Destination SPACE web interface

The objective is to establish a product lifecycle for the Destination Weather Station beyond use during camp. To accomplish this objective, the idea is to create a web portal where students can contribute data collected with the weather station.

With a repository of weather data collected by students, students will be able to analyze a diverse dataset for use in school projects, personal projects, overall data analysis and critical thinking.

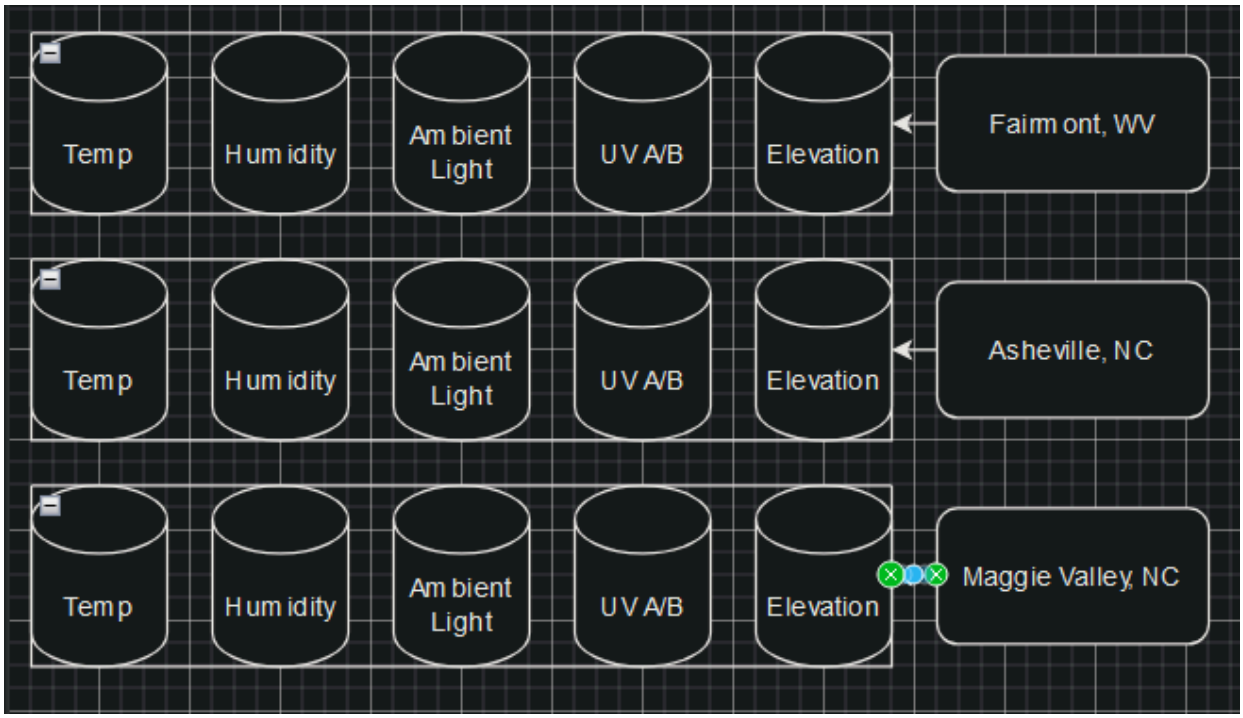
PLAN
The first version of this software will require a database, a way to read data from that database, a way to write data to that database.

Database
https://en.wikipedia.org/wiki/Relational_database
https://cloud.google.com/learn/what-is-a-relational-database

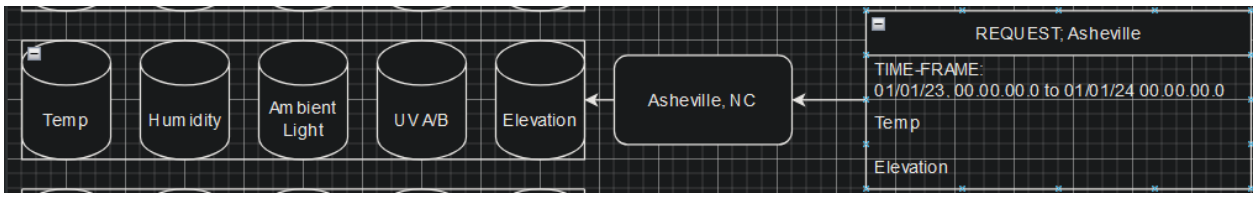
Data will be stored in a series of databases that share common columns or rows. Each of these databases will be tagged with telemetry data which will be built into the request for the data to be viewed on the frontend

Columns will be data type, the row will be timeframe. Data will be submitted in increments of time, the increment amount is TBD, it should probably be whatever increment weather stations measure data in for compatibility's sake.

timeFrame (Column 0)	Data (Column 1)	Data (Column 2)
MM/DD/YY HH:MM:SS.1	12	99%
MM/DD/YY HH:MM:SS.2	12.1	99%
MM/DD/YY HH:MM:SS.3	12.2	98%



Reading Data
Data will be called in batches using a custom API that requests pre-filtered data. There should be built in queries that the user can pick from like the last week of data from any given location. The required elements to build a query will be timeframe (first point, last point), Location (pre-defined list that user can search), data types (columns of DB that aren't column 0).



Once the server receives the request, It will query the DB for the data specified in the request, package that data in a JSON, send that JSON back to the client that requested it. TBD how the frontend will turn that JSON into a nice to view pretty format.

Users will NOT have to authenticate to read data, which is intended to be permanent. This database and data access design is very scalable and performant, even if someone requests the entire database it should not bog down the server, or the network to and from the server as the JSON should not be larger than a few KB and should be longer than 1 minute to parse into a readable format on the frontend.

Writing Data

This will be tricky and will be implemented last

Users will have to authenticate to be able to contribute data. Authentication will start at registration, where the user submits their email and is sent a one time passcode to verify that the email is theirs. They will then choose a password and username. DS will NOT store any email addresses of any users beyond registration. There will be no password reset functionality, I would entertain the idea of manually intervening to reset passwords, but there will be no email addresses associated with users. This is because of the telemetry requirements when submitting data. The combination of general location, email, username, could possibly be used to attach a user to the person behind the username and because of that possibility, this application will not have that functionality to proactively address that problem.

Data submission steps once authenticated:

Clean data;

1. Submit raw data in CSV format
2. On the client, CSV is parsed into the the same format as the DB
3. On the client, the formatted CSV is parsed into a JSON with the same function that is used to read data from the DB
4. JSON is sent to the server where it is added to the DB, not sure how this is going to be executed on technically

Considerations:
How can data be verified as valid and legitimate? Maybe there can be an immutable reference DB that is referenced every time someone submits data that can be used to pass/fail the submission depending on the variance from the reference DB.

What happens when someone submits data from the same location as someone else during the same timeframe? The DB will only have one entry for every given time increment for every variable that is measured

Should data be tagged with the contributors username? What are the security implications of this? Are there licensing concerns with this?

Is there a standard for collecting weather data that we can be compliant to? Could DS potentially contribute crowdsourced data to other repositories?