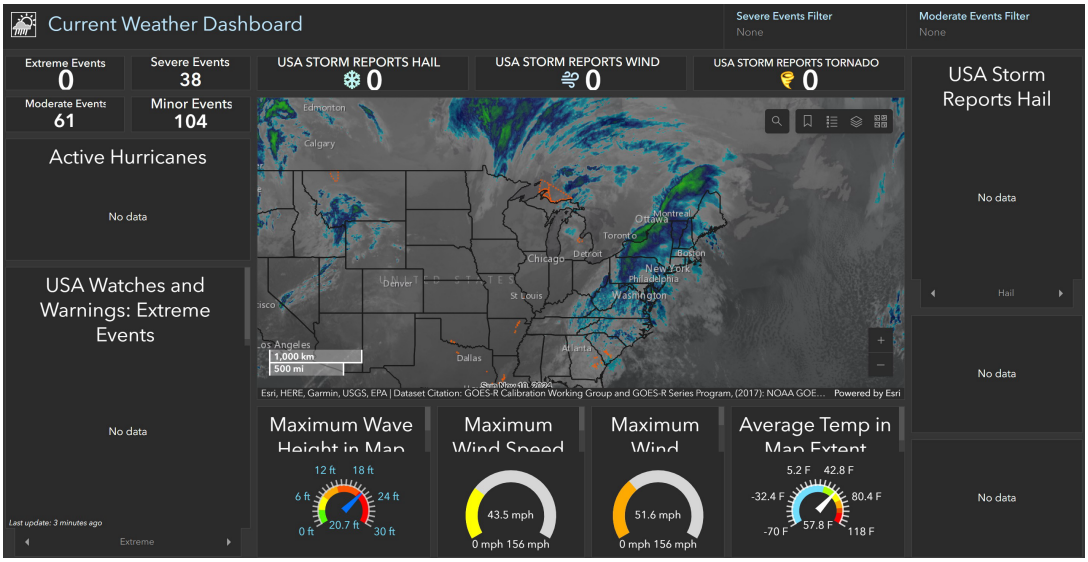

Figure 1: Arcgis Weather Dashboard

2.2 Ambient Weather Dashboard

The Ambient Weather dashboard provides detailed, real-time weather data sourced from personal weather stations, offering hyper-localized information that is essential for users who need precise, neighborhood-specific updates. As shown in Figure 2, it tracks key weather metrics such as temperature, humidity, wind speed, and rainfall, ensuring a comprehensive view of current conditions. Furthermore, the dashboard is highly customizable, allowing users to choose the specific weather parameters they wish to monitor and arrange the data widgets to suit their needs, whether for gardening, event planning, or professional purposes. Additionally, it includes historical data analysis capabilities, enabling users to observe weather trends over time. This tailored and interactive approach makes the ambient Weather dashboard a valuable resource for anyone wanting to stay informed about local weather conditions.

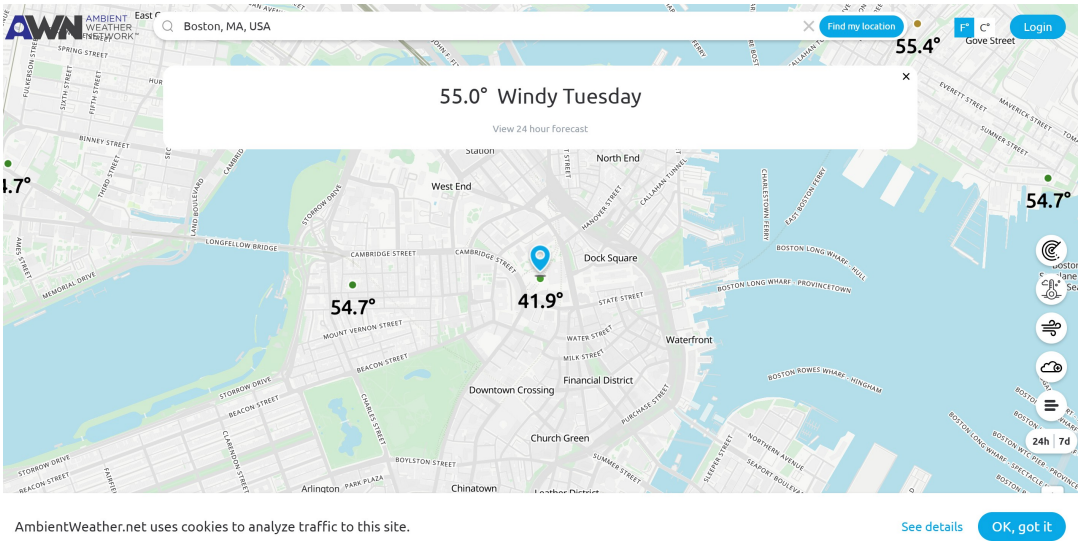

Figure 2: Ambient Weather Dashboard

3 Project Structure

As shown in Figure 3, the flow of this project has three sections: Data sources, transformation, and deliverables. The first section involves retrieving data from multiple sources and integrating them into a singular pipeline and feeding them into the next section of data processing and transformation. Data transformation refers to the cleaning of data in order to address null values and outliers, and pre-processing refers to standardizing the data into a format that makes it more readable for users. Finally, the final section of the project is to produce readable visualizations that helps draw conclusions and portray them in distributable formats.

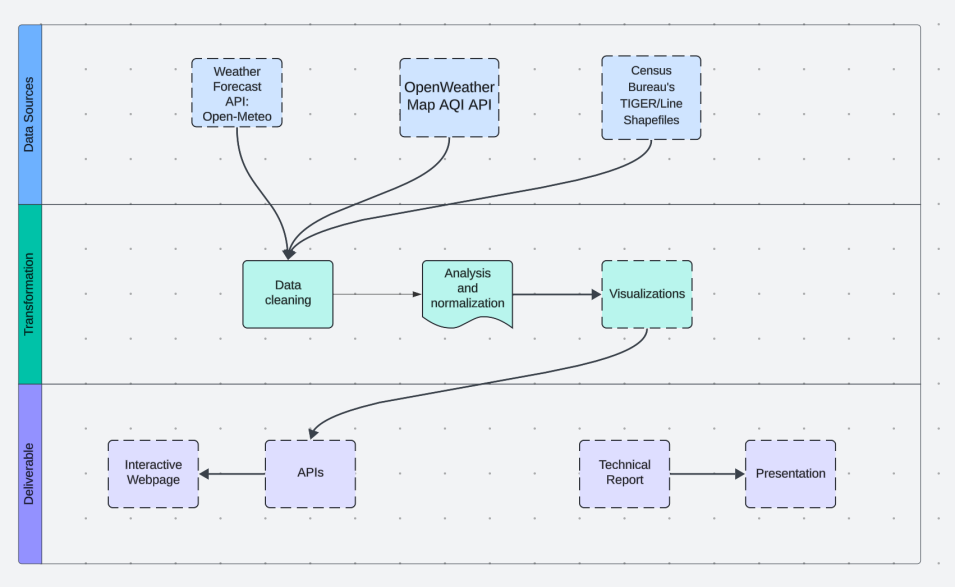

Figure 3: Project Process

4 Data Pre-processing

4.1 Data retrieval

Our weather data is retrieved from three sources through API calls, Mateo, AQI, and Census Bureau's TIGER/Line Shapefiles, and pipe-lined into a single data frame that is then cleaned, analyzed, and plotted for inference. The raw, combined data includes the following features: latitude, longitude, weather code, daylight duration, sunshine duration, max temperature, min temperature, max uv index reached, max precipitation probability, and AQI value. These features are then filtered down to the **12 features** we are focusing on for the dashboard: **date, county, latitude, longitude, weather code, daylight duration, sunshine duration, max temperature, min temperature, max uv index, max precipitation probability, and aqi value**.

4.2 Data Analysis

Gathering data from the three sources resulted in obtaining more than 25 features, from which they're funneled down into the more relevant 12 features mentioned in the previous section. Consequently, analysis was done on the spread of data for those features. For analysis, the features were divided into three different groups based on their data scale. The bar chart shown in Figure 4 for spread of data for sunshine and daylight duration where each unit is two hours.

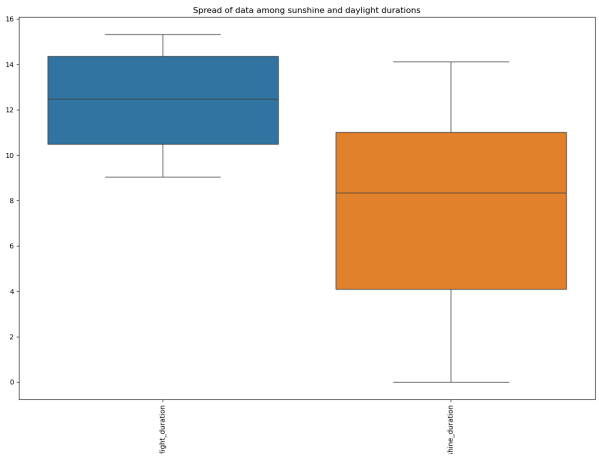

Figure 4: Sunshine-daylight duration data spread

Figure 5 shows the spread of minimum and maximum temperature data. Figure 6 shows the spreadof the remaining features.

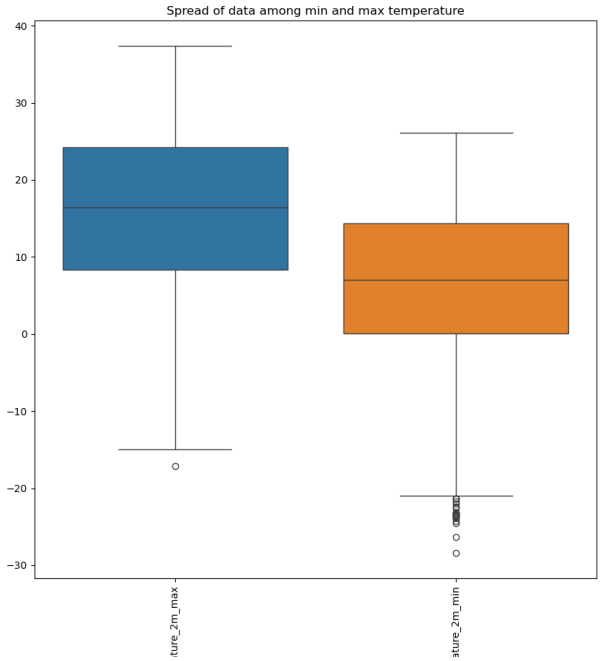

Figure 5: Min and max temperature data spread

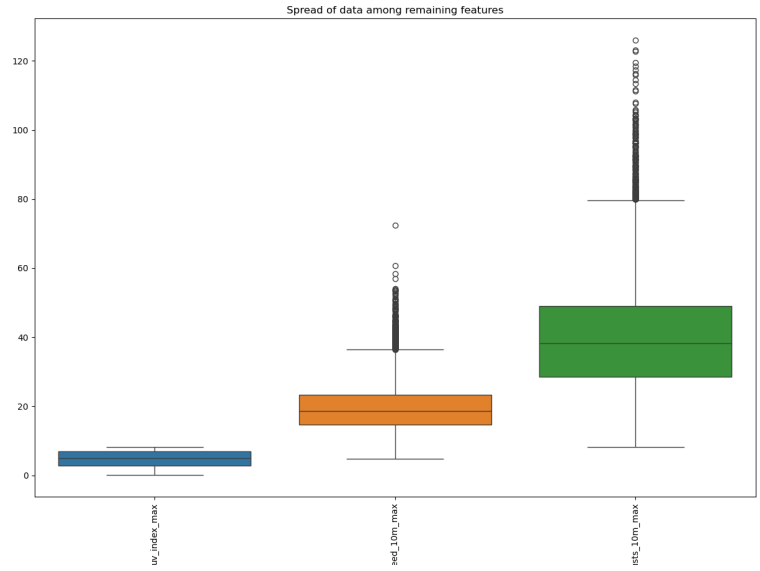

Figure 6: Data spread among the remaining features

4.3 Data Cleaning

The bar charts in the previous section show many outliers that need to be adjusted. Using the quartiles from there data spread, the data points are clipped to an upper and lower bound, such that lower bound is q1 - 1.5 * IQR and upper bound is q3 + 1.5 * IQR, where q1 is the first quartile, q3 is the third quartile, and the IQR is the interquartile range.

5 Webapp Architecture

5.1 Front End Architecture

The frontend is written in HTML, CSS, JavaScript and we are using **Axios** for API communication and **AnyChart** library for data visualization. The frontend will enable users to select weather metrics, date ranges, and visualize data interactively.

Visualizing the Weather Data on the Frontend

The **fetchWeatherData()** function serves as a bridge between the frontend and backend, handling user input, data processing, and visualization. Here's how the frontend workflow is structured:

1. **User Input Retrieval**
   - The function gathers inputs from the DOM, including the selected county, metrics (e.g., temperature, precipitation), and date range.
2. **Constructing API Request**
   - A GET request is dynamically built for the /weather endpoint, incorporating user-selected parameters.
3. **Processing the API Response**
   - The JSON response from the backend is parsed to extract:
     - info_type: Represents the name of the weather metric (e.g., temperature or rainfall).
     - values: An array containing date-value pairs, formatted for charting.
4. **Rendering with AnyChart**

- The extracted data is fed into the AnyChart library to generate a line chart. Key aspects include:
  - Each metric is visualized as a distinct series, represented with a unique color for clarity and easy differentiation.

This setup ensures a seamless user experience by combining real-time interactivity with visually engaging data representation.

5.2 Backend Architecture

The backend API for the Weather Visualization App is developed using Flask, ensuring a lightweight yet robust architecture for handling requests. For data extraction, we utilized multiple sources such as OpenMateo, OpenWeatherAPI, and TIGER Shapefiles, enabling a rich and comprehensive dataset.

To prepare the data for analysis and visualization, we employed Python libraries like Pandas for data cleaning and manipulation. Seaborn was leveraged for creating informative visualizations, and Folium was used to generate detailed maps of Massachusetts counties. This combination of tools ensures efficient data processing and visually appealing representations, creating a seamless backend experience.

Routes

1. **Index Route: (GET /)**
   - **Purpose**: Displays an interactive map of Massachusetts with county-level weather data.
   - **Implementation**:
     - Uses **Folium** to create a map.
     - Dynamically generates popups for each county with real-time weather data.
     - Fetches weather data using the county_weather_data dictionary.
     - Encodes county boundaries into GeoJSON to render shapes on the map.

2. **Weather Data Route (GET /weather)**
   - **Purpose**: Fetches historical weather data for a specific county and time range.
   - **Parameters**:
     - countyName: Name of the county.
     - typeOfInformation: List of weather metrics (e.g., temperature, humidity).
     - fromDate, toDate: Time range for the data query.
   - **Implementation**:
     - Filters the historical weather dataset stored in data_frame using the provided parameters.
     - Returns a structured JSON response with selected metrics and their corresponding values over the specified time period.
   - **Response**:
     - Success: JSON object containing filtered weather data for the specified county and date range.
     - Error: Returns 400: Missing required parameters or Returns 404: No data available for the given criteria.

Example Response:


```
{
 "county_name": "Middlesex",
 "data": [
   {
```

```
    "info_type": "temperature_2m_max",
    "values": [{"Date": "2024-01-01", "value": 45.3}, ...]
  },
  {
    "info_type": "precipitation",
    "values": [{"Date": "2024-01-01", "value": 0.2}, ...]
  }
 ]
}
```


6 Dashboard design and visualizations

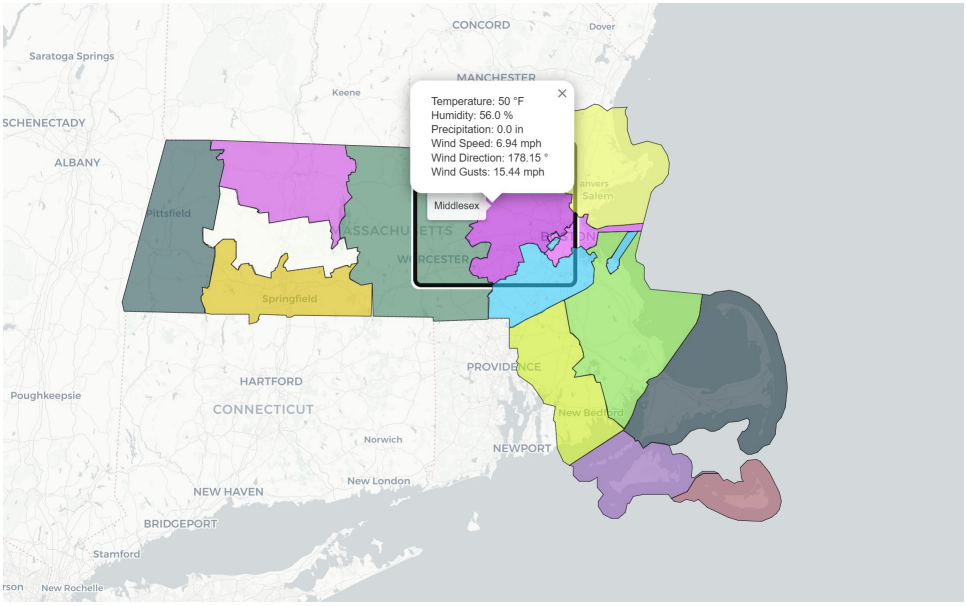

Figure 7: Chloropeth Map - Massachusetts

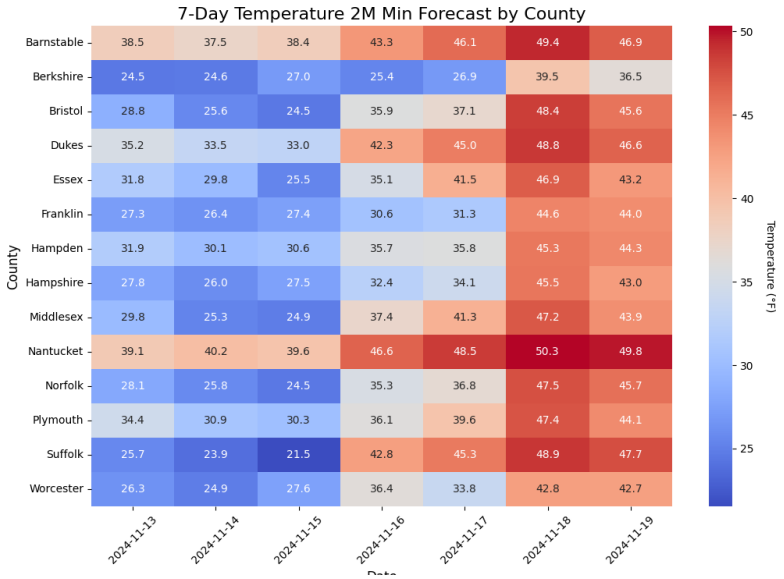

Figure 8: 7-day temperature forecast heatmap

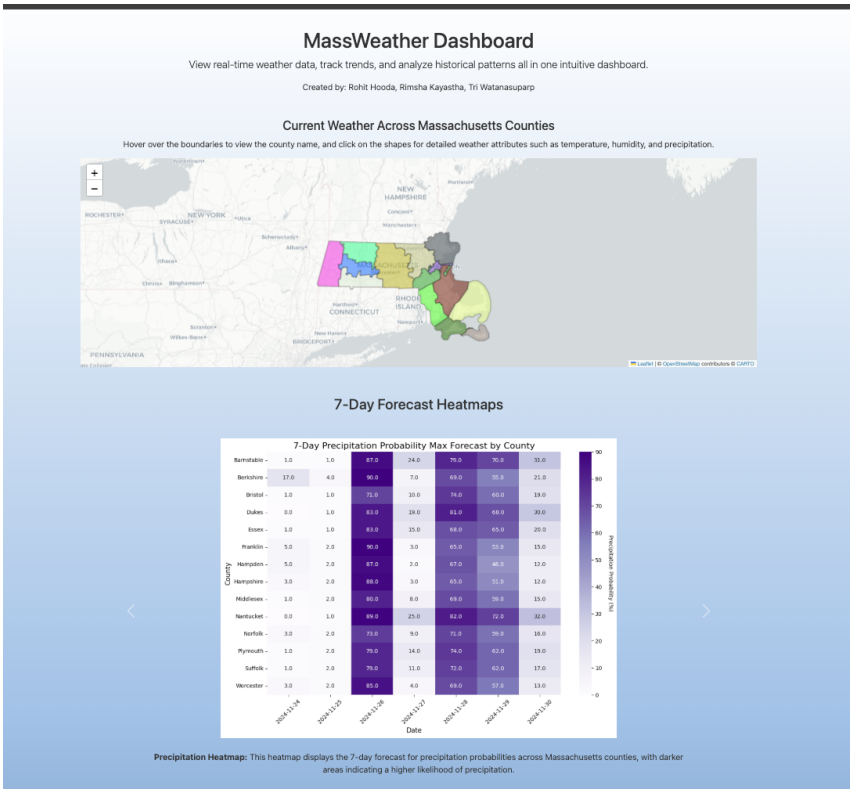

Figure 9 : Website Dashboard

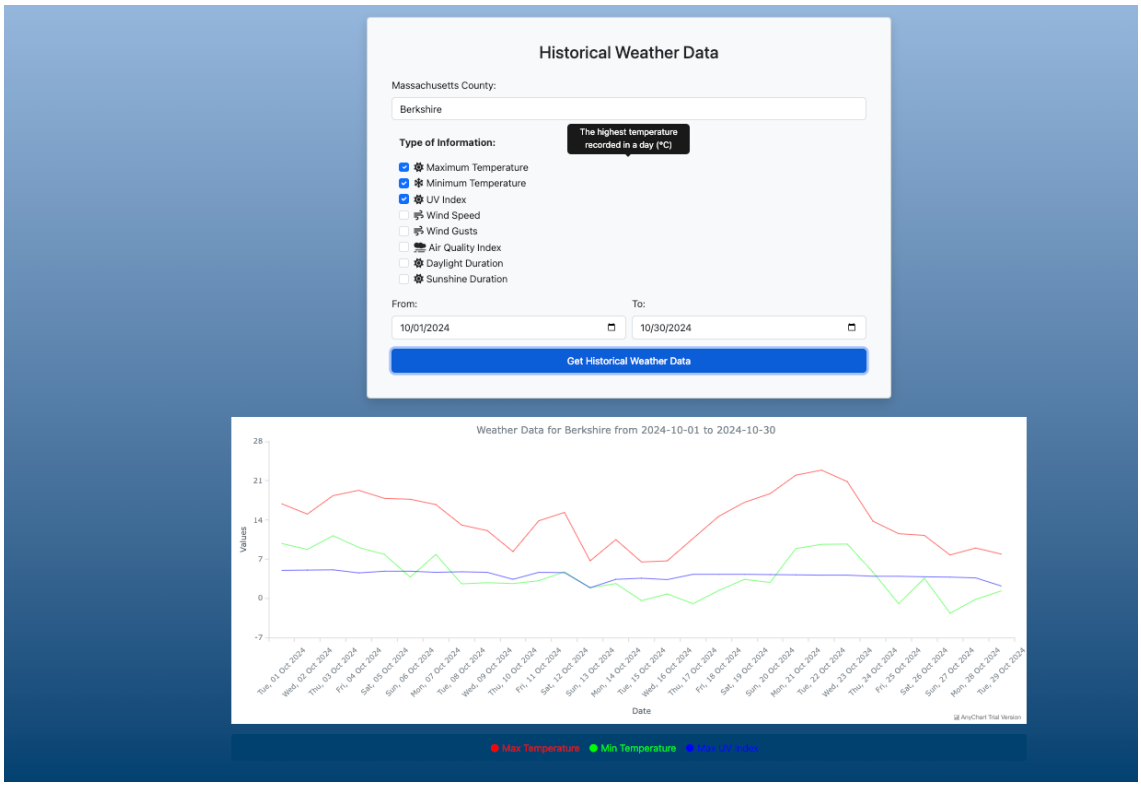

Figure 10: Historical Data Plot

7 Future Improvements

1. **Database Integration**: Store processed weather data in a relational database for scalability instead of calling the API every time.
2. **Authentication**: Implement user authentication for secure access to routes.
3. **Frontend Enhancements**: Serve dynamic maps and visualizations using a frontend framework (e.g., React).
4. **Expanded API Support**: Add more granular endpoints for specific weather parameters

8 References

1. https://www.arcgis.com/apps/dashboards/f7f008a8452747d0a8d86ed69d77cdc4
2. https://ambientweather.net/