MSU Greenhouse Project Requirements Document

G. Mathers, D. Pollock, D. Portillo, K. Pulliam, E. Mondragon

2/21/2023

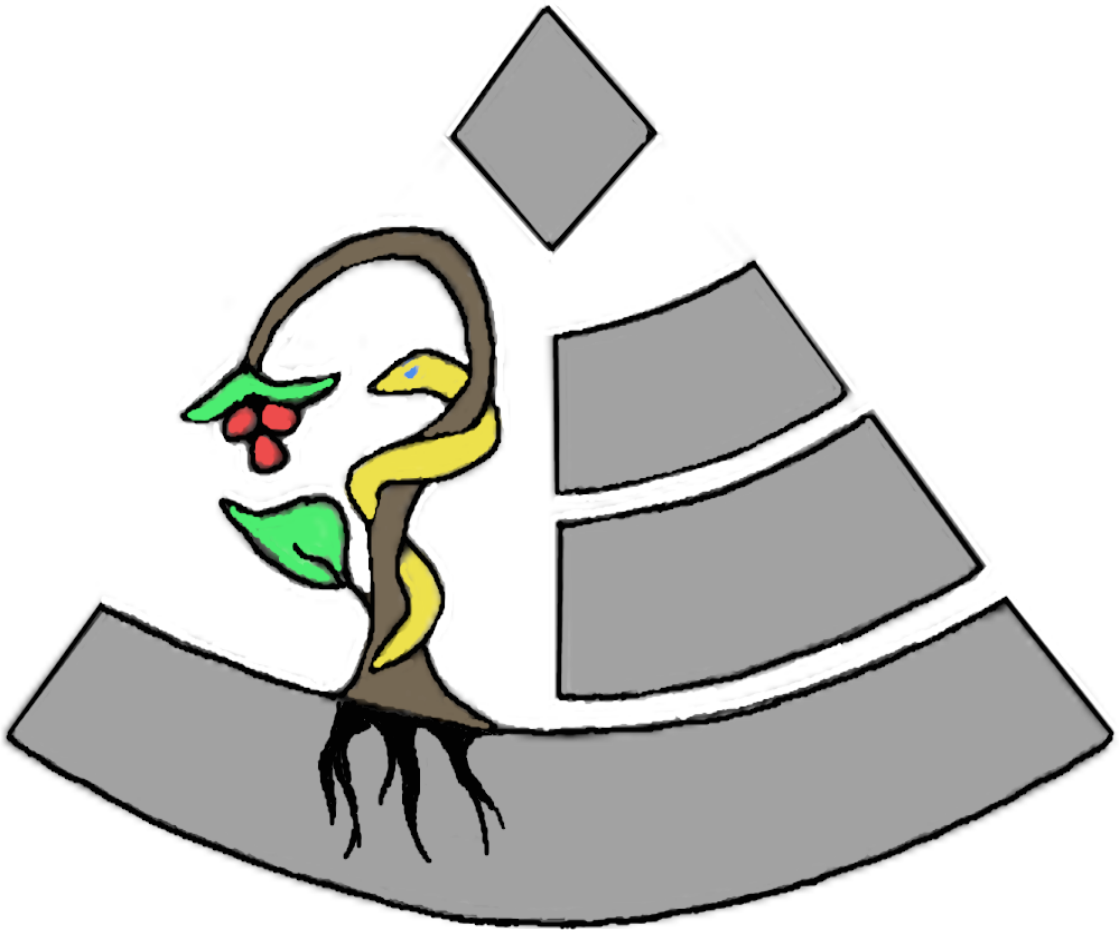


Table of Contents

1. Introduction/Overview

The purpose of this document is to describe the current greenhouse setup. There are only analog controls of the system that facilitate the inner functionality of the greenhouse such as opening/closing blinders, increasing/decreasing temperature in the various rooms, increasing/decreasing airflow, etc. With this extremely manual system there is no way of knowing the current/past temperature and humidity measurements of the various room environments, which does not allow for effective experiments among the professors and grad students. To help solve this inefficiency we have proposed a web application capable of taking temperature and humidity measurements, storing these measurements in an organized database, and displaying the measurements in a visually appealing user interface. This paper is organized as follows, first we will discuss the scope of the project, then move to who the users are and the needs they foresee for a more effective and further advanced MSU greenhouse.

2. Scope

To deliver a list of features that is sure to grow over the lifecycle of this project. Surely the entire scope of this project will be very difficult to fully accomplish which is why we have narrowed down the huge wishlist originally given to us to two main baseline components of the greenhouse. Temperature and humidity. The front end will be built around these two units of measurement and will allow further incorporation of features as time progresses. In the home page the user will be able to add different rooms to monitor and be able to name them. With this comes the “agent”, the agent will be produced upon adding a room to the home page. An executable file will be downloaded into the browser and all the user has to do is execute it on the target raspberry pi that will take the temperature and humidity using the Sense HAT’s described later on in the document. This will automatically associate the raspberry pi with the server and begin taking measurements for this room. There will be one more templated page within the UI that will show the individual room statistics over time. We aim to provide charts that can have dates specified to show data gathered from the database on the individual rooms. Filterable on both temperature and humidity.

2.1 - Main objective

The group was formed together to build a scalable application that is able to gather various units of measurement in the MSU greenhouse and display them in a database fed user interface. Creating clean, readable, commented code is essential for the groups that will come after us and will be a main focus.

2.2 - Specific goals

We have many goals in this project. The first being a working backend system that is able to gather the established units of measurements and store them in a database. Secondly, a server that serves up the HTML/CSS using templated data for the front end. Thirdly, a dynamic, well equipped front end that showcases multiple features not originally discussed. Setting attainable goals is key in this project, and we have set ones that we think are reachable by the end of semester.

2.3 - Further goals

Features that are outside of the scope, but may be considered in future course projects are boundless in terms of potential work. With proficient funding in real industry grade equipment that can be either soldered onto the raspberry pi or separate functioning equipment with interfaces that can also communicate to the server over HTTP, features can be stitched to the application easily because of the documentation that will be featured in the Github repository. There had been talks of purchasing CO2, ethylene gas, light measurement tools they had wanted to include as well as the ability to control the inner functionality of the greenhouse itself. Funding provided with the expansion of Bolin hall could be allocated toward upgrading the controllable functionality in the greenhouse past that of a completely analog system. If the new system includes an application programming interface, it could be integrated into the application as well. This kind of system could control functionality such as irrigation, ventilation, mist/fog, carbon dioxide enrichment, application of fertilizers, etc. So long as the systems purchased are considered for functionality within the application this project could go for several more iterations.

2.4 - Overview of document

In the remainder of this document we aim to explain our application in much greater detail. Going over who the users are, the functional/non-functional requirements, user interface, hardware requirements, etc.

3. Users

The intended users of this project are professors, and other authorized personnel, of the biology department who preside over and conduct research on the greenhouse as well as any faculty or students who they give permission to use, maintain, or take measurements of the environmental conditions of the Greenhouse. These individuals will be known as Admins. Other individuals not conducting research or are students will be known as Users.

3.1 - Use Cases

The intended use cases as shown in a simplified manner in Diagram. 1 and in far more detail in Diagram 2. for Users are:

- Observe Data (Temperature and Humidity)
- Manage user notes (notes written by the same user on the rooms.)
- Login and Logout of the system

The Use cases for Admin are far more extensive as noted in Diagram 2. Which also encompasses much of the same use cases for Users.

- Observe Data (Temperature and Humidity)
- Manage All notes (notes written by any user)
- Login and Logout of the system
- Manage Users (Add/delete users, ect.)
- Manage Alerts (Alerts that automatically send emails should certain thresholds on measurements are met I.e. Temperature being too high or low.)
- Manage Rooms (Adding or deleting rooms within the greenhouse
- Manage Agents (Agents are responsible for automatically taking measurements and compiling them on the database)

The Use cases for Rooms Agents are simply:

- Accessing and logging General Data into the database

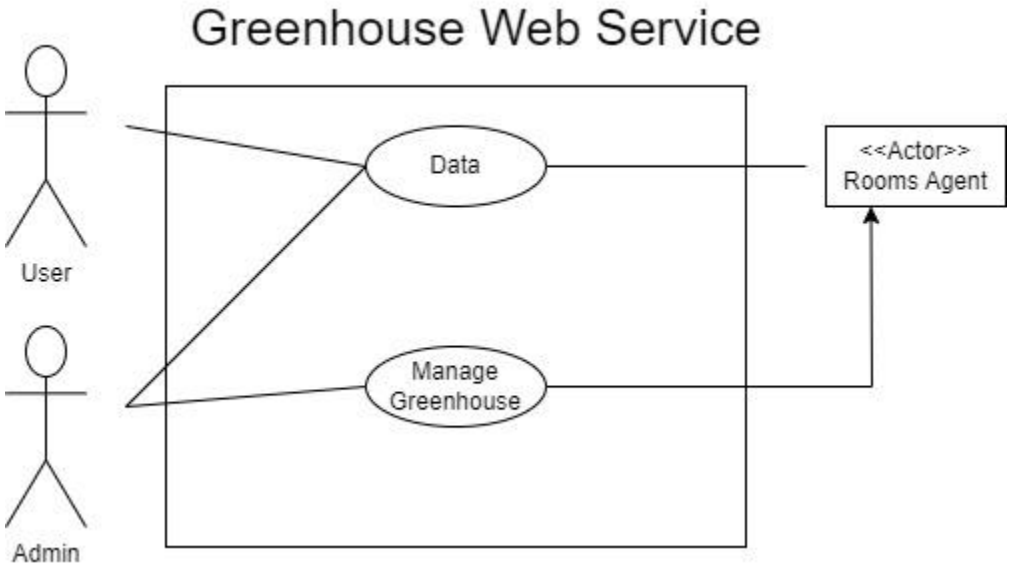


Diagram 1. A simplified overview of the Use Case Diagram.

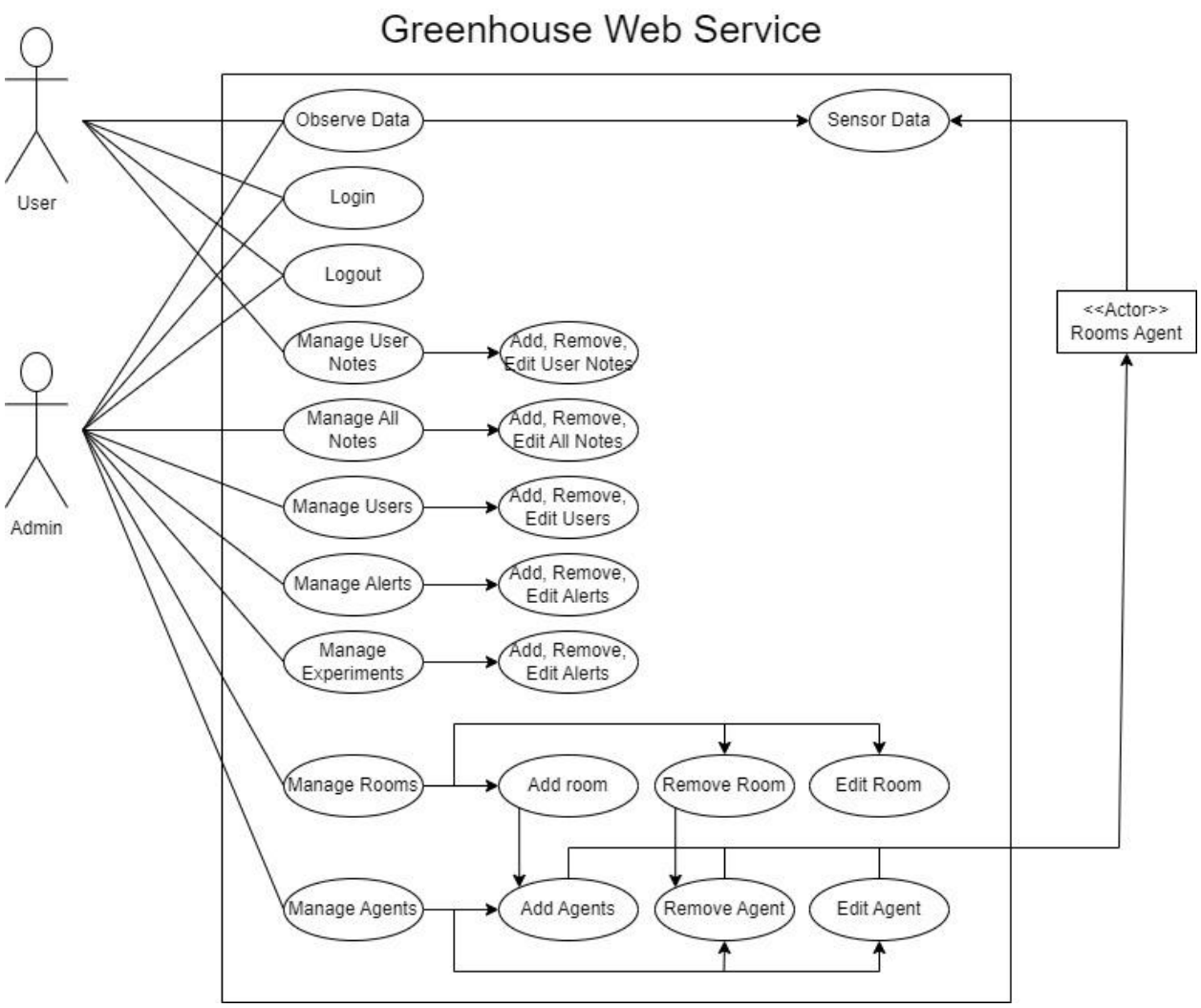


Diagram 2. Detailed Use Case Diagram

3.2 - Scenarios

Some potential scenarios in which the system would prove useful would include long term research on plants that would require careful measurements of environmental conditions, using the measurements to adjust conditions in the greenhouse to account for drastic changes in temperatures, and to provide data that might inform any issues from any systems within the greenhouse malfunctioning. Additional others include setting up the 4th empty room inside the greenhouse for use or simply placing notes on when the last time the plants were watered.

4. System

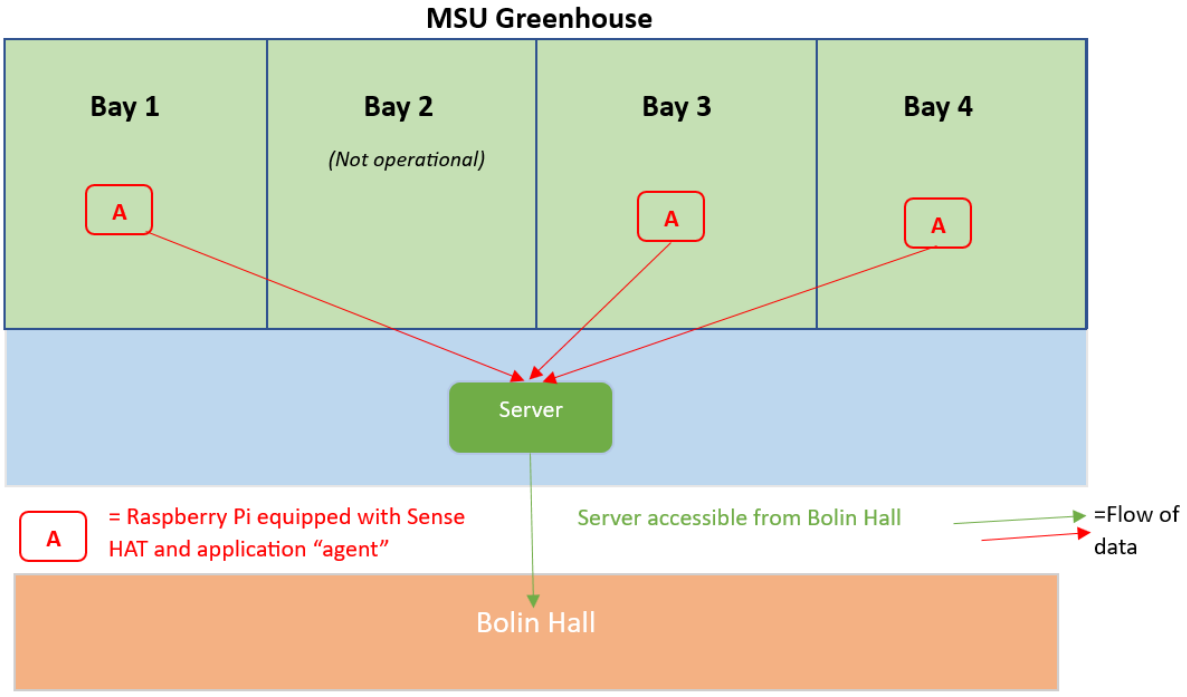
The following sections will explain the system and how the MSU Greenhouse group is approaching the software. Many parts of this section are technically worded. I will highlight the words and provide explanation in section

4.1 - Development Environment

We will be able to develop this application from virtually anywhere. The backend team will simply need access to the **raspberry pi's** and sense **HATs** while the front end team will develop the UI from their desktops/laptops. For final testing of the system we will have everything setup and deployed in the greenhouse. Hardware requirements for this project are minimal, as it will be accessible from the browser. Backend development tools will include **Python**, **Flask**, **SQL**, and various other modules. All the functions will be routes communicated over **HTTPS.** A **REST** like approach is being taken to the back end.

4.2 - Target Environment

The environment shown below is what we are working with. It shows the approximated greenhouse, agents on the **raspberry pis** that are gathering data, the server, and flow of the data. The arrow from the server to Bolin details how we will build the server to respond to requests from staff browsers only. This adds an extra layer of security from the outside world.



4.3 - Functional Requirements

Gather and store temperature/humidity data from raspberry pi's across the greenhouse in the servers database. The server should be able to respond using multiple defined endpoints to the UI/Agents. A login functionality within the UI should be secure and able to effectively authenticate users. The ability to add/remove rooms in the greenhouse environment and associate agents with those rooms to collect data for them. Within the home page we should have a settings section where you can add users using administrator credentials and define the alert thresholds/methods for each room. Also, the home page should have a preview of all rooms in the greenhouse including current temperature, humidity, and brightness. Each room should have charts that can take date ranges and produce visually appealing results. Alerting mechanisms in place that alert on specific temperature/humidity thresholds predefined in the individual room configuration (email, SMS?) for whatever works best for the professors.

4.3.1 - Issues

The first issue that comes to mind is gaining access to the staff wifi network. I will need to do more research on how the networks are laid out and if they will be easily accessible from the greenhouse. We may need to look into having a network administrator as a contact for this project. Another will be connecting the front end to the back end and having the data be templated from the database into the **HTML**. This will simply be tricky, nothing that will prohibit us from completing this portion. Lastly, budget, we have received $400 which all went into the temperature/humidity taking abilities and the 4 **raspberry pis**. This left us no money for the desired CO2 reading functionality.

4.3.2 - Major Subsystems

RoomAgent will handle functionality of the application on the raspberry PIs equipped with the **Sense HATs**. The main point of this system is to package the temperature, humidity, and brightness readings that will take place every 3 minutes and send it to the raspberry pi acting as the **RoomsServer** When a **RoomAgent** is generated by the user adding a room, it will be pre configured based off of the room name to report back data for the room the user specified. Room names must be unique. **RoomsServer** will handle all of the requests coming from this constant influx of requests every 3 minutes or so on average from each **RoomAgent**. There will be many predefined requests/response scenarios built into this sub system all documented within a swagger-ui that they will be able to ask for in the "{server_ip}/docs" route. This will make the documentation for the backend auto generated and in a very easily understandable format.

4.3.3 - Major Functions

- observe(data) checks updated values as they come in for each room that sends back data. If any room experiments are set to alert on a specific threshold of temperature/humidity it will launch an email to the alerted emails. They can also set a timer on how many alerts they want to see and at what frequency. Will run with the post method for measurements.

- email_alert(room, data, message_type=”threshold”) grabs the room value and does a check of all users who have it set for them to be alerted. It will then send each of them the message_type directly to their email. Message type is defaulted to threshold for now since that is the only functionality it has. But if the agent is expanded to service other measurement tools in the greenhouse rooms we could add “message_types” to be sent with the data and room name to the user. Could even make them HTML that we have regular expressions parse through and place the threshold values that were violated on that given reading. They should also be able to set the amount of alerts they get when errors occur in the configuration settings for the alerts.

- ensure_check_in(room) will communicate via ping “agent_ip” and check to make sure it is still reaching the device that is hosting the agent. I like the idea of the server checking in as well to add the pre-test for any troubleshooting of the software. Can also measure the amount of time it has been down if the agent goes down for some reason. This can be an alert to any admins listed to monitor the greenhouse.

4.3.4 - Major Classes Identified

- **Greenhouse**: is the main class of the application, this class defines the individual greenhouses instance. A name can be assigned for the home page of the U.I, which will also be the greenhouses name. Any room created within the U.I will be listed in the Greenhouse object. All experiments created within the rooms will be listed here as well, as the server and any users.
- **Measurement:** will provide the appropriate units and formulas to handle everything data wise for temperature. Convert_kelvin, convert_farenheit to name a couple class methods, temperature in celsius as self attribute.
  will provide the appropriate units and formulas to handle everything data wise for humidity. Percent: float, and a timestamp of when it got the data.

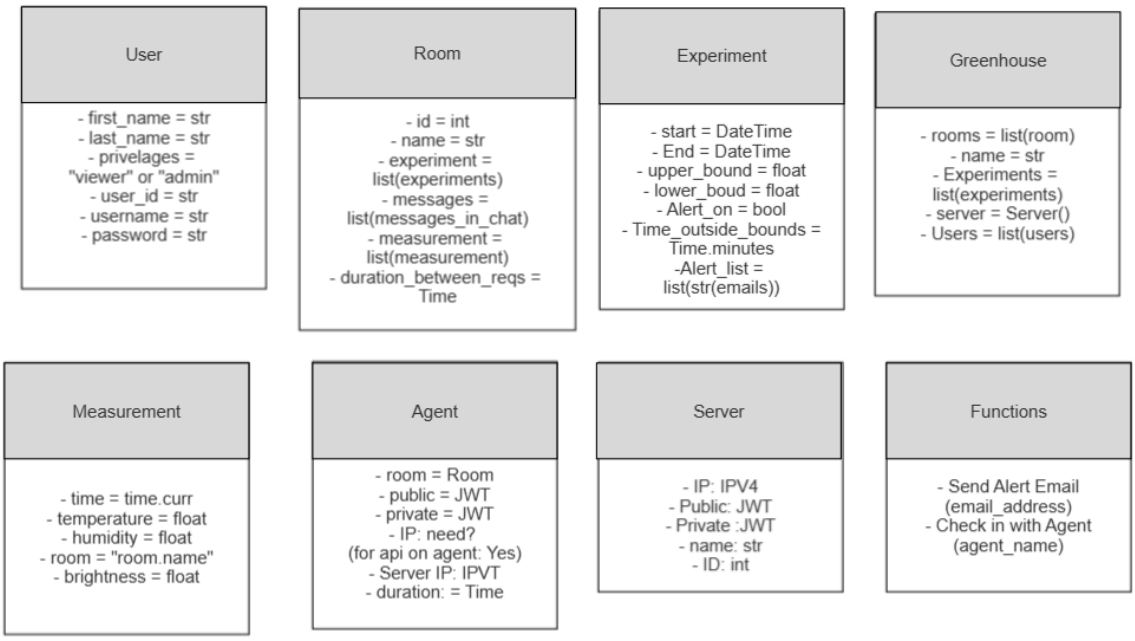
- **Room**: will represent a newly created room. Holds attributes like name, id, agent_id, readings=[(temp,humidity,brightness,timestamp)]. Creating one will only need a name for the user. Once the room is created, the Agent .exe will begin downloading into the browser. All the user will have to do is download the agent onto the desired raspberry pi equipped with a Sense HAT. (This is desired usage capabilities) We will also write documentation on how to set up more of them and configure them to where they are able to report back to the server.
- **Server**: Will hold any attributes about the server, such as IP address, name, public and private API keys to communicate with the agents it has assigned to it. Essentially defines all information needed to communicate to the API and in turn, the database.

- **Agent**: the individual agent that is produced for the room upon creating a new room and downloaded directly into the browser. This class will handle creating the new agent for them with the correct details and responding with it whenever a new room is created. Main functionality would be to grab the data from the Sense HAT and send it straight back to the server, fall asleep for 3 minutes as a background service on the raspberry pi, wake up, and do it again. 3 minutes is fairly random, this duration will be customizable within the agents respective room page.

- **Experiment**: will add great functionality within the room page. An "Experiment" can have a start and end date, upper and lower temperature/humidity thresholds, the option to alert with an email whenever these thresholds are set, and even specify which users' emails will be notified whenever the alert is sent. A special variable will be placed to measure the total time outside of the thresholds as well. Being able to track the state of the experiment as it went can be a very powerful tool, especially when being able to measure how much liquid was used to water the plants, how much fertilizers were used, the level of CO2, etc.

- **User**: A user will hold attributes like first_name, last_name, username, password, email, phone, admin=False. This is the user account used to login to the web application. Only an admin user will have the option to create users and if they will be admin or not.

Figure on the next page documents the database models/tables and their corresponding columns/attributes.



4.3.3 - Minor System Functions

- charts_range_change(start, end, room) given a start and end time this function will grab all of the database entries in a given start and end date the user specifies. Used in the event of the user changing the specified chart ranges.

- post_message(user_id, message, room, time_stamp) will post a message to the rooms messages attribute linked to the user who posted it.

- authenticate(username, password) for the login screen to check the validity of the username/password combo. Value will lay in the database hashed, so nobody can view the username/password in storage in case someone obtains the database.

4.4 User Interface Specifications

4.4.1 - Minimum Viable Product

At the least the front end team will be developing a login screen using HTML/CSS as well as the page that shows a preview of each room and its current readings for temperature, humidity, etc. This will be the very least we accomplish in case getting the UI to communicate with the backend gives us more trouble than expected. The customer would like for us to develop a user

interface which contains a login screen and then a screen that will show the data for each of the bays of the greenhouse.

4.4.2 - Plans for UI

Once we have accomplished developing the minimum viable product, we would like to add the following features:

- Viewing each individual room in depth
- Notes sections for each bay for non-verbal communication
- Ability to add a room to the readings screen
- Application accessible on all platforms (web, mobile, mac, windows)
- Have a filter option to show readings over different periods of time
- Have a settings option for admins to adjust rooms' floor/ceiling for readings

4.4.3 - Overview of UI

- **Login Page**
    - Text Boxes for user email and password
    - Account authentication will communicate with the server on the Raspberry Pi
    - Ability for user to sign up for email updates on the greenhouse bays
- **Home Page**
    - Displays preview of all rooms with live readings from the server (Temp. and Humidity)
    - User is able to select a room which will take them to the page for that specific room
    - Settings cog that will take user to set up screen or a settings popup window within the home screen
    - Notes section that regards to the entire Greenhouse
- **Room Page**
    - Each bay of the greenhouse is accessible via a container on the room page
    - Containers display a preview of the rooms stats with charts or measurements
    - Settings cog for user to change settings for the room
    - Collapsable notes section within room page
- **Settings**
    - User is able to adjust the floor/ceiling level that the temperature and humidity should be above or below
    - User is able to adjust the frequency that the agent will fetch the readings from the server

        - If admin user is able to adjust settings for other users and add or remove users

- **Notes Box**

    - Hideable notes section for each individual room
    - Notes section will be used for users to leave notes whenever they have checked on a bay or attended the plants
    - Different Icons or labels for notes pertaining to a specific experiment and indicating what form of work was done

4.4.4 - Development Tools

- **VScode**

    Visual Studio Code will be the IDE for developing the user interface section of the project. VScode allows for simple installation of extensions useful for several types of development. With built in source control tools VScode enables seamless collaboration between the frontend developers of this project. VScode is also supported on all platforms including web and mobile, giving flexibility needed to develop this project.

- **JavaScript/HTML5/CSS**

With the complexity of the user interface that is planned to be developed, the web development process will contain communication with the backend and dynamic showcasing of the data being sent from the backend. JavaScript will function as the backbone of the frontend primarily within the section of the user interface which communicates with the backend. We aim to develop an aesthetically pleasing user interface that will not require updating within the near future. The combination of HTML and CSS will allow the front-end developers of this project to design and implement a professional user interface and experience for the faculty and graduate students using the *GreenWatch* remote monitoring system.

4.5 Non-Functional Requirements

4.5.1 - Budget

Due to the nature of this project, multiple hardware pieces will be needed in order to meet the requirements. The total cost is estimated to be around $400. There will be a collective review in each stage of the project to ensure that deadlines will be met and features are implemented properly.

4.5.2 - Management

In order to ensure all objectives will be reached within a timely manner, **Trello**, a project management tool, will be used to keep all members on schedule. **Discord** will also be used as a means of communication within the group. To keep the project code organized, the version control software **Git** will be used alongside **GitHub** to store the repository remotely. Every member will be listed as a contributor for the repository.

4.5.3 - Technical

To allow for complete interoperability on any platform, the software will be built as a web application. This will involve the use of HTML, CSS, and JavaScript for the initial prototype on the front-end. The back-end will utilize Python and REST APIs to send data between all devices. Originally, developing a windows desktop application with Visual C# was considered. However, due to the chance of a user not having access to a windows system, choosing the web application route was a more viable choice.

4.5.4 - Performance

This application is expected to return data reliably as the condition of the plants in the greenhouse is of high importance. It is also a priority to provide a smooth user interface so its users can navigate the software with ease. Efficiency is not a concern in this software, just as long as the data is consistent across all instances.

4.5.5 - Security

Since this web application will be publicly available on the MSU network through a static IP address, basic username and password authentication will be used to secure access. Also, each user account will have different levels of access. For example, a basic user will only have the ability to view the data acquired. An administrative user will have the ability to set a threshold on the data, which will send out an alert via email if the threshold is compromised.

4.7 Hardware Requirements

When it comes to the hardware aspect of the setup, there will be a lot to be integrated and taken into consideration. At the moment the list goes temperature, humidity, carbon dioxide (CO2), and the measure of light in the room. The start of this project is to gauge out if there are components in the market to monitor the state of a greenhouse. Seeing what the components are capable of first can give an idea the number of sensors needed and how much distance is needed to have optimal data. Figures 1-4 show images of the hardware necessary for the project and are located below.

4.7.1 - Raspberry Pi 3b

Raspberry pi 3 is a micro controller in the PI series. This development board, as seen on Figure 1, is a single board computer that works on the Linux operating system. The board contains 40 pins, uses include processing operating voltage (3.3V), raw voltage input (5V, 2a), even includes flash memory of 16 gigabytes (SSD memory card). With the board being supported by Python, this language is the one that will be mainly used when it comes to hardware.

4.7.2 - Sense HAT

Sense HAT is an add on board that extends the capabilities of the raspberry pi. With it some of the extensions include monitor pressure, humidity, temperature, color, orientation, and movement. Though for greenhouse monitoring humidity and temperature are the two main ones that will be implemented. As seen on Figure 2, the 8x8 white matrix that is visible is a LED matrix that acts as a visual aid for what is going on with the sensors.

4.7.3 - Photodiodes

Photodiodes are sensors that are used to measure the intensity of light. These diodes, as seen on Figure 3, have no internal gain but can operate at much higher light levels than other light detectors. With this it will be possible to measure the amount of light being laid on the greenhouse room. The diodes can also operate at high operational speed and is the best option when it comes to light measuring.

4.7.4 - K-30 CO2 Sensor (Probable expense)

K-30 is a carbon dioxide (CO2) NDIR sensor, where it has the potential to run for a few years. Compared to the most common electrochemical sensors, NDIR are more accurate and have a faster reaction rate. One of the couple reasons that the K-30 is being used more recently is because it is much easier to talk to.

Figure 1. Raspberry Pi 3

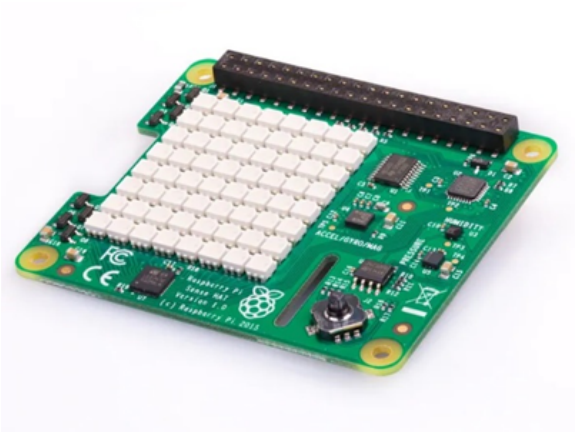


Figure 2. Sense HAT

Figure 3. Photodiode

Figure 4. K-30 CO2 Sensor

5. Other Deliverables

At the minimum, the software requirements, project plan, user manual, github repository and a minimum viable product should be completed by the end of this semester.

5.1 User Manual

The user manual should include a step-by-step, in-depth guide that properly explains how to use the web application. It should contain visual assistance to help explain how to navigate throughout the application.

5.2 GitHub Repository
This online repository will contain any documentation, layout designs, and source code used throughout the lifetime of the project. Within the repository, a list of all past contributors should also exist.

5.3 Developer Manual
The purpose of this manual is to allow new developers joining the project to get brought up to speed on the inner workings of the software. This is to prevent any misinterpretations of documentation in the future.

5.4 Requirements Document
This document itself will be passed along so that a record of the start of this project, and all requirements decided upon in the beginning, may be kept.

5.5 Project Plan
The project plan will include the majority of features from the requirements document, as well as desired features yet to be implemented. A roadmap will be used to showcase the progress made thus far.

6. Risks

There are several factors that could inhibit the process of the greenhouse project, both in the future and present. It is imperative that it is taken into consideration that these risks can ultimately hinder the project. Water getting into the raspberry pi boards, some unknown entity getting into the application, physical damage to the raspberry pis, etc.

6.1 Hardware Risk
Since actual hardware is going to be used for the development of the project, it must be said that the equipment will be at risk of being damaged. Some of the issues at hand can include severe weather, watering of the plants, clumsy people, and so on. Though it is easy to think of potential hazards, the solutions are equally as easy to follow the ideas. Creating a box to protect the sensitive parts of the hardware can be created (as seen on Figure 5), the placement of the box could also play a big role. Another risk that can be taken into consideration is the capabilities of the hardware. A great example are the photodiodes, even though they are capable of operating at high speed, they are temperature dependent and have poor temperature stability. When it comes to risks with hardware it all becomes situational and most of the time it is not known until it happens, that is why it is best to visualize everything before it happens.

6.2 Software Risks

When it comes to the issues that involve the language of the machines, it starts to become difficult to dial down where specifically the problem is. Some of the common cases that present themselves are faulty data readings, someone who is not in the program accessing the information of not only the greenhouse but the users as well. When it comes to these situations it is best to do occasional tests to ensure that the readings of the system are close to accurate. Some of the other risks that need to be factored are security within the log in system itself. Though one of the main reasons that a localized network was decided was based on the idea that user information needed to be considered.

Figure 5. Raspberry Pi Board in environmental casing

6.3 Risk Probability

Figure 6. Risk Probability Table

Risk	Probability(1-5)	Severity(1-5)	Prevention
Hardware Failure	3	5	Keeping an eye on the hardware, and observing any anomalies in the data.
Being hacked	2	5	With it being a localized network, the only way is to ensure not to go around sharing the site to unknown people. The use of JWT's in the backend for user authentication will also limit this probability.
Inaccurate Readings	2	3	Having an occasional data matching, compare the data recorded by a person and the system.
Damage to hardware	4	4	Only way to prevent is by protecting and being cautious is best when working with these sort of systems
User Interface complications	1	3	When it comes to things integrated to the system the only way to prevent is by knowing what issues could happen
Loose wiring	2	4	Having as little exposed as possible is the best way to prevent wires being tripped on

7. Glossary of Terms

Raspberry Pi: A mini computer that runs a small O.S called raspbian. We will use them as data gatherers throughout the greenhouse.
Sense HAT: The Sense HAT is an add-on board that gives your Raspberry Pi an array of sensing capabilities. The on-board sensors allow you to monitor pressure, humidity, temperature, color, orientation, and movement.
Linux: An operating system under the Unix umbrella of operating systems.
HTML: Hyper-text Markup Language. The standard document template to be displayed in a web browser.
CSS: Cascading Style Sheets. A style sheet language used to define the visual aspects of a document created using some markup language.
JS: JavaScript. A high-level computer programming language.
HTTPS: Hyper-text Transfer Protocol Secure. Ensures connections are secure over a network via encryptions.
Python: High level, general purpose programming language used to build out the logic of the application. It is also what the server and database will be interfaced with.
Flask: A python module used to build servers.
SQL: Standard query language. A relational database management tool.
REST: Representational state transfer

8. References

A. Wilcox-Hirst, "**Interfacing a K30 CO2 sensor with a Raspberry Pi for Remote Air Quality Sensing**," *Compute Nodes*, 17-Dec-2020. [Online]. Available: https://computenodes.net/2017/08/18/__trashed-4/. [Accessed: 13-Feb-2023].

Initial State, "**How to build a raspberry pi temperature monitor**," *Medium*, 02-Dec-2021. [Online]. Available: https://medium.com/initial-state/how-to-build-a-raspberry-pi-temperature-monitor-8c2f70acaea9. [Accessed: 13-Feb-2023].

"**IOT based Green House Monitoring System using Raspberry Pi**," *YouTube*, 18-Nov-2018. [Online]. Available: https://www.youtube.com/watch?v=RYaguyzxSIA. [Accessed: 13-Feb-2023].

Jenfoxbot and Instructables, "**Raspberry pi irrigation controller**," *Instructables*, 13-Oct-2017. [Online]. Available: https://www.instructables.com/Raspberry-Pi-Irrigation-Controller/. [Accessed: 13-Feb-2023].

N. Rawlinson, "**Build a greenhouse monitor with a Raspberry Pi**," *IT PRO*, 04-Dec-2020. [Online]. Available: https://www.itpro.com/network-internet/internet-of-things-iot/357985/build-a-greenhouse-monitor-with-a-raspberry-pi. [Accessed: 13-Feb-2023].

Peter Czanik March 9, P. Czanik, M. 9, |, 1 Comment, Steve Ovens (Alumni, Stephan Avenwedde (Correspondent), and Peter Czanik Peter is an engineer working as open source evangelist at Balabit (a One Identity business), "Collect sensor data with your raspberry pi and open source tools," *Opensource.com*. [Online]. Available: https://opensource.com/article/21/3/sensor-data-raspberry-pi. [Accessed: 13-Feb-2023].

R. Barnes and R. runs R. P. Press, "**Hydroponic gardening with a Raspberry Pi**," *The MagPi magazine*. [Online]. Available: https://magpi.raspberrypi.com/articles/hydroponic-gardening. [Accessed: 13-Feb-2023].