Team 7 Sprint Two Planning Document

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

Sprint Overview:

This planning document will describe how our project will be divided into each member's tasks for the course of sprint two.

Goals:

- Continue to improve the design of the web application
- Allow a web application user to view weather data that has been collected
- Allow a web application user to filter weather data based on different weather criteria
- Collect weather data from weather stations and their sensors
- User will be able to configure startup settings for their weather station

Scrum Meetings:
We will meet every Tuesday, Thursday, and Saturday from 3:00 PM to 4:15 PM for our SCRUM stand up meetings with all group members that are available at the time. We will discuss the progress of each task and any issues that have arisen.

Scrum Master:
Kathryn Frankewich

Risks and Challenges:
We have a few challenges for this sprint. Some team members will be working with languages that they have not been exposed to before. Also since multiple team members will be working on different components of the project, communicating between different components, including the database, server, and web application, will be a potential challenge.

Current Sprint Detail:

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.
2. As a weather station owner, I would like to:
    a. View my personal weather station data.

b. View aggregated weather data.

Task Description	Owner	Estimated Time (hours)
Create web application side PHP code to handle a request to filter data from a user. Once a filter is created by a user and submitted, it will be sent to the backend.	Austin	15
Web application side JavaScript geolocation code to get the location of the user. This will then be sent to the backend when the home page is opened.	Austin	10
Implement server function to accept data filtering requests from web application	Bridgette	5
Implement controller functions to parse and retrieve filtering data	Bridgette	15
Retrieve filtered data from database using functions in Model and send back to web application	Bridgette	5
Modify controller code to accept a user's location and retrieve information from the weather station closest to them	Bridgette	5

Non-Functional web application code:

Task Description	Owner	Estimated Time (hours)
Refactor web application code (pages are created using a single page template).	Austin	5
Create a mobile friendly version of the website.	Anna	10
Ensure both versions of the website are responsive.	Anna	15
Make the website easy to navigate so the user can easily find what they are looking for.	Anna	5

Non-Functional Weather Station Tasks:

Task Description	Owner	Estimated Time (hours)
Write code for temperature and humidity sensors	Matt	8
Write code for pressure sensor	Matt	5
Write code for light sensor	Matt	2
Build wind sensor (3d print)	Matt	5
Publish JSON data to web server	Matt	5
Respond to web queries with correct JSON data	Matt	5
Implement initial setup asking users about settings	Kathryn	15
Implement saving and opening app settings	Kathryn	9
Assign UUID and publish location and alias to database	Kathryn	6

Team Member	**Hours**
Anna Benjamin	30
Austin Klasa	30
Bridgette Kuehn	30
Kathryn Frankewich	30
Matt Molo	30
TOTAL:	***150***

Backlog:

Functional:

3. As an amateur meteorologist (web application user), I would like to:
    a. View weather data on a mobile app (if time allows)
    b. Customize what weather data is shown.
4. As a weather station owner, I would like to:
    a. Publish well-formatted data for analysis of the weather.
    b. Choose to publish my personal weather data.
    c. Integrate my weather station data with a mobile app (if time allows)
5. 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:

1. Performance:
    a. Make sure pages load in 3 seconds or fewer.
    b. Make sure the server, database, and Raspberry Pi clients send/receive information to and from one another in less than one second.
2. Security:
    a. Make sure people can successfully keep their data private.
3. Scalability:
    a. Make sure that the system can handle as many or as few clients as necessary.
4. Usability:
    a. Make sure the users can easily and quickly figure out how to find their weather information on our product.
    b. Make sure users know how to customize the interface to their liking.