Team 7 Project Backlog

Anna Benjamin, Kathryn Frankewich, Austin Klasa, Bridgette Kuehn, Matt Molo

1. Problem Statement:

Weather is currently communicated to the general population through means of weather channels and websites that display forecasts for a broad locale. However, some people may prefer more accurate, personal weather data for their specific location (i.e. people who live in rural areas). Additionally, weather enthusiasts may want to gather their own weather data. Our weather utility and corresponding web application would allow such personal monitoring of the weather. By having the weather utility interact with a web application, those using our system would be able to also interact with each other.

2. Background Information:

Audience:

The target audience of this project is weather enthusiasts, Raspberry Pi users and programmers, and homeowners. By making our software open source, it will allow such users to have access to the program and have a personal weather experience.

Similar Applications or Systems:

There are several other similar weather systems that allow you to set up a wireless weather kit outside and monitor the data from inside your home on a small monochromatic or LCD screen. Additionally, there are personal weather stations that cost on the upwards of thousands of dollars, that have high overhead to set up. Other options to get weather include accuweather.com or weather.gov.

Existing Limitations:

The aforementioned similar applications are usually riddled with a myriad of advertisements. Also, some of the frequented applications such as accuweather.com and weather.gov are limited to weather systems that gather data via weather stations that happen to be high tech, expensive, and impersonal. Due to this restriction, these more advanced systems are limited as to where their weather stations are located, and their data in turn is also limited.

In contrast, Personal Pi in the Sky will be ad-free and modifiable in order to meet the needs of anyone who desires to know the weather at any point in time. Data will be easy to grab via an open REST api, and software can be downloaded and modified to complete the plethora of tasks the user may wish to accomplish.

3. Requirements:

Functional:

1. As an amateur meteorologist (web application user), I would like to:
    a. View weather data from all weather stations.
    b. View weather data from a specific station.
    c. View weather data on a mobile app (if time allows)
    d. Customize what weather data is shown.
    e. View historic weather data graphically.
    f. View current weather data.
2. As a weather station owner, I would like to:
    a. Build my own weather station.
    b. View my personal weather station data.
    c. View aggregated weather data.
    d. Publish well-formatted data for analysis of the weather.
    e. Choose to publish my personal weather data.
    f. Integrate my weather station data with a mobile app (if time allows)
3. As a Raspberry Pi developer, I would like to:
    a. Modify the source code to build upon the framework.
    b. Write the code for the weather sensors in the Raspberry Pis.
    c. Publish the aforementioned code.

Non-functional:

Architecture

The web application will use a client-server architecture to handle the users and their chosen actions. There will be a single central database that each weather station connects to. The database will supply the web application with the necessary data that is requested by the user.

The database, or backend, will be developed using MySQL and Java and the web application, or frontend, will be developed using JavaScript, PHP, HTML, and CSS.

Performance

Performance is an essential part to the design of our weather system and web application. Our weather system that will be running on the Raspberry Pi should be able to collect data points at frequent intervals as well as send the collected data to the database in thirty seconds. Additionally, because our web application is going to be retrieving significant amounts of data, both current and historical, care will be taken to try and have loading times less than 10 seconds when attempting to view data. A timeout error message will be implemented if loading becomes excessively long. However, the system will attempt to retrieve the data the user requested in the quickest and most efficient manner.

Security

Security is not a very important issue for this project because users will not have accounts. Rather, those with their own raspberry pi weather stations will have a randomized number assigned to their device and be able to give their weather station an alias name. Weather station owners will be able to choose whether or not to keep their data private through settings in their weather station. Since no personally identifiable information will be stored with their data, the amount of risk associated with a breach of security for this collected weather data is minimal.

Scalability

Scalability is a concern, because the amount of weather stations will not be limited. The database must be able to handle as many weather stations that users register. However, the more weather stations, means more data will be saved in the database. The database will also have to be able to search through itself fast enough, so that web users do not experience a delay in looking at data. The web application should also be able to handle the traffic of several users accessing it at once.

Usability

The web application will be designed in a user-friendly manner. To achieve this, the web application will use colors that are easy on the eyes and enhance the user's experience. Additionally, the web application will have a logical, organized layout that is arranged for efficiency. No advertisements will be present on the web application, which

will allow users to have a smooth, uninterrupted experience. Lastly, the web application will include a page dedicated to teaching users how to build their own weather station step by step.