Green Thumbs Project Report

A Project for Plant Ownership Automation

This report describes an innovative project showcasing the use of technology for sustainable automated plant care

YUANCHENG, CAO

Halicioğlu Data Science Institute, University of California, San Diego, yuc094@ucsd.edu

SHANE, BENETZ

Electrical and Computer Engineering, University of California, San Diego, sbenetz@ucsd.edu

JACKSON, N, WHEELER

Computer Science and Engineering, University of California, San Diego, j1wheele@ucsd.edu

1 INTRODUCTION

In a world where efficient time management is one of the most difficult tasks to succeed at, not many people have the ability to slow down and do an everyday life activity– plant care. As urbanization accelerates and lifestyles become increasingly fast-paced, the challenge of maintaining greenery within our living spaces grows more complex while its necessity remains constant. In response to this, we present our innovative solution: an automatic plant watering system. Imagine a world where the vibrancy of greenery effortlessly coexists with our fast-paced lifestyles, where the act of nurturing plants is not a time-consuming chore but a seamless, automated experience. This is the future we envision – a future where the intersection of technology and nature not only eases our daily lives but also contributes to a more healthy and appealing living environment.

This report takes you through the journey of conceptualization, development, testing, and collaborative efforts that culminated in the creation of our automatic plant waterer. The Green Thumbs automatic plant watering system is a solution that provides users with an all-in-one plant care system where they can check the health of their plants constantly, automate the watering of their plants without having to be plant experts, and remotely manage their plants from anywhere. The motivation behind our project stems from the recognition of the need for a reliable and automated solution to ensure the well-being of plants, not just as decorative elements but also as essential contributors to a healthier indoor environment. The subsequent sections of this report will delve into the motivation behind our project, exploring the driving factors that led us to embark on this endeavor. We will then proceed to system development, illuminating the technological and design considerations that shaped the creation of our automatic plant waterer. Following this, the testing phase will be detailed, shedding light on the iterative evaluation process that ensures the system's effectiveness and user-friendliness.

Collaboration is a crucial aspect of innovation, and in this report, we will spotlight the collaborative efforts that brought diverse skills and perspectives together to shape our project. Finally, we will conclude by discussing our findings, and potential improvements, and offer insights into the broader implications of our automatic plant watering system.

Join us on this exploration of merging technology with nature, as we strive to make sustainable plant care an effortless reality regardless of how much of a green thumb someone has.

2 MOTIVATION AND BACKGROUND

The driving force behind the Green Thumbs automatic plant watering system is rooted in a comprehensive and lived understanding of the challenges faced by individuals in maintaining their indoor greenery. In the face of increasingly busy lifestyles, people often find it difficult to allocate time for plant care, resulting in wilting plants and decreased happiness when a living thing dies under one's care. Recognizing this human-centric challenge, our motivation was to develop a solution that seamlessly integrates with modern living, making plant maintenance more accessible and less burdensome.

On the technical front, the motivation stemmed from the desire to create a smart and adaptive system capable of addressing the intricacies of plant care. The challenge lies in developing a technology that not only automates the watering process but does so with a level of sophistication that accommodates diverse plant species and responds dynamically to environmental conditions. This technical ambition reflects our commitment to crafting a system that not only eases the burden on individuals but also contributes to the overall well-being of indoor plants, fostering a harmonious relationship between technology, humans, and nature. In essence, our project seeks to marry user convenience with technological ingenuity to redefine the experience of plant ownership in the contemporary world.

Indoor plants are recognized for their multifaceted contributions to occupant health, particularly in the context of enclosed environments. The physiological and psychological advantages associated with incorporating greenery into indoor spaces are substantial, aligning with both environmental and human-centric considerations.

The benefits of keeping plants inside one’s house have been widely accepted as beneficial. Firstly, indoor plants act as natural filters, playing a pivotal role in the purification of indoor air. Noteworthy studies, such as NASA's Clean Air Study [1], have meticulously identified specific plant species capable of mitigating prevalent pollutants. Even though modern studies have not concluded exactly how much impact they have in larger indoor spaces [2], they can’t do anything but help regarding oxygen generation. This air purification capacity not only aligns with sustainable urban living but also directly impacts the health and well-being of occupants. An additional benefit is the influence on indoor humidity levels. Through transpiration, plants release water vapor, thereby contributing to the regulation of indoor humidity. This regulation allows for maintaining optimal humidity levels and is not only helpful in fostering respiratory health but also important for the comfort of indoor occupants.

Another interesting facet is the psychological impact of indoor plants which is something that engineering students and the general population alike who spend most of their time indoors must acknowledge. Numerous studies point to the stress-reducing attributes of indoor greenery, establishing a correlation between human well-being and exposure to natural elements within built environments. Additionally, the cognitive benefits, including enhanced concentration and productivity, demonstrate the relevance of integrating plants into indoor spaces. The incorporation of greenery into spaces that would be otherwise completely manufactured adds significantly to the experience one has in the space, which encourages us to focus on human-centric design as much as possible for this project.

There are quite a few systems and strategies out there that try to accomplish the task of keeping plants alive. We have selected a small group of methods and products that we feel represent the current selection of solutions on the market for users.

Hand watering is the age-old method of watering plants. This is pretty self-explanatory as a plant owner would take a container of choice, fill it with water, and walk around to each plant to water it based on when they think it needs watering. This method has been around for as long as indoor plants have existed. The benefit is that it is essentially free and allows the user full control over what happens to their plants. On the flip side, everything that happens to the plant is the responsibility of the owner. Based on many interactions with people and even personal experience, this gives too much responsibility to the user and forces active participation in keeping the plant alive, requiring consistent effort and knowledge of each plant type to know when and how much to water them. This often is foiled by forgetfulness, traveling plans, and lack of knowledge.

The next solution that remained popular for a while in recent times was watering globes. These devices, usually made simply of glass, aim to be a refillable automatic watering system that you can simply stick into the soil of the plant. Their design is that there is a larger globe that you fill with water that is attached to a spike that you flip upside down and stick in the plant. Basically, the water only comes out of the tip of the spike when the soil is dry enough that air can enter the tube and change the pressure inside the globe full of water so that more water comes out. One benefit of this is that it provides an easy way to consistently water plants automatically, relieving the responsibility of constantly having to hand water the plant. It allows the user to not worry about watering until they recognize that the globe is empty which removes the need for constant checkups on the plant. The issue with this strategy is that the user cannot adjust the watering style based on the plant in which it is used. That has limited users' to using these with plants that can take constant watering. Although this still encompasses a large selection of plants, many users have reported that their plants have been over-watered while using it and for that reason, many people only use them on very specific plants.

The final group of devices that exist on the market are automated irrigation systems. These systems are usually a mixture of some kind of water pump run on a timer that is connected to a tubing system routed to a group of plants. This interconnected system usually involves some setup of items such as a water reservoir, automated pump, and tubing to each plant. One popular version of this system is the RainPoint [5] as seen on Amazon with which you can automate watering over WiFi through their app. This is probably the most controllable watering system that exists which allows a significant store of water to be used over time. It gives the user the ability to water their plants manually or automatically from anywhere. The system assumes that all plants in it should be watered the same as it cannot control individual watering by the plant. It also requires connecting all the plants in a chain and adding a water reservoir at a certain height/placement from the pump. This still requires the user to know how much to water each plant, as well as to monitor the plants' condition in relation to their watering.

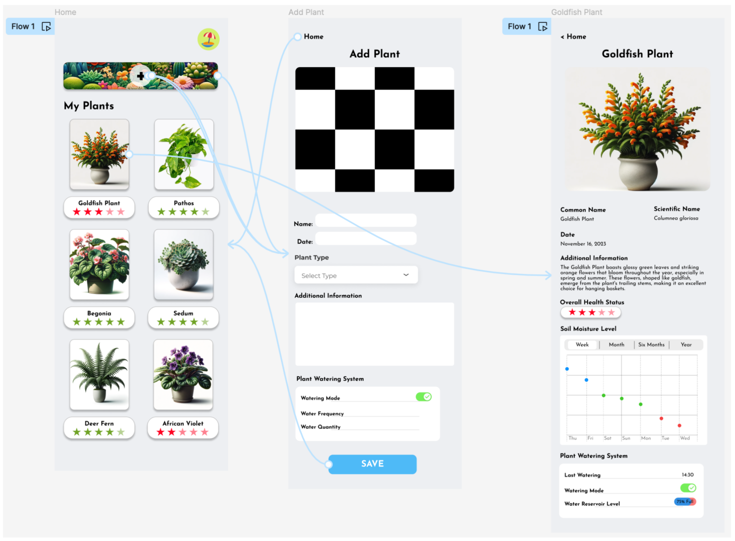
Our solution aims to combine the best of all of the existing methods while adding an even smarter aspect to it. We aim to provide a system that can store water and dispense it into the plant, be controlled remotely, that will take care of watering for the user if desired, and that can give the user as much information about the plant as possible without needing to be physically present for observation.

3 Design

The solution that we propose is a three-part system that provides an interface for the plant, an interface for the user, and a connection between them, our server. We accomplished each goal that we put forth in the way that we thought would be most user-friendly.

Firstly, we wanted to provide an interface where the user can easily and enjoyably interact with their collection of plants. This was designed around the idea that most people like to collect different plants of different species around their spaces, so each one must be cared for uniquely. Since plants are living organisms, we decided to give each of them a profile so the user could check in individually and get their care history. It is kind of like a health report where each plant can be given a name and the user will know exactly how it is doing.

Our mobile application's design journey began with a focus on accessibility, simplicity, and aesthetics, and underwent significant evolution through various stages of prototyping and user feedback. The initial prototype, as shown in Figure 1, centered around a user-friendly "Home" page. This page featured a prominent call-to-action button leading to the "Add Plant" page and displayed existing plants in an aesthetically pleasing card format, each card illustrating the plant's image, name, and overall health status.



Common Name
Goldfish Plant

Scientific Name
Columnea gloriosa

Date
November 16, 2023

Additional Information
The Goldfish Plant boasts glossy green leaves and striking orange flowers that bloom throughout the year, especially in spring and summer. These flowers, shaped like goldfish, emerge from the plant's trailing stems, making it an excellent choice for hanging baskets.

Last Watering 14:30
Watering Mode
Water Reservoir Level

Figure 1: Prototype of Mobile App

The first draft of the “Add Plant” page was quite detailed. Users could upload an image or provide a URL, enter the plant’s name, select the acquisition date, choose the plant type, and add additional information. The highlight of this page was the Plant Watering System. It offered a choice between automatic and manual watering modes, with ‘On’ indicating the system’s automatic watering based on sensor data and algorithms, and ‘Off’ for manual watering, where users controlled the frequency and amount of water.

Clicking on a plant’s image on the “Home” page led to its individual profile. This included detailed information such as the plant’s name, acquisition date, additional details, overall health status, and a visualization of the soil moisture level. Users could view data over different time periods – a week, month, six months, year. The moisture levels were color-coded (blue for wet, red for dry, and green for optimal moisture), making it easy to understand. The plant profile page also enabled the user to edit their watering preferences and displayed the last watering date.

As the design moved towards its final form, several significant changes were made based on the data available from Bluetooth and our database. We introduced login and signup pages to enhance user security, removing the requirement for users to upload or link plant images. Plant cards in the final design displayed the plant’s name, size, and type, focusing on the essentials. Users could now delve into each plant’s health status by examining soil moisture levels, sunlight levels, and watering history. The visualization of soil moisture and sunlight levels was simplified to display average over 30-minute intervals for the past two hours, with a new swipe functionality to view past or future data. Due to time constraints, we removed the plant watering system feature. The “Add Plant” page required users to initialize the plant watering device via Bluetooth, with detailed setup instructions provided. This shift towards Bluetooth integration marked a significant step in enhancing the application’s usability and interactivity. The final design of our mobile application reflects our commitment to creating a user-friendly interface that balances functional needs with aesthetic appeal. It stands as a testament to our iterative design process, highlighting our adaptability and responsiveness to technical constraints and user feedback.

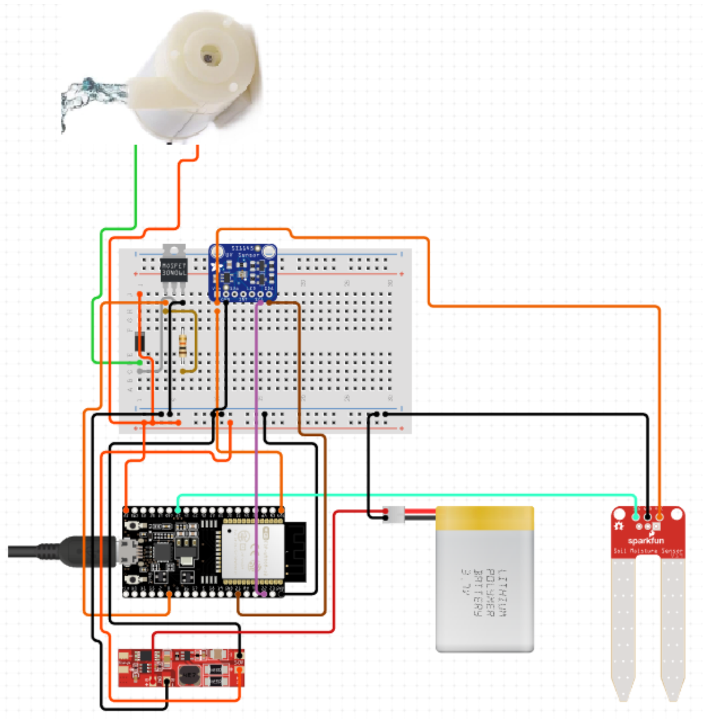


Figure 2: Prototype of Hardware for Plant Device

The design for the plant monitoring and watering device was a pretty straightforward idea, although we started with many different ideas for what we could do with it. We wanted something compact that could store water which would be able to deposit water into the plant by itself. This was a combination of the ideas of watering globes and irrigation systems. This would solve the problem of not having individual watering plans for each plant and of having to connect a network of plants together. Our true goal for this part of the project was to create something that the user could pull out of the box and stick straight into the plant with no more setup required than filling it with water and telling it what kind of plant that it is in. We chose a small microcontroller that would run on battery and that was power efficient so that this device could last for long periods of time.

The way we came up with this design was thinking what do plants need? Water and sunlight. So the two sensors we focused on were a soil moisture sensor and a sunlight sensor. We wanted to be able to tell the user if the plant was getting what it needed. One of the first prototypes we drew up for the design is pictured in Figure 2 above.

This design developed a fair amount based on having to troubleshoot some components. It took a while to research the right sensors, so the ones pictured above are different from what we ended up with. Additionally, we had to end up powering the water pump motor differently than what we had hoped was the easy design. Eventually, we brought in a separate battery pack and a relay switch to power and control the water pump. We also wanted a solution that didn't involve a breadboard since we wanted the device to be compact and be able to fit closely with whatever we used for our water reservoir.

The water reservoir probably took the longest to decide on because it left the most room for design options. Since none of us had much experience in 3D modeling, we decided not to spend much time on it and just chose to use a bottle with a straw coming out of it. Many ideas could only be custom-printed to neatly house all our components alongside the water reservoir. We also thought about getting a water level sensor so that we would be able to tell the user when to refill the water, but we decided this was nice to have and not a key part of the design.

One of the best parts of our design was that we chose a low-power microcontroller that could be battery-powered and last for a long time. We didn't want to use the regular Raspberry Pi because they don't last very long on battery life and we don't want to force a user into keeping their plants plugged in. They also have too many extra components that are unnecessary, so we chose to go with a much smaller device. The longer the user doesn't have to interact with the device, the better. That's why after setting it up, all the user needs to do is check to refill the water. The rest is done for them through the server.

We added the last and most important piece of the design, the server, as a way for users to manage our system from anywhere. This is how we solved the problem of not being able to travel and take care of plants at the same time. The ultimate goal for this part of the system was for it to be like cloud computing and storage. This would allow the user to be able to keep track of any amount of plants and have everything be personalized to them. On top of being just remote access to the system, we wanted our server to be the brains of the operation. The server is the easiest thing to update and therefore in the future, we can update the watering algorithm much easier than if it is baked into each device.

The most novel part of our design resides in this server as well. Most users don't know how to care for specific plants without doing research on how to water each new plant they get. That is why when you set up a device in our system, you select the plant type, and from there, our server should be able to know exactly what type of watering and sunlight should be applied to the plant. We achieve this by accessing a database of plant types and their corresponding watering styles. Initially, we considered integrating pre-existing databases for this purpose but eventually decided that it was beyond the scope of our project. Our watering algorithm would know exactly how to water each plant based on its current status, whether that be a time-based approach or a consistent watering approach. The server would also be able to report every piece of data to our user through the app and give the most accurate reassurance to the user about the status of the plant.

4 System Development

4.1 Architecture

Let us now explain our system architecture. Reference the diagram below for a general overview of our system architecture.

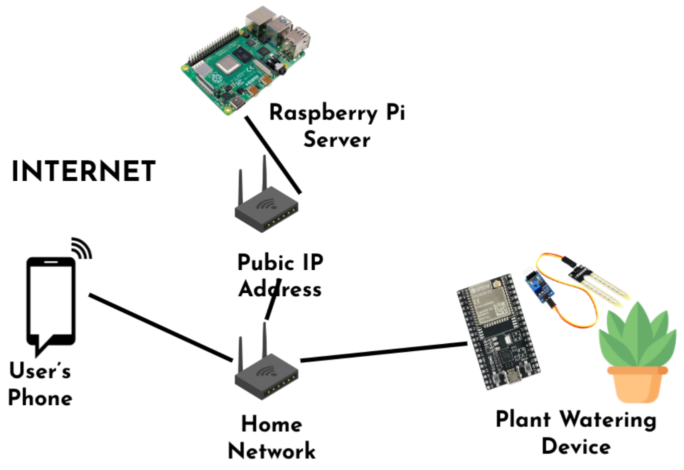


Figure 3: Depiction of our System Architecture

From the user's perspective, they would first buy our plant–watering device online. Then, they would download our mobile phone application. Thus the item labeled "User's Phone" and the item labeled "Plant Watering Device" in the graphic above are both interacted with by the user. Then the user will place the plant watering device in the plant they desire to water.

Now, let us discuss the networking and the input/output of this system.The Raspberry Pi that we have acted as the server. The User's Phone and the Plant Watering device are the clients in our intricately developed system architecture.

4.1.1 Raspberry Pi Server

Our Raspberry Pi server is accessible via a public IP address. For the project, this Raspberry Pi server was located in Jackson's house. The Ngrok service was then implemented to make this Raspberry Pi's API server publicly accessible. With the Ngrok service, we were provided a Ngrok public IP address, and all traffic going to that public IP address was routed to our Raspberry Pi device, sitting in Jackson's home. This allowed us to run an API server on the Raspberry Pi and make that available publicly. Thus, during development, our team members were able to access our API server to test the networking code they implemented.

For our product, it also makes the user's experience much better to have this publicly accessible server. In this way, the user does not also have to buy a Raspberry Pi for their household to work as a server in our plant watering architecture. Rather, and much better, they can use our publicly accessible server. For this project, the server was implemented on a Raspberry Pi that used Ngrok to make it publicly accessible, however, in the future this server code could easily be moved to a cloud server.

What did we have running on our Raspberry Pi? We had an HTTP server running on our Raspberry Pi. This API server handled almost all the communications in our project. This API server was responsible for communication between our plant watering device and the server and between the phone and the server. What did we use to implement this API server? This API server was implemented in Python using the Uvicorn and FastAPI libraries. The benefits of these libraries were that they made creating an API server very quick and easy, and even automatically developed the documentation for this API server. This documentation can be accessed by requesting the "/redoc" endpoint of our server.

4.1.2 Device Setup and Bluetooth Communication

Next, let us talk about the setup of our system, which will give us a better picture of the system architecture and the networking aspect of our system as a whole. When the user buys our device, they first fill it with water and then place it in the plant they want the water. The next step is then for the user to download our app, and create an account.

After creating an account, they can initialize the plant watering device via Bluetooth through our app. On our app, they can go to the add plant button, where instructions will be given to them on how to initialize the plant watering device over Bluetooth. The most important thing is that the user sends their home network credentials to the device so that the device can get onto their home network. Once the plant watering device is on their home network, it can make HTTP requests to our publicly accessible server. After the WiFi is set up, they then fill out the plant's name, size, and type which all gets sent to the device over Bluetooth so that the device can register itself to the server with the user's ID. From then on all communication that the plant watering device does will be directly to the server. Thus, you can see that the initial setup of our device needs Bluetooth communication in order to work.

For using Bluetooth on the phone side, we were not able to get the functionality embedded into our own app, so we had to use a 3rd-party app called LightBlue. This app allows you to connect to any BLE device and read and write values to and from it. This is a pretty low abstraction application so the only way to send messages to the device was first converting them to hexadecimal and then pasting them into the LightBlue app to be sent. Because of this, the setup pages on our app allow you to input all the information, and then it outputs the hexadecimal string that you can copy and paste into the LightBlue app. There is one step for pasting in the WiFi credentials and another step to paste in the user-selected device properties. Although it would have been nice to have Bluetooth integrated into our own app, this was still a very effective solution and saved a lot of time on development.

4.1.3 Device and Server Communication

Now, let us get into more of the details of how the networking works once the device is set up on the user's home network. There are two possible HTTP POST requests that the plant watering device can make to the server. These are "/device/initialization" and "/device/check-in". Once the user initializes the device over Bluetooth, then the device makes a device initialization HTTP request to the server.

The device initialization request includes an ID for the user, a name for the plant, a size for the plant, and a type for the plant. This is all the information that is needed to initialize a device in the server's database. Once this device is initialized in the server's database, the server replies to that device with its unique device ID. Then using this ID, that device can call the HTTP check-in request.

How our system works is that our plant watering device will go to sleep for a given amount of sleepTime, and then wake up and make the check-in HTTP request. This check-in request reports the device's sensor data to the server. This sensor data includes the device ID, soil moisture data, sunlight sensor data, and the device battery level. All of this data is then recorded and saved by the server. The server then replies to the device with a command to water (if it is necessary), and with the amount of time for the device to sleep. This allows all control to be handled from the server side and allows power saving to be implemented on the device side so that it can run on battery power and not have to be plugged in.

4.1.4 User's Phone and Server Communication

Then, from our mobile application, there are four different API endpoints that are accessible. "/user/new-account" and "/user/login" are used to handle user account functionality. "/user/{user_id}/plants" gets all the plants for a given user. "/plants/{deviceId}" gets the detailed device sensor information for a given plant.

What does this look like in practice? In practice, the new account endpoint will be requested whenever the user wants to make a new account. This API call will create a new account with a new username and password in the database and return a unique user ID. The login endpoint is similar, however, it takes a username and password and evaluates it against the database to return the user ID that corresponds to that username and password, if applicable. These two API endpoints are used to handle all the functionality around user accounts.

The get plants API endpoint is a way for our application to get all the plants that correspond to a given user. This API endpoint takes a user ID. Using this user ID it will return the names and the device IDs corresponding to each of that user's plants. This API endpoint ensures that there is no conflict between different users and the plants that they receive, for each plant/device corresponds to a single user. From this API request, our application can then construct a list of plants that belong to the user. Then the user can select one of those plants in order to get more information about them.

When the user selects one of their plants to get more information about them, then the plant info API endpoint is called: “/plants/{deviceId}”. Our application will use the device ID corresponding to the plant that the user clicked to make this API request. This API request will then return all the sensor data from the device that is attached to the plant. Our app can then display that data to inform the user on how their plant is doing. This data includes the sensor data, sunlight data, the device battery level, and a list of timestamps for when the plant was watered.

4.1.5 Privacy and Security

How are information privacy and security handled in our project? The user's home network SSID and password are one thing that our device needs to function. The device receives this information over Bluetooth and uses it to connect to the WiFi. In order to ensure privacy and security, our device does not share this information with our server, rather it is kept private on the device.

At the moment, our system uses HTTP to communicate. This is an unencrypted protocol. In the future, we would like to upgrade to HTTPS, which would ensure secure communications over our system. In addition, we could implement the encryption of user passwords to improve privacy and security.

Another issue of privacy and security is our publicly facing API server. In order to prevent SQL injections, for all accesses to the SQL database we cleansed all user input before any SQL commands we executed. This protects against SQL injection attacks.

Another security measure we could implement on our server-side would be to block any IP addresses from accessing our server if we ended up getting too many requests from them in a short amount of time. This could hopefully help prevent server overload from DDOS attacks.

4.2 Technology Used

4.2.1 Server

On the server side we used a Raspberry Pi with a 16 GB SSD. On the server, we created a Python script that would run in the background, which would host our server on one of the ports of our Raspberry Pi. In order to do this we implemented the Uvicorn and FastAPI libraries into our project. Uvicorn made it possible to run a service on a port on our computer, and FastAPI allowed us to easily create an HTTP API server for our application.

For the development side of things, some care was taken to set up the ability to develop the server and update the code on the server. Port forwarding was set up on the home network of Jackson's house for remote SSH connections to be made to the Raspberry Pi, which was located in Jackson's house. Then, by enabling SSH on the Raspberry Pi and setting it up so that it used key authentication instead of password authentication, Jackson was able to remotely access the Raspberry Pi via SSH. This was an important technology in the development of our project.

As mentioned earlier, Ngrok was utilized to allow our API server to be accessed publicly. and Ngrok was set up so that it ran during the startup of the Raspberry Pi. This simplified the configuration of the Raspberry Pi, and if it ever restarted, it made it easy to get the Raspberry Pi back up and working. On Ngrok we had one endpoint which was accessible over both HTTP and HTTPS protocol. The HTTP endpoint was needed for the mobile application and the plant watering device to easily access and communicate with the server. On the other hand, the HTTPS endpoint was helpful for viewing the API server’s documentation from a web browser, which automatically upgraded to HTTPS upon connection to the endpoint. Thus Ngrok was a very important technology in our project.

One helpful thing about Ngrok is that it allows the user to have an unchanging domain name for free. This allowed us to have a consistent domain name as an endpoint for our mobile application and plant watering device to access in order to communicate with our server. If the name had to change every time the Ngrok server was started up, then it would be difficult to make those changes in the source code of our device and mobile application every time that new Ngrok server was started up. Thus having a consistent domain name was a contributing factor to Our success.

Now, let us talk about our database. The database on our server was crucial in our project. It stored all the data related to users and their accounts and devices and their sensor information. We needed a convenient way to store all this information on our Raspberry Pi. How did we do this? We implemented SQLite to store our information succinctly and efficiently. Using the SQLite3 library in Python, we were able to easily communicate with our SQL database through our server's Python code.

How did we structure our database? The schema of a database is incredibly important when thinking about long-lasting support and efficient data storage. We created a table for users which held the ID of a user, their username, and their password. This allowed us to have a single table where all information about users was stored, and with the user ID, we were able to have a unique user ID that could be used in other tables in order to link rows in that table to that user.

We also created a devices table that stores a device ID, a user ID, the plant name, a plant type, and a plant size. This table lists all the available devices and connects those devices to a specific user via the user ID. It also gives basic information about the plant that the device is connected to. When a device is initialized it is added to this devices table.

We then had two tables for logging information that devices report to the server. These tables are the device logs table and the water logs table. The device logs table stores the device ID that is reporting a specific log, the time stamp of that log, the soil moisture data, the sunlight data, and the device battery at the time of that log. This table is used to log the data from all device check-ins. This is the table that is referenced when the user desires to get more information about their plant. The water logs table is simply a table including the device ID and timestamp. This table logs a history of the timestamps at which a given device watered its corresponding plant. This table is accessed to get a history of when a particular plant has been watered over time. Both of these tables utilize automatically incrementing log IDs in order to differentiate different logs. In the case of both tables, the device ID is used to indicate which device the log corresponds to. In this way it's very simple to get all the logs for a particular device: simply use a SQL statement to return all the rows in which the device ID is equal to the device ID you are looking for.

Our last table was a plant-type table. This table was used to store constant information about different plants and their watering needs. This table can be expanded in the future to implement a plant watering algorithm for when to water any given plant based on the sensor data reported by the device attached to it. The plant-type tables store the soil moisture information that was used by our simple algorithm which decides when to water a plant based on its soil moisture.

4.2.2 Plant Watering Device

4.2.2.1 Microcontroller

The backbone of our physical device setup used for monitoring and watering the plant was a Loliin D32 ESP32 microcontroller board. While only running on 3.3V, this device was the powerhouse of our system and controlled everything including driving the soil and sunlight sensors, doing the calculations on their readings, Bluetooth communication using Bluetooth Low Energy (BLE), HTTP communication over WiFi, and controlling logic for the water pump to water the plant. This little board which is about 1" by 2" in size was able to do all of this only running on 4MB of flash memory and 2 cores. The libraries available on the ESP32 platform are pretty extensive, so even though the WiFi and Bluetooth use the same onboard radio, there are automatic switches that allow us to use them both at the same time. One benefit of this board was that manufacturing/procurement-wise, each board only cost about $7. The board includes around 30 GPIO pins including pins capable of analog read and write as well as digital to analog conversion. Additionally, the device has a JST battery port and modulator so that it can run on battery quite easily. This board is one of the best for low-power applications as it consumes very little while also being able to go into "deep sleep" where it takes less than 0.2 mA. In theory, with our 1800mAh battery, this device could last up to months on one charge.

4.2.2.2 Sensors

We mostly just used two sensors in our system: a capacitive soil sensor and a BH1750 light intensity sensor.

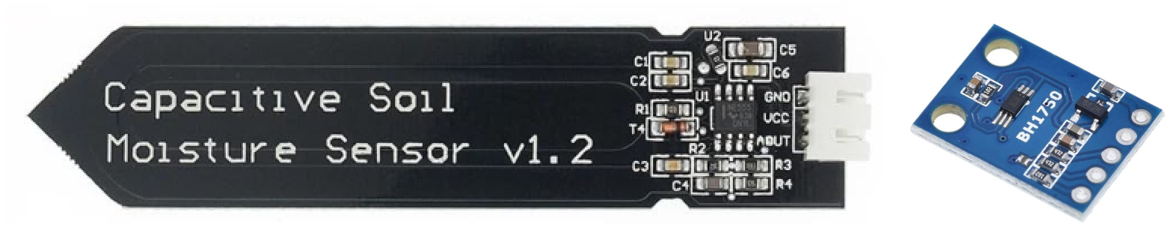


Figure 4: Soil Moisture sensor (left), Sunlight sensor (right)

The capacitive moisture sensor has one U-shaped plate on the outside edge of the device and another plate inside. It measures the capacitance between those two plates which changes by the conductivity of the substance surrounding the sensor (i.e. when there's more water around the sensor which is more conductive than air, the capacitance measurement will change). This value is output to our microcontroller at an analog voltage which is then scaled to a value between 0-100%. We found that in dry soil the sensor would read about 2.7 mV and in water it would measure closer to 1.4 mV. This range allowed us to read soil moisture as a percentage which could be reported to the server. On top of that sensor, we used the BH1750 which is similar to sensors used in phones to control the brightness of the screen based on the ambient lighting. This sensor uses a photodetector to measure the light intensity, and all the computation to turn that into a value is done on the sensor and is provided to the microcontroller using I2C protocol over serial ports. This value ranged from 0 in the dark to around 65000 with a flashlight pointed at it. We reported these values directly to the server without interpretation.

4.2.2.3 **The Actuator**

The way that we provided water to the plant was through pumping water out of a reservoir (a plastic bottle) and down a straw that was inserted into the plant. The submersible water pump that we used was essentially a pretty basic 3V-5V DC motor that we turned on using a relay. Since our microcontroller is not a high-powered device, we couldn't directly power the motor from the GPIO pins without drawing too much current and causing faults with the rest of the board (we tried this and it started smoking). The solution we found was to use a 3.3V relay module which was powered by a battery pack (2AA batteries) and only use the GPIO output of our board as a logic signal to turn it off and on. When the relay was on, it would connect the water pump to 4.5V using 3 AA batteries which gave it enough power to move water pretty quickly. To control watering, we would use the percentage of soil moisture to set the time for how long to pump water for. A higher percentage means water for shorter periods (~0.5 seconds at a time) and a lower percentage would mean water for longer (~3 seconds at a time). This would be important for different water strategies as for some we wanted a constant water level and for others, we wanted to wait until it was dry to then fully water it back to 100%.

4.2.3 **Mobile Application**

In our mobile app's development, we chose a combination of JavaScript and React Native, a framework that enables the creation of cross-platform apps with a single codebase. This approach was particularly advantageous given our team's diverse computing environment, which consists of both Mac and Windows systems. Additionally, our decision to use Expo Go significantly streamlined our development process. Expo Go facilitates the testing of our React Native application on iOS devices, which is crucial given that all team members possess iPhones. With Expo Go, we could instantly view changes on actual devices, enhancing our ability to rapidly prototype and refine features. The combination of React Native and Expo Go was instrumental in accommodating the varied technological environments within our team while ensuring a consistent development experience.

4.3 Features

Let us dive into the different features of the system!

4.3.1 Automatic Plant Watering Feature

First of all the primary and essential feature of our system is the automatic plant watering feature. This automatic plant watering feature is not a simple timer. Rather, our automatic plant watering device smartly waters your plant at the proper time according to the type of plant you have, the size of your plant, the sunlight in the area, and the soil moisture of the plant. This describes our ideal algorithm, which would be an AI algorithm, for our plant watering system. This automatic watering feature is the central feature of our system because it allows our users to plug in our device and then not worry about the health and longevity of their plants.

How is this feature implemented? Let us dive into this matter. First of all, we have an intricately developed and designed plant watering device. This device consists of a microcontroller, a soil moisture sensor, a sunlight sensor, and a water pump for adding water to the plant. All of this runs on a simple battery. Thus the user does not need to plug our device into the wall, but rather it can last for a very long time on battery power and water your plant as it does so.

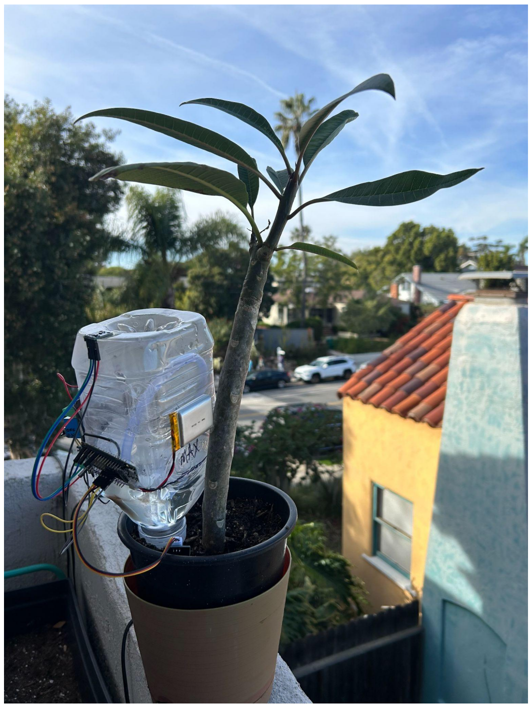


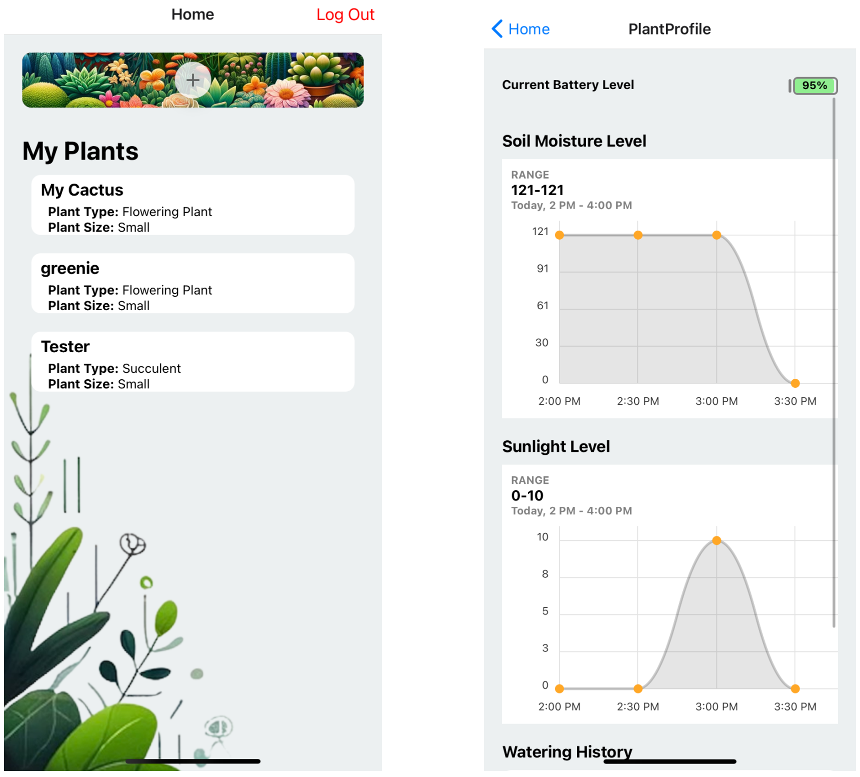
Figure 5: Depiction of our Plant Watering Device Plugged Into a Plant

How does our device survive for such a long time on battery power? The flow of the software is to first turn on the Bluetooth server on the device which allows us to dynamically send WiFi login credentials to the board. Once the WiFi is set up and the user has decided on the parameters for the plant, the device will register itself over an HTTP POST to our server. After device setup, all that the board needs to do is take an average of the sensor values for around 10 seconds, post them to the server, and check if it needs to water. If it needs to water, it would pump water for a short amount of time then pause for a designated period, then check soil moisture again, and keep looping until it sensed that the goal had been reached. If the goal was reached or it didn't need water, it would go to sleep for a designated amount of time (somewhere around 30 minutes to an hour) and then wake up and repeat the sensor check-in, water, and sleep pattern. By being asleep for most of its lifetime and only waking when necessary, our device can optimize its power consumption and make itself a reliable and practical product for our users to interact with.

How do we decide when to water the plant? Our server is the powerhouse for the algorithm deciding when and how much to water each plant. Our server takes in all the data that the device reports to it and puts that data through our algorithm to decide whether or not to water that plant. This complete algorithm is something that is still in the works for our project. In theory, there can be styles of watering such as constant solid moisture or only watering once the soil is completely dry. The amount of water to pump also depends on the size of the plant. Our minimum viable product for the algorithm is to set the moisture goal to around 80% and have a threshold that the plant can be within. This threshold size was based on the plant size since the larger plant may need a larger threshold since the soil sensor is only in one small part of the plant.

4.3.2 User Access to View Plant Status

Imagine going on a trip not knowing how your beloved plants are doing. There should be some way to see the status of your plants and to see if they are being watered. With our system, this dream is made a reality. We have implemented a mobile application through which users can view how any one of their plants are doing. This is crucial to user experience.



My Cactus
Plant Type: Flowering Plant
Plant Size: Small

greenie
Plant Type: Flowering Plant
Plant Size: Small

Tester
Plant Type: Succulent
Plant Size: Small

Figure 6: A depiction of the user interface of our mobile application, which allows users to view all of the plants that have devices connected to them and to view how those plants are doing.

As you can see from Figure 4, our mobile application has a page where the user can view all the plants that they have devices connected to. Then, by clicking one of these plants, they can view a graph of the soil moisture of their plant over time, a graph of the sunlight their plant is receiving over time, a history of when that plant was watered, and the battery health of the plant watering device attached to that plant. This provides crucial information to the user with regard to the status and the health of their plants.

How do we provide this information? All of the sensor data that our plant watering device collects is sent to the server. Because all of this data is stored on the server, this data can be retrieved by the user when they log into their account on our mobile application. Through simple API requests, as detailed in the system architecture section, this feature is made a reality.

5 Testing and Evaluation

5.1 API Testing

One of the crucial aspects of testing that we performed was testing that all of the different API requests were working correctly. In other words, testing that communication between the different devices in our system was working correctly. This is something fundamentally important for the system overall, and it is something that connects the work of different teammates, so it was an important item to test.

One of the ways we performed tests on the APIs was through sending POST and GET requests from our computer to test and make sure that the API endpoints themselves were working as expected. Once this was good, then we would test from the respective devices to see whether the API requests were working from our devices.

From our tests on the API endpoints and the communication between our different devices, everything seemed to be working very well. The communication and networking between our devices is one of the strengths of our system. This addressed our initial goals perfectly with regard to communication in our system.

5.2 Watering Functionality Testing

Another test we did was of the device's watering functionality. We tested to see how our device waters the plant. At first, we tried to make the microcontroller directly activate the water pump. However, the microcontroller was not strong enough for that. Thus, after implementing a relay, we had a functioning water pump.

We then were able to test the watering capability of our system by forcing water from the server side, telling the microcontroller to water to a specific goal moisture level. The microcontroller would then do multiple tries to try to reach that soil moisture level from its soil moisture sensor by watering it.

This functionality was a little more difficult to implement than we had anticipated. For one, sometimes the water got stuck in the straw since it would not drain down into the plant fast enough when our device attempted to water the plant. Another thing was that sometimes the soil moisture sensor would not be updated immediately after the watering because it takes some time for the water to propagate through the soil. Overall the watering solution worked pretty well. However to bring our project to the next level we would want to improve this watering system and solution.

5.3 Watering Algorithm Testing

Our algorithm for deciding when to water is based on the soil moisture level of the given plant attached to our plant watering device. This means that if the soil moisture for a particular plant type is below the threshold defined in our database, then our server tells our device to water that plant until it reaches the goal soil moisture level, which is also a value in our database. One problem though is figuring out what these values of soil moisture threshold and soil moisture goal should be for any given plan type. Thus, we needed to do some testing to figure out these values.

In addition to this, we also were testing to see whether the soil moisture sensor would get accurate readings. One thing we found when testing was that the soil moisture sensor would sometimes report percentages greater than 100. Thus, we had to adjust the soil moisture sensor. Also, depending on where the soil moisture sensor was placed and how deep it was placed the soil moisture value may be different. The compactness of the soil around the sensor determined what kind of range of values it would have.

Taking into consideration these two matters, we ran a series of tests on the weekend before the demo. We placed our plant watering device in a plant, connected it to our server, and let it sit to see if it would function correctly and properly.

The results are that our system correctly watered when it was supposed to water our plant. We also, at the same time, made some adjustments to the soil moisture sensor so that it would report accurate values. After our device watered our plant, the soil moisture remained high for the next two days, as the soil was very well saturated with water (it was a little bit overwatered since we also tested the watering system out more than it would have usually been triggered).

Our system addressed our initial goals in this area well. Although the algorithm could be improved for the watering of the plant based on the plant type, plant size, etc… for which we would likely need to implement an AI algorithm to solve.

5.4 Mobile Application Testing

Another component of our testing involved testing our mobile application. This application is what displays the user's plants and the sensor data for specific plants. In order to test this we collected real-world data from our plant watering device which was placed in a plant over the weekend. Then we went on our app and viewed the device sensor data corresponding to that device, which had been collecting real-world data.

The results of our testing were that we were able to confirm that our mobile application was retrieving all the data that our device had sent to the server. We did notice that the displaying of the data, including the correct ranges for each data type, could be slightly improved. So, in the end, the mobile application went above our initial expectations for our project.

6 Collaboration

6.1 Structure of the Team

Since our team only consisted of 3 members and since there were basically 3 fronts to our system, we were able to divide up pretty easily to each take on the part that interested us the most. This ended up being a pretty equal amount of work for each of us and we were able to work alone on things which allowed faster development.

Since Yuancheng had a background focused more on data analytics and design, the app seemed like a perfect fit since this was part of our system that would be user-facing and needed to model data well while providing a user-friendly experience. Yuancheng headed this part of the project from the design phase to the actual implementation all the way to reliability testing. Shane provided some small input with the Bluetooth functionality of the app, but Yuancheng took that project head-on from choosing our development structure to making it all come to life. The first week of development was mostly designing the look and functionality of the app where Yuancheng successfully created a user interface that was appealing and had all the functionality we needed for the project. The next week was focused on getting the application development up and running, which proved difficult, but he was able to set it up and get a basic app running where we could enter plant information into a form. The next two weeks were when a lot of the backend and app flow came into the picture. Yuancheng first developed the ability to go from logging in to a home page to add plants. Once he and Jackson communicated about the API more, he was able to get a functional login process and the ability to get real data from the server and graph the data. During the end of development, Shane was able to add in the Bluetooth intermediate steps before all the final testing of the app was done in order to get the full flow of the app nailed down.

For the server side of the project, Jackson was able to take on just about all aspects of the process. With some background in servers and running remote setups, he was able to get our server up and running quickly. While we all had some input on what the API would look like, he did most of the design and implementation. He also did everything database-related to be able to store watering strategies, device properties, and user accounts. The first week he quickly got a basic local server working on the Raspberry Pi and we were able to interact with it and check API calls with a microcontroller making calls to it locally. After that Jackson spent some time getting the remote aspect of the server running where we could leave the Raspberry Pi running at his home while being able to ssh into it remotely and start/stop the server and make edits to it. This took some iterations and a lot of tinkering to get the networking set up correctly with Ngrok. While most of the API to interact with the server was developed early in the process, the logic behind to server gradually increased over the next couple of weeks starting with the database functionality of storing users and plants, and then the ability to read data from the plant waterer and store it correctly, and finally the logic that responded to the waterer with how much it should be watering. The process of developing the server's API was a constant effort to add functionality where we iteratively needed more information to and from the server.

Shane was able to take on the plant waterer hardware aspect of the project since he has a background in Electrical Engineering and had some experience with projects using microcontrollers programmed by Arduino. He was able to do all of the hardware selection and physical device testing to be able to get meaningful results from all the data we were collecting. The task of iteration on the hardware implementation first started with trying to get sensor readings on the device and discover their meaning. It took the first week to think through what libraries and connections needed to get set up so that we could see consistent readings when putting the soil moisture sensor in and out of the water and the light sensor in and out of the darkness. The next week involved setting up HTTP communications with the server and sorting out what values to send to the server and how to get good readings and averages. The week after that, effort was focused on setting up Bluetooth communication and parameterizing all the places where we had hard-coded values. After running into issues powering the water pump, it became a constant effort to try new setups for powering the motor and it eventually got figured out. Once that was figured out, the last week was spent iterating on how watering would happen and how to get goals from the server and make them happen. The final week was also spent setting up the low-power aspect of the device by attaching the battery and controlling the logic of when we wanted the device to wake up and go to sleep.

Overall, we mostly all stuck to our niche of the project while also providing suggestions to each other and discussing how communication between each part would happen. We required reviews from other members of the group when making pull requests on our GitHub project, so we all had to get partially familiar with each other's code and design.

6.2 Problems/Issues and How They Have Been Solved

As a group, we did not run into many problems regarding communication and division of labor. Everyone held up their part and since there wasn't too much overlap, we didn't hit too many road bumps. One issue that we ran into was when Shane and Yuancheng were both working on parts of the app, and we ended up running into merge conflicts on Git. We were able to get on a call with each other and resolve the conflicts so that neither of us lost work we had done. The other hurdle that we had to clear was the issue of app development since IOS development is most readily available on MacOS and Shane only had capabilities for work on Windows and Linux. This was solved by Yuancheng choosing the React Native framework that allowed development across platforms which helped not just isolate all the development and testing responsibility on him.

7 Conclusion and Future Work

7.1 Summary of the System

Our system as a whole provides a very convenient and practical way for users to keep their plants alive without having to worry about them. Our system will automatically water users' plants for them. Overall our major contribution was to develop the system and the architecture to make this automated plant watering dream a reality. We did the work to develop the user-facing mobile application, the plant watering device, and the code for the back-end server.

All of this can and should be further expanded upon. One feature of our project is that we made it very modular. This allows for future development to easily improve our product. With regards to the networking between the server device and phone, this same networking works irrespective of what technology is used on the ends. In other words, the server code could be placed on a server in the cloud and our system architecture would still function the same. Or the plant watering device could be upgraded and changed, our system would still work. The same goes for the mobile application. This allows for further development of our project.

7.2 Future Work

We could take this project much further if more time and resources were devoted to this effort. In fact, this system could become a commercializable and sellable product. If this were the case, the system could then be used in many people's homes in order to allow them to more easily and efficiently keep plants alive. This could be an extremely useful product for customers. I imagine that if this system were developed correctly, there would be a large interest from consumers. Added onto the fact that there is not a lot of competition in this area already, this could be a profitable business.

What changes to our system should be made in order to make this a reality? One of these changes would be an improved plant watering device. If an improved plant watering device with all the appropriate sensors could be made, and it was made in such a way so that it was easily manufacturable at a low cost, then it could make this business idea a reality. It would take time in order to develop a device that could easily do this and be easily manufacturable. In addition to this, the device could also be improved to include other sensors including a water level sensor, which would allow it to report to the user how the water level of their water reservoir is doing.

One important improvement to our system would be an AI algorithm for deciding when to water a given plant based on the device information that it receives. The more sensors we have on the device the better data the AI algorithm would have to work with to decide when and how to water a plant. We would love to be able to use a LLM or just find a really good database where we could ask just about any plant type and it would give us the correct watering and sunlight needs. We could then translate this into our watering algorithm so that we could have an even more diverse and correct strategy to keep plants healthy.This AI algorithm would be crucial to our system overall, as it would be what decides when to water a given plant. It would take much time to train and develop this algorithm for our system. However, this could greatly improve trust in our system as a whole.

Another change or improvement to our system would be on the users' end when they are deciding what type of plant their plant is. At the moment the user has to manually select what type of plant their plant is and report that to our system. Ideally, we would be able to implement AI image recognition for the user simply to take a picture of their plant and then our AI system would be able to decide what type of plant that is and report it to our server. This would make the user experience much better in this regard.

More improvement to our system would be an improved mobile application. At the moment our mobile application is basic, with the ability to view a list of the user's plants and to select those plants to get more information about each of them. It also has a way to add a plant and establish a Bluetooth connection to a new device in order to set it up to work on a plant. General improvements to the app in this way would make the user experience far better. Integrating Bluetooth would remove the need to copy and paste into the 3rd party app in order to set up a new device to connect it to your plant so that it shows up on our app. However, in the future, it would be beneficial to expedite this initialization process. This could be ultimately done in a way that is similar to many smart devices today. Continuing with comparison to other smart devices, it would be nice to implement an Alexa feature where you could ask how your plants are doing and get notifications of when your plants need to be refilled or need more sunlight.

Another improvement to our system would be an improved soil moisture sensor. At the moment we have one soil moisture sensor on our device. However, if that sensor is misplaced that could be a problem. That problem could lead to inaccurate watering of the user’s plant. Thus it would be advantageous to first of all include directions in the setup of our device so that a mistake is not made in the placing of our device in a plant. These instructions would require adequate preparation and testing on our end to figure out what is the right placement in the first place. In addition, it would be good to have multiple soil moisture sensors attached to a single device, in order to more accurately read the soil moisture of a given plant. Not only this, but because we have learned that soil moisture readings, and the proper soil moisture to water a plant to, may be dependent on the soil type itself, it would be good to look further into this matter and do research on soil moisture versus soil type as we think about our AI algorithm.

Overall the implementation of all these new features and changes will significantly enhance our system and product, making it more marketable and desirable to consumers, and potentially leading to a profitable business.

REFERENCES

[1] Amazon. (n.d.). RAINPOINT Automatic Plant Watering System. Retrieved from https://www.amazon.com/RAINPOINT-Automatic-Watering-Irrigation-Self-Watering/dp/B09GYDBC7Y/ref=sr_1_1_sspa?crid=1A5RL8TEQQBAW&keywords=rainpoint&qid=1702682792&sprefix=rainpoint%2Caps%2C179&sr=8-1-spons&ufe=app_do%3Aamzn1.fos.17d9e15d-4e43-4581-b373-0e5c1a776d5d&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&th=1

[2] American Lung Association. (2017, Feb 15). Do Houseplants Really Improve Air Quality? American Lung Association Each Breathe Blog. https://www.lung.org/blog/do-houseplants-really-improve-air-quality

[3] Jeong S. J., Song J. S., Kim W. S., Lee D. W., Kim H. D., Kim K. J., et al. (2008). Evaluation of Selected Foliage Plants for Improvement of Indoor Humidity. Hort. Environ. Biotechnol. 49 (6), 439–446.

[4] Lee, M. S., Lee, J., Park, B. J., & Miyazaki, Y. (2015). Interaction with indoor plants may reduce psychological and physiological stress by suppressing autonomic nervous system activity in young adults: a randomized crossover study. J Physiol Anthropol, 34(1), 21. https://doi.org/10.1186/s40101-015-0060-8

[5] Wolverton, B. C., Johnson, A., & Bounds, K. (1989). Interior landscape plants for indoor air pollution abatement (NASA Technical Paper 195219). National Aeronautics and Space Administration.