Smart Farm

BY

Gunn Chaiyapornparn 62011118

Narawich Kittijirayu 62011155

Tairo Kageyama 62011257

Jirapong Pansak 62011319

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF ENGINEERING IN COMPUTER INNOVATION ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2022

SCHOOL OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

PROJECT CERTIFICATE

Project Title	Smart Farm	
Student Name	Mr. Gunn Chaiyapornparn	62011118
	Mr. Narawich Kittijirayu	62011155
	Mr. Tairo Kageyama	62011257
	Mr. Jirapong Pansak	62011319
Degree	Bachelor of Engineering in Computer Innovation Engineering	

Project Advisor Signed: _____

(Asst. Prof. Dr. Panarat Cherntanomwong)

Project Title	Smart Farm	
Student Name	Mr. Gunn Chaiyapornparn	62011118
	Mr. Narawich Kittijirayu	62011155
	Mr. Tairo Kageyama	62011257
	Mr. Jirapong Pansak	62011319
Degree	Bachelor of Engineering in Computer Innovation Engineering	
Project Advisor	Asst. Prof. Dr. Panarat Cherntanomwong	
Academic Years	2022	

ABSTRACT

This project proposes a Smart Farm that leverages advanced technology to optimize the cultivation process and increase yield. The farm is equipped with a range of sensors, monitoring systems, and automated tools that provide real-time data on the environmental conditions and other key metrics. Using this data, the system is able to adjust various parameters, such as temperature, humidity, and lighting, to ensure optimal growing conditions for each plant. The proposed Smart Farm aims to provide an innovative and sustainable approach to cultivation, maximizing yield and quality while minimizing waste and environmental impact.

ACKNOWLEDGEMENTS

We would like to express our gratitude to Asst. Prof. Dr. Panarat Cherntanomwong for mentoring us. Her constructive feedback and criticism provided a very constructive meeting.

Additionally, we would like to extend our sincere thanks to the stakeholder, for trusting us and providing a place to work on the project, and lending some pieces of equipment to implement our project, without which this project would not be possible.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS/ABBREVIATIONS

Symbols/Abbreviations	Terms
CIE	Computer Innovation Engineering
SIIE	School of International Interdisciplinary Engineering Programs
MQTT	Message Queuing Telemetry Transport
CoAP	Constrained Application Protocol
UI	User Interface
API	Application Programming Interface
IoT	Internet Of Things
LoRa	Long Range
LoRaWAN	Wide Area Network Based on LoRa
SSL	Secure Socket Layer
TLS	Transport Layer Security
HTTP	Hypertext Transfer Protocol
OCF	Open Connectivity Foundation
AI	Artificial Intelligence
ESP32	Expressif 32 Microcontroller
IR	Infrared Radiation
URL	Uniform Resource Locator
MQTT2DB	Hardware module database service
CO2	Carbon Dioxide

CHAPTER 1
INTRODUCTION

The agricultural industry is rapidly expanding and with it comes the need for innovative and efficient farming solutions due to population growth. In response to this demand, we have developed a Smart Farm system. The system utilizes a variety of sensors to gather real-time data on key growth factors such as temperature, humidity, and light intensity. This data is then processed by our system which adjusts the environment to maintain optimal growing conditions. This allows for precise control of each parameter, ensuring that the plants receive the ideal conditions for their growth. This has the benefit of reducing the need for water and energy, resulting in a more environmentally friendly and cost-effective farming solution.

Overall, the Smart Farm system represents a significant step forward in farming technology, providing farmers with the tools they need to produce high-quality crops in a sustainable and efficient manner.

This chapter introduces the background, objectives, scope, methodology, expected outcomes, and project timelines for the Smart Farm project.

1.1 Background

The surging demand for high-quality plant products, driven by various factors such as increasing consumer interest in health and wellness, culinary trends, and the growing popularity of indoor gardening, has propelled the growth of the horticultural industry. As a result, an increasing number of farmers are entering the market to cater to this rising demand for premium plant products. However, plant farming presents unique challenges. Unlike traditional crops, certain plants require a very specific set of environmental conditions to thrive. Factors such as temperature, humidity, and light intensity must all be carefully controlled in order to produce optimal plant growth and quality.

In response to these challenges, a new generation of plant growers has emerged, incorporating cutting-edge technology to optimize growth conditions and maximize yields. The Smart Farm system is designed to monitor and control growth factors automatically, utilizing sensors and algorithms to maintain optimal growing conditions. By providing the ideal environment for plants, the system aims to increase yields, improve efficiency, and reduce costs. In addition to its technological advantages, the Smart Farm system also offers significant sustainability benefits. By conserving water and energy, the system reduces the environmental impact of farming.

Overall, the Smart Farm system represents a significant step forward in farming technology, providing farmers with the tools they need to produce high-quality crops in a sustainable and efficient manner. As the agricultural industry continues to grow, we can expect to see more innovative farming solutions like the Smart Farm system emerge.

1.2 Objectives

1.2.1 Optimize growth conditions: The primary objective of the Smart Farm system is to create optimal growing conditions for many kinds of plants. This includes maintaining consistent temperature, humidity, and light levels, among other factors.

1.2.2 Increase yields: By providing the ideal growing conditions, the Smart Farm system aims to increase yields and produce high-quality crops.

1.2.3 Improve efficiency: The Smart Farm system is designed to automate many of the farming processes, reducing the need for manual labor and improving efficiency.

1.2.4 Enhance sustainability: The system incorporates sustainable practices such as water and energy conservation, reducing the environmental impact of plants farming.

1.2.5 Reduce costs: By optimizing growing conditions and increasing efficiency, the Smart Farm system aims to reduce costs associated with farming.

1.2.6 Ensure consistency: The system is designed to maintain consistent growing conditions, ensuring that each plant receives the same level of care and attention.

1.2.7 Improve data collection: The Smart Farm system is able to collect vast amounts of data on growing conditions and plant health, providing valuable insights for future farming endeavors.

1.3 Scope

We are creating a system that has 4 main features. Firstly, our system would be able to do manual control of the system. The user would be able to control all the actuators in the system. Secondly, we would have an automation system. This system could schedule tasks like automatically watering the plants at specific times of the day. Thirdly the system could automatically maintain the growth factors such as CO2 level. Finally, to help the farmer have a better idea of their farm, the system would keep track of the parameters and show the data to the farmer.

1.4 Methodology

The process of this project is to be implemented as follows.

1.4.1 Design our system diagram for our system.

1.4.2 Design the ingress to the application from the internet.

1.4.3 Design the hardware solutions for each of the parameters

1.4.4 Implementation and integration of farm

1.4.5 Installation and Testing of the system

1.4.6 Refinement and Optimization

1.5 Expected Outcomes

1.5.1 The system should be able to maintain its value between the user's settings.

1.5.2 A system that is able to create and execute automation according to the user's settings.

1.5.3 The farmer would be able to see the data collected from the sensors.

1.5.4 The farmer would be able to manually control the farm actuators.

1.6 Project Timeline

To manage the project timeline, we have divided the work into three main parts: Frontend, Backend, and Hardware. Each team member has been assigned to a specific part, and we have set a timeline to guide our progress. Our goal has been to adhere to the timeline as closely as possible. Here is an overview of the timeline:

<table>
<tr><th></th><th>Frontend</th><th>Backend</th><th>Hardware</th></tr>
<tr><td>Milestone 1
(January 1st - January 31st)</td><td>UI design</td><td>Designing Database, API</td><td>Design hardware solution</td></tr>
<tr><td>Milestone 2
(February 1st - February 28th)</td><td>Implement UI,
Service deployment</td><td>Implement
Backend</td><td rowspan="3">Develop the hardware system.</td></tr>
<tr><td>Milestone 3
(March 1st - March 17th)</td><td colspan="2" rowspan="2">Software Integration</td></tr>
<tr><td>Milestone 4
(March 18th - April 8th)</td></tr>
<tr><td>Milestone Final
(April 9th - April 30th)</td><td colspan="3">Integration
Installation
Refinement and optimization</td></tr>
</table>

Table 1.1 Table showing the timeline of working.

Despite encountering slight delays in certain areas of the project, we have successfully adhered to our timeline and delivered the expected work. The predetermined timeline has played a crucial role in effectively managing our time and resources, ensuring that we can complete the tasks within the desired timeframe.

The rest of this report is organized as follows:

Chapter 2 provides a comprehensive review of the state-of-the-art farming technologies. It examines various existing systems and technologies related to farm management and automation, highlighting their strengths, weaknesses, and potential areas for improvement.

In Chapter 3, we present the methodology employed and the design process undertaken to develop our farm management system. It outlines the key components, architecture, and solutions proposed to address the challenges identified in the literature review. The chapter also discusses the implementation details and considerations taken into account during the design phase.

Chapter 4 focuses on the evaluation and testing of our system. It presents the results obtained during the implementation and deployment of the system in an actual farm setting. The chapter discusses the performance, functionality, and usability of the system, providing insights into its effectiveness in maintaining the farm and improving overall productivity.

Chapter 5, the final chapter concludes the report by reviewing the work undertaken and summarizing the key findings and contributions of the project. It reflects on the successes and limitations of the system and discusses potential areas for future improvement and expansion. The chapter also highlights the implications of the research and explores avenues for further research and development in the field of farm management systems.

Overall, this report provides a comprehensive overview of the research, design, implementation, and evaluation of our farm management system. It aims to contribute to the existing knowledge in the field and provide insights into the potential of technology-driven solutions for efficient and sustainable farm management.

CHAPTER 2
REVIEW OF LITERATURE

This chapter discusses the technologies and techniques used in Smart Farm. The investigation served 2 purposes: firstly, we wished to identify requirements of indoor plant farming (section 2.1); and secondly, we wished to establish a system to automate the farming of plants indoors so we will be going through the basics of IoT devices (section 2.2) and corresponding sensors (section 2.3) and actuators (section 2.4). Finally, section 2.5 summarizes the chapter.

2.1 Plants growing techniques.

The crops and plants growth will consist of the various stages of plant growth and provide recommendations for optimizing environmental conditions. Understanding the distinct requirements for each stage, including light, carbon dioxide levels, humidity, and temperature, is crucial for successful cultivation. By adhering to appropriate practices, growers can enhance the growth and overall yield of various plants.

Seedling Stage:

The seedling stage typically spans a period of two to three weeks, during which seeds develop into seedlings. It is essential to provide a nurturing environment during this phase to ensure healthy growth.

Vegetative Stage:

The vegetative stage is a phase in the life cycle of a plant during which it focuses on leaf and stem growth. The greatest requirement for nitrogen, which provides the nutrients that drive the formation of new cells, comes during this vegetative stage. Healthy vegetative growth is nourished by the easily absorbed proteins found in organic fertilizers and amino acid supplements.

Flowering Stage:

The flowering stage is when a plant produces its fruit and flowers. During this time, the plant experiences a growth surge and might potentially double in size. They will not actually begin to create fruits and blooms before this growth spike. The best results for farmers come from maintaining ideal humidity and temperature conditions.

Carbon Dioxide Requirements:

To enhance the photosynthesis process, it is recommended to supplement carbon dioxide (CO2) levels during the periods when the lights are on. The optimal CO2 concentration for general plants is not less than 330 parts per million (ppm).

Humidity Considerations:

Maintaining appropriate humidity levels is vital for cultivation. During the light-off periods, humidity tends to rise as the plants undergo dehydration. It is recommended to maintain a humidity range of 40% to 60% to ensure optimal growth conditions.

Temperature Guidelines:

For the vegetative stage, maintaining a temperature range of 20 to 25 degrees Celsius is ideal, while the flowering stage benefits from temperatures between 25 and 30 degrees Celsius. By adhering to these temperature ranges, growers can create favorable conditions for growth.

Optimal cultivation of various plants requires careful attention to each growth stage's specific requirements. By providing the appropriate light, fertilizers, carbon dioxide levels, humidity, and temperature, growers can maximize plant growth, resulting in a successful harvest. Implementing these guidelines will contribute to the overall quality and yield of crops.

2.2 Internet of Things (IOT)

IoT devices typically consist of three key components: sensors, connectivity modules, and actuators.

2.2.1 Sensors

Sensors are responsible for gathering data from the surrounding environment. They can measure various parameters such as temperature, humidity, light intensity, motion, or even detect specific events. The sensors convert these physical measurements into digital data that can be processed and transmitted.

2.2.2 Connectivity Modules

Connectivity modules establish the communication link between the IoT device and the wider network or internet. These modules can include technologies like Wi-Fi, Bluetooth, cellular networks (2G, 3G, 4G, or 5G), or specialized protocols like Lora WAN or Zigbee. They enable the device to connect to other devices or a central hub.

2.2.3 Actuators

Actuators are components that carry out actions based on the received instructions from the network or other devices. They can include motors, switches, valves, or displays. Actuators enable IoT devices to perform tasks or trigger responses in the physical world.

2.2.4 Communication Protocols

To enable seamless data exchange, IoT devices utilize various communication protocols, depending on the application and connectivity requirements. Some commonly used protocols include:

2.2.4.1 MQTT (Message Queuing Telemetry Transport)

MQTT is a lightweight, publish-subscribe messaging protocol designed for constrained devices and low-bandwidth, high-latency networks. It enables efficient communication between IoT devices and servers, using a publish-subscribe model.

2.2.4.2 CoAP (Constrained Application Protocol)

CoAP is a specialized web transfer protocol for constrained devices and networks. It allows IoT devices to communicate over IP networks with minimal overhead and power consumption.

2.2.4.3 HTTP (Hypertext Transfer Protocol)

HTTP is a widely used protocol for communication between web servers and clients. IoT devices can use HTTP to exchange data with web servers, enabling integration with existing web services and applications.

2.2.4.4 Zigbee

Zigbee is a low-power, wireless communication protocol designed for short-range communication. It is often used in home automation systems, where devices need to interact with each other within a limited area.

2.2.5 Data Processing and Analytics

IoT devices generate vast amounts of data, and processing this data efficiently is crucial for extracting meaningful insights. Depending on the device's capabilities, data processing can occur at various stages:

2.2.5.1 Edge Computing

Some IoT devices are equipped with sufficient computational power to process data locally. This approach, known as edge computing, enables real-time analysis and immediate response to critical events, reducing latency and dependence on cloud services.

2.2.5.2 Cloud Computing

In cases where the IoT device lacks computational resources, data is transmitted to the cloud for processing and storage. Cloud-based analytics platforms leverage the power of large-scale computing to perform complex data analysis, generate reports, and derive valuable insights.

2.2.6 Security Considerations

Given the sensitive nature of the data collected by IoT devices, ensuring security is of utmost importance. Security measures include:

2.2.6.1 Authentication and Authorization

IoT devices need to authenticate themselves and obtain authorization before accessing networks or exchanging data. This ensures that only authorized devices can interact with the system.

2.2.6.2 Data Encryption

Data encryption techniques, such as SSL/TLS, are used to protect the confidentiality and integrity of data during transmission. Encryption prevents unauthorized access or tampering of sensitive information.

2.2.6.3 Firmware Updates and Patching

IoT devices should regularly receive firmware updates and security patches to address vulnerabilities and bugs. This ensures that devices are equipped with the latest security measures and safeguards against potential threats.

2.2.6.4 Access Control

Implementing robust access control mechanisms helps regulate device interactions and restrict unauthorized access. This can involve secure user authentication, role-based access control, and secure API endpoints.

2.2.6.5 Data Privacy

IoT devices often handle personal or sensitive data. Implementing privacy measures such as data anonymization, consent management, and secure data storage practices helps protect user privacy and comply with relevant regulations.

2.2.7 Integration and Interoperability

IoT devices are designed to work within a larger ecosystem, seamlessly integrating with other devices and systems. To achieve interoperability, standardization of communication protocols, data formats, and device management frameworks is essential. Industry initiatives like the Open Connectivity Foundation (OCF) and the Thread Group aim to establish interoperability standards for IoT devices.

IoT devices combine sensors, connectivity modules, and actuators to gather data, communicate with other devices or servers, and perform physical actions. Communication protocols enable seamless data exchange, while data processing allows for meaningful insights and automation. Security measures protect device integrity and user data. Integration and interoperability enable IoT devices to collaborate within a larger ecosystem. Understanding these inner workings is crucial for harnessing the potential and ensuring a secure and interconnected IoT environment.

2.3 Sensors

The Smart Farm system uses sensors to gather real-time data on key growth factors such as temperature, humidity, and light intensity, and actuators to control these factors in response to the data collected. The system is controlled by the ESP32 microcontroller, which processes the sensor data and controls the actuators to maintain optimal growing conditions for the plants.

2.3.1 Humidity & Temperature Sensor

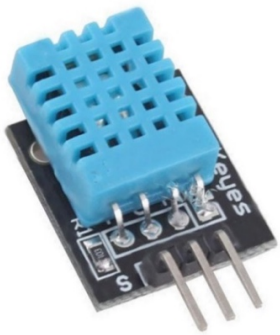

Figure 2.1 DHT11 Humidity & Temperature Sensor [1]

In the case of the Smart Farm system, the DHT11 temperature and humidity sensor is used to monitor temperature and humidity levels in the growing environment. The data collected by the sensor is then processed by the system's microcontroller, which uses it to adjust the environmental conditions to maintain optimal growing conditions for the plants.

The DHT11 sensor is a low-cost and reliable temperature and humidity sensor that is commonly used in various systems and projects. It contains a thermistor and a capacitive humidity sensor in a small plastic housing, which communicates with the system's microcontroller through a single-wire digital interface. This makes it easy to use with microcontrollers like the ESP32.

The thermistor measures temperature by detecting changes in resistance, which are then converted into digital signals that can be read by the microcontroller. Similarly, the capacitive humidity sensor measures changes in the dielectric constant of a thin polymer layer due to water molecule absorption or release. These changes are also converted into digital signals that can be read by the microcontroller.

2.3.2 CO2 sensor

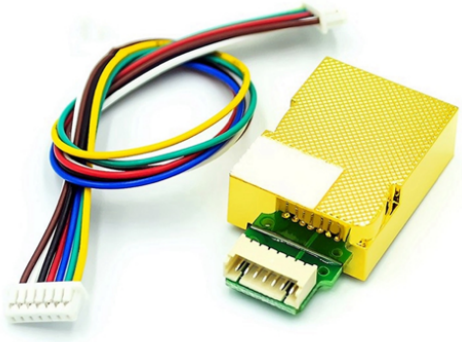

Figure 2.2 NDIR Infrared CO2 Sensor [2]

A CO2 sensor, specifically an NDIR Infrared CO2 Sensor module, is used in the Smart Farm system to collect data on CO2 levels. This data is used to be displayed as parameters and graphs. The sensor operates based on the NDIR principle, using

infrared light to measure CO2 absorption in the air sample. It provides CO2 concentration data in parts per million (ppm) through digital or analog outputs and offers built-in calibration, temperature compensation, and configurable parameters for accuracy and customization. It is commonly employed in applications like indoor air quality monitoring and ventilation control systems that require precise CO2 measurements.

2.4 Actuators

2.4.1 Servo

Servo is implemented in the light system. The farm light uses a dial to control light strength. Strong power and 360-degree spinnable servo can be used to move the dial. The Futaba S3003 servo (360 degrees) satisfies these conditions.

Figure 2.3 Futaba S3003 servo [3]

The S3003 is a popular and reliable servo motor that can rotate continuously through a full 360 degrees. It has a plastic case that contains a DC motor, gears, and a control circuit and communicates with the control circuit through a standard three-wire interface. The S3003 has a torque rating of 3.2 kilograms/ centimeter at 4.8 volts and 4.1 kilograms / centimeter at 6 volts and a speed of 0.23 seconds/60 degrees at 4.8 volts and 0.19 sec/60 degrees at 6 volts, making it suitable for small to medium-sized applications that require continuous rotation.

2.4.2 IR transmitter

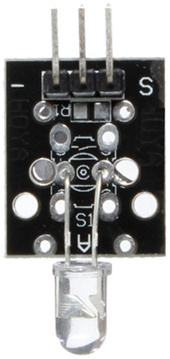

Figure 2.4 IR Infrared 38KHz Transmitter Sensor [4]

The IR Infrared 38KHz (Kilohertz) Transmitter Sensor Module is a small electronic component used for transmitting infrared signals, commonly used in remote control systems for devices like televisions, DVD players, and air conditioners. It contains an infrared LED that emits light at a frequency of 38kHz and has a built-in driver circuit controlled by a microcontroller that enables it to transmit signals. This module is a simple and reliable component that allows for remote control of devices through the transmission of modulated 38kHz signals and is used in a variety of applications. This IR transmitter is used to control air-conditioners.

2.4.3 Diaphragm pump

Figure 2.5 R385 DC 6V-12V Diaphragm Based Water Pump [5]

The R385 DC 6V-12V Diaphragm Based Water Pump is a compact and efficient pump designed for water circulation and transfer applications. This water pump features a diaphragm design, which allows for smooth and consistent water flow. The diaphragm is driven by a DC (direct current) motor, providing reliable and quiet operation. The pump is capable of working with a voltage range of 6V to 12V, offering flexibility in power supply options. Additionally, it has low power consumption, making it energy-efficient and cost-effective. This water pump can be used to deliver water to the Smart Farm. By connecting the pump to the network of hoses running throughout the farm.

2.4.4 Relay

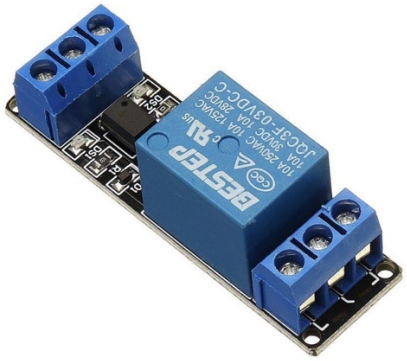

Figure 2.6 1 Channel 5V Low Level Trigger Relay [6]

The 1 Channel 5V Low-Level Trigger Relay is an electronic component that serves as a switch in electrical circuits. It is commonly used in various applications where low-voltage control signals are needed to control high-voltage or high-current devices. This relay operates with a 5 volts input voltage, making it compatible with a wide range of microcontrollers, Arduino boards, and other electronic systems. It utilizes a low-level trigger mechanism, which means it is triggered by a low voltage signal, typically 0 volts or ground level. The Relay is a versatile component that can be effectively utilized to control various electrical appliances in a Smart Farm. By connecting the relay to both the microcontroller and the electrical outlets, it acts as an

intermediary, allowing the flow of electricity from the source to the desired appliance. This enables seamless control and automation of multiple electrical devices within the farm, enhancing efficiency and convenience.

2.5 Chapter Summary

In Chapter 1 we proposed a system to automate the farming of several plants.

In this chapter, the techniques and stages of growing plants are discussed in section 2.1. Observations on the solution for the system architecture were made in section 2.2, and the relevant hardware is observed in section 2.3.

The next chapter presents the design of Smart Farm, which is a system intended to automate and interact with the user. Users will be able to control and monitor the farm through the provided platform.

CHAPTER 3
METHODOLOGY

3.1 Introduction

In Chapter 2 we identified key knowledge and technologies we can use to implement the Smart Farm.

This chapter describes the design of the smart farm system, a system that allows farmers to manage farms at scale with automation. First, the chapter describes the project's design methodology (section 3.2), and interesting problems (section 3.3). A proposed solution is then discussed (section 3.4).

3.2 Design Methodology

The design and development start with the requirements, creating an adjustable system to serve and specifications made by the farmer. This means that our system needs to be flexible to satisfy the requirements of various plants.

3.2.1 System Architecture

The system is designed using a microservice architecture, each of the components is built to do one specific job. The system uses HTTP and MQTT protocols for communication.

The system has 2 main sides split into the software side and the hardware side. The software side connects to each other using HTTP protocol. The app uses Cloudflare tunnels to connect securely to Cloudflare servers. Users that want to use the service can connect to Cloudflare and it will redirect the traffic back to our system.

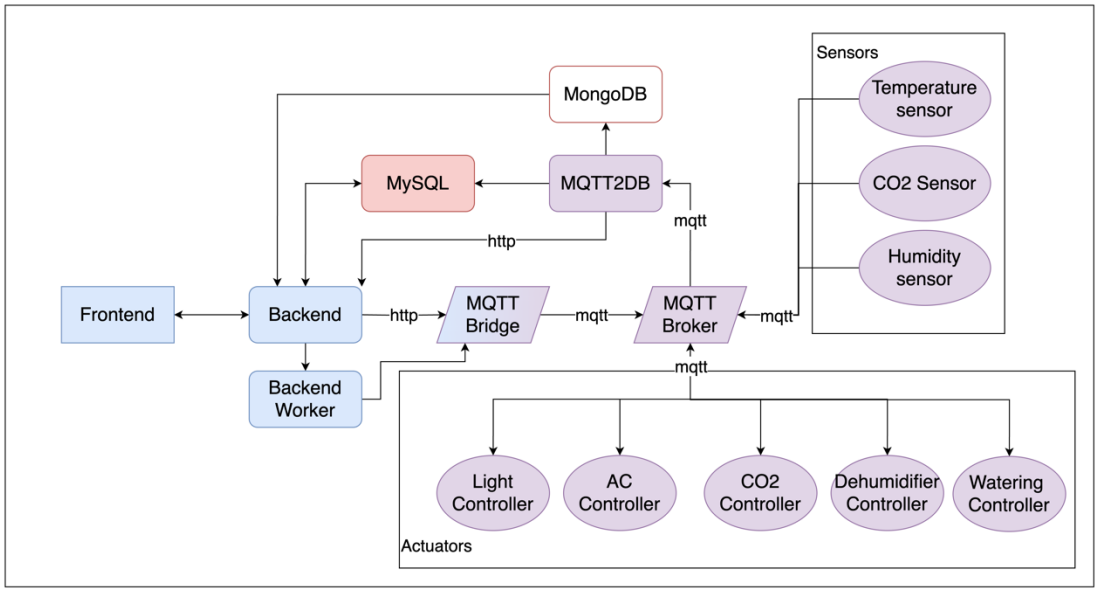

Figure 3.1 System Diagram

When the traffic arrives at our system, it will enter through our reverse proxy. A reverse proxy is a tool that allows traffic to come from one endpoint and distribute it to multiple services. For example, if we have a website we can have /home go to the frontend service and the /api/home go to the backend service. This allows us to have the API and the frontend served from the same URL. The reverse proxy we are using is called Traefik. Traefik will direct traffic from the Cloudflare tunnels ingress and route it between our components.

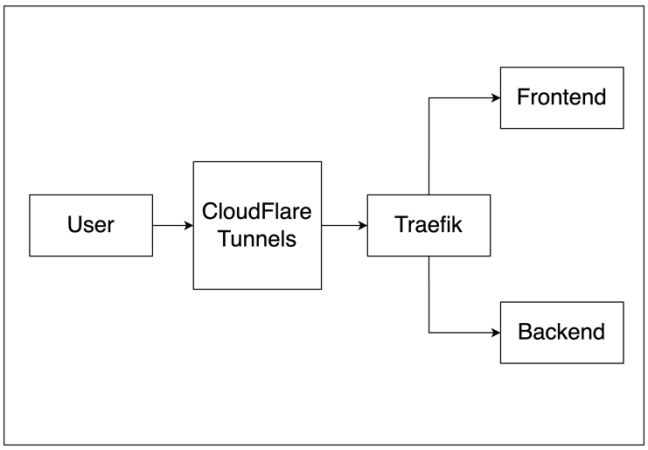

Figure 3.2 Route Diagram

Traefik is a cloud-native reverse proxy, it can connect to the docker socket to provide an easy provisioning interface for the reverse proxy. Instead of writing config files, Traefik can get configs by using the labels of the container. This means that we do not need to change the config file every time we make a change to the application.

The main user interaction would be with the front-end module. This module provides the frontend for the user. This is the only user-facing module since everything can be done from it.

The coordination is done using the backend module, each user interaction is sent to the backend for processing. Actions such as turning on and off the lights will be sent to the MQTT bridge. The bridge converts the HTTP request from the backend module to the MQTT message.

Actions that are not done immediately will be sent to the backend worker system; the backend worker will then create a task to do at the specific time, this action occurs in the same way as the backend module going through the MQTT bridge.

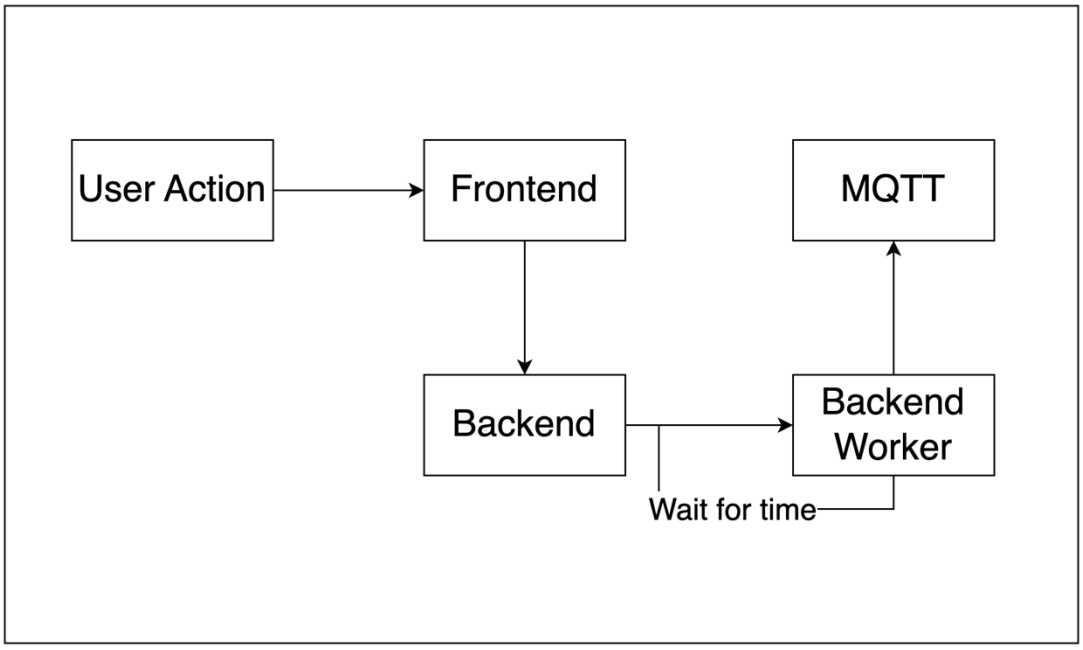

Figure 3.3 Scheduling Diagram

The operations in the hardware space will be on the MQTT side. The hardware sends MQTT messages to the mqtt2db service. The service will store the data in MongoDB and will also update the device state in our MySQL database.

To link both sides of the system we use a HTTP to MQTT bridge, this bridge will allow us to convert HTTP requests to MQTT messages.

The MySQL database will be used to store and collect various types of data and statistics related to the farm. It will serve as a central repository for information such as farm details, including location, size, and owner information. Additionally, it will store real-time data regarding the status of air conditioners, including their operational state, temperature settings, and any relevant alerts or notifications.

MongoDB will be employed specifically for storing time series data generated by sensors within the farm. This data will include temperature readings, humidity levels, and carbon dioxide concentrations. MongoDB's flexibility and scalability make it an ideal choice for handling large volumes of time-stamped data points, enabling efficient storage, retrieval, and analysis of the collected information.

MQTT2DB functions as the repository service for the hardware module, assuming the responsibility of transmitting and receiving data between itself and the hardware module.

The hardware module consists of several distinct modules, including the sensor module, light controller module, watering controller module, dehumidifier module, carbon dioxide controller module, and air conditioner controller module.

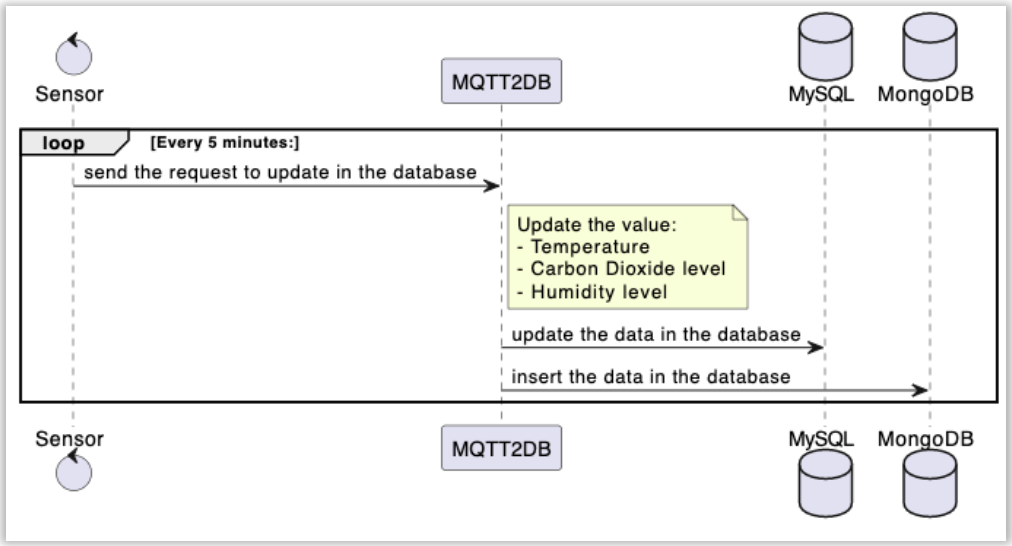

Figure 3.4 Sensor module sequence diagram

The sensor module comprises three primary sensors: a temperature sensor, a humidity sensor, and a carbon dioxide sensor. These sensors are integrated into a single module. The sensor module establishes a recurring loop to continuously send data to the MQTT2DB service, ensuring that the data is updated in the MySQL database for current information, while also inserting it into the MongoDB database for time series data.

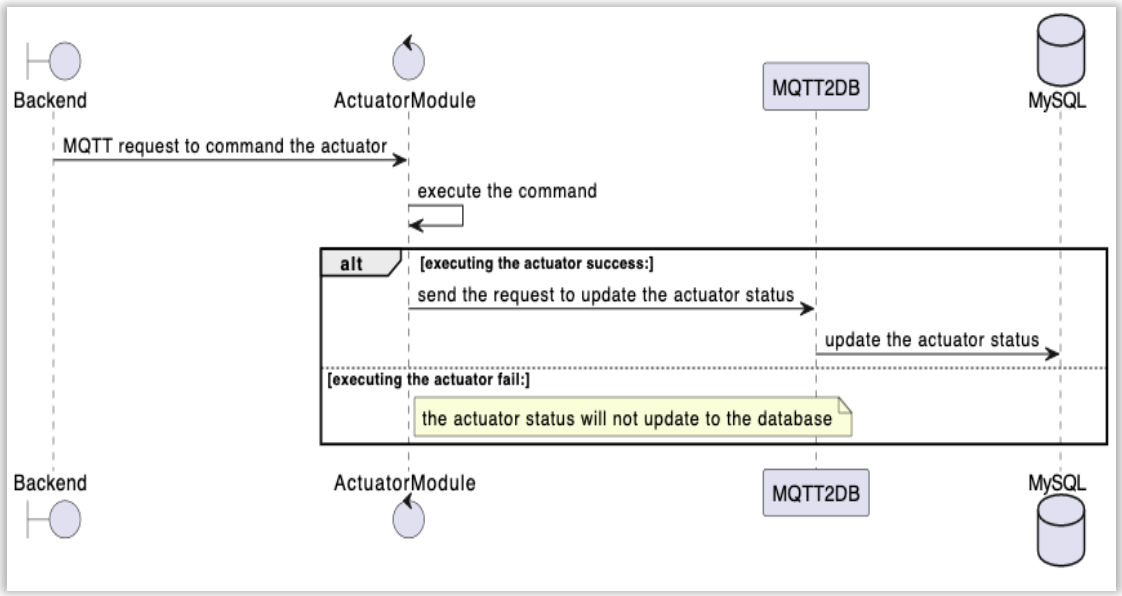

Figure 3.5 Actuator module sequence diagram

The actuator modules, such as the light controller module, watering controller module, dehumidifier module, carbon dioxide controller module, and air conditioner controller module, receive MQTT requests from the backend system and respond accordingly. Once the requested actions have been executed, these modules send MQTT requests to MQTT2DB to update the status of each module in the database.

3.2.2 User platform

The software component acts as a vital intermediary linking users to the hardware, enabling them to efficiently monitor and regulate farm performance. Through its user interface, the software fosters an improved user experience and facilitates user interaction with the application. Our software is comprised of four key elements, namely the front-end, backend, backend-worker, and MQTT2DB, each of which plays a critical role in supporting the system's functionality.

3.2.2.1 Frontend

The front-end component of our system is developed to fulfill the stakeholders' requirements in terms of functionalities. It provides users with a platform to conveniently monitor the status of farms and control all the devices within them. The interface allows users to conveniently schedule watering, air conditioning, and farm lighting, as well as set thresholds for the system to maintain regarding CO2 and humidity levels.

Following the requirement-gathering stage, we employed Figma, a collaborative tool for developers and UI designers, to design the user interface. With Figma, our team was able to efficiently create, share, and prototype the user interface for our web applications.

In order to ensure accessibility for users on various devices, our design includes both a landscape layout for computer screens and a portrait layout for mobile phone screens. Our goal is to create a web application that can be easily accessed and used by users regardless of the device they are using. Moreover, we design the interfaces in both light and dark themed, to correspond with the current trends.

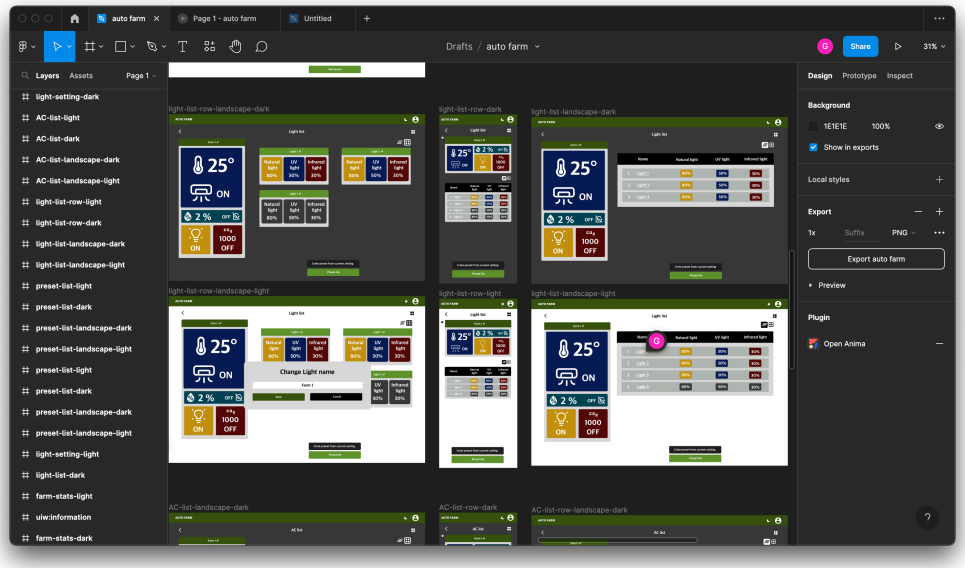

Figure 3.6 The Admin registration page

In the design, we use different colors and icons to represent each sensor and actuator, so it is easier for the user to recognize each component.

1. **Blue** – represents the AC and the temperature.
2. **Teal** – represents the dehumidifier and the humidity level.
3. **Yellow** – represents the farm light.
4. **Red** – represents the CO2 tank and the CO2 level.
5. **Light** blue – represents the watering system.

Figure 3.7 The interface shows the farm status and the controlling panel.

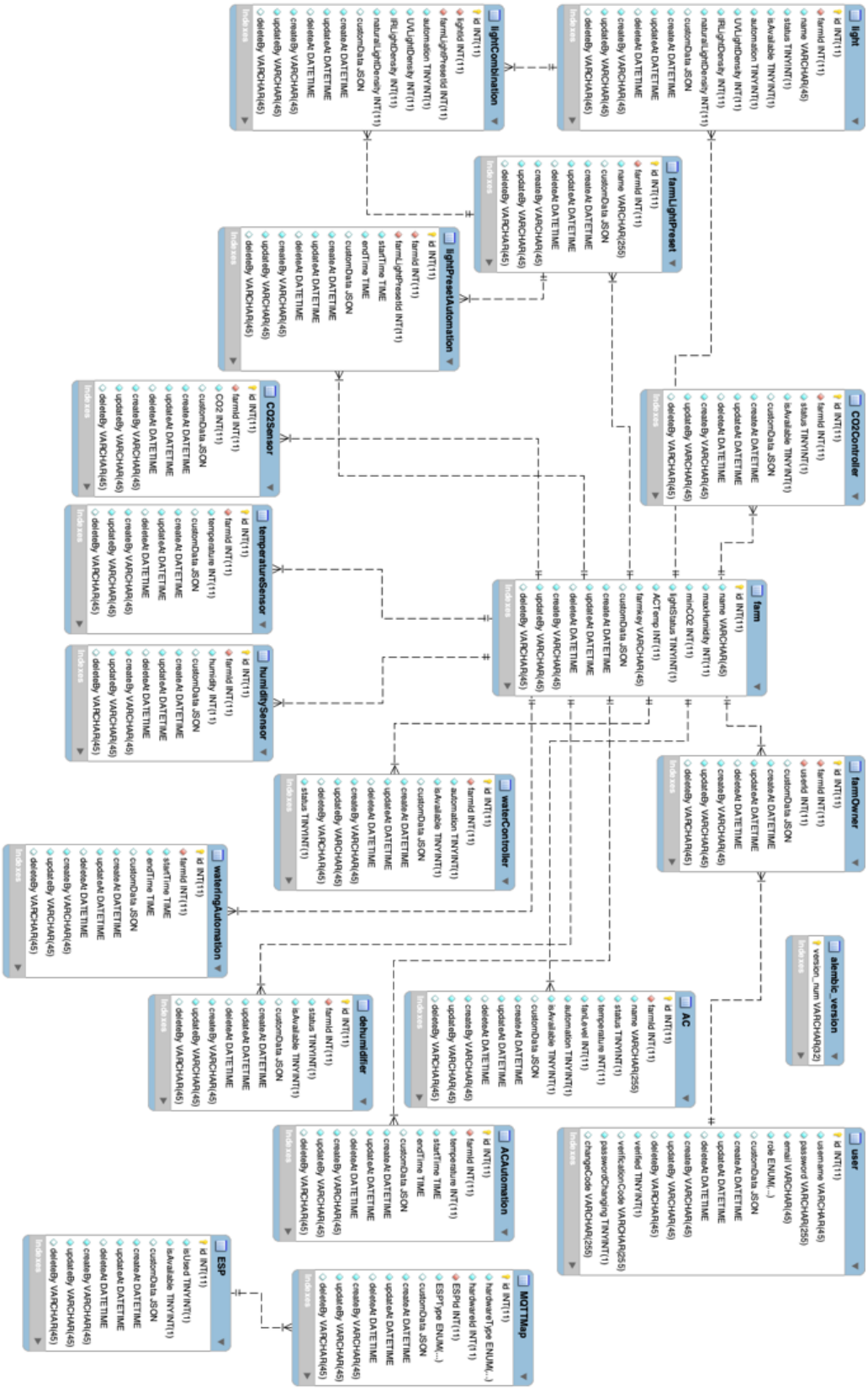

Figure 3.8 The ER diagram for entities in the system.

3.2.2.3 Entities

Upon completing the Figma design, our next step was to establish the domain of the system, encompassing all entities and their associated data structures in the database. Our system is supported by two databases, MySQL, which stores farm, sensor, and automation data, and MongoDB, a NoSQL database that serves as a time-series database for temperature, CO2, and humidity levels.

The data in the MySQL database consists of 6 main sections there are users, farm, sensors, actuators, automation, and ESP32.

1. **Users** – stores all the user-related data.
    a. **Users –** Stores user information such as username, email, and user id.
    b. **FarmOwner** – Stores the linkage between users and farms. This relationship is many to many. A user can own multiple farms and each farm can be owned by multiple users.

2. **Farm** – Stores the farm information.
    a. **Farm** – This table stores the farm information and statuses. Some of the information is farm name, farm key, CO2 threshold level, and humidity threshold level.
3. **Sensors** – This group contains all the information about each sensor we integrate into the farm.
    a. **CO2Sensor** – This table stores the CO2 sensors' information and keeps the CO2 level measured by each sensor.
    b. **temperatureSensor** – This table stores the temperature sensor information and keeps a record of the measured temperature in degrees Celsius.
    c. **humiditySensor** – Similar to the temperature sensor and CO2 sensor table. This table stores all the information about the humidity sensor together with the measured humidity level.
4. **Actuators** – This group is similar to the sensors, but rather keeps the information about all the actuators and electrical devices integrated in the farm. This relationship is one-to-one since an actuator can only belong to a single farm.

a. **Light** – This table stores all the light information including the light name, light status, and light automation status.
b. **CO2Controller** – This table stores the information about the CO2 controller, which is attached to the CO2 tank. The data stored in this table includes the status of the CO2 tank.
c. **WaterController** – This table stores the status of the water controller. This controller is the relay connected to the water pump. If the status is true, that means the pump is on.
d. **AC** – this table stores the AC or the air conditioner data. The information includes the name, status, and the temperature.
e. **Dehumidifier** – This table stores the data of the dehumidifier. This information includes the status of the dehumidifier which is the status of the relay connected.

5. **Automation** – This group basically keeps all the scheduled tasks for the backend–worker service. The basic data kept in each table is the name of the start time and the end time of the task.
    a. **LightPresetAutomation** – This keeps the scheduling task of the farm light in each batch. The strength of the Natural light, Infrared light, and Ultraviolet light is retrieved from the preset assigned to the scheduling task.
        i. **farmLightPreset** – This table keeps the information about the preset of the light. Preset means the combination of light strength in each batch. This will be retrieved once the scheduled tasks are conducted.
        ii. **LightCombination** – This keeps the combination of Natural light, Infrared light, and Ultraviolet light. And all these combinations belong to the light preset.
    b. **ACAutomation** – This table keeps the scheduling tasks of the AC in each farm. The information in this table includes both turn-on time, turn-off time, and the temperature.
    c. **WateringAutomation** – This table also keeps the scheduling task but since we design the watering system to turn off automatically after 5 minutes, this table keeps only the start of the scheduled task.

6. **ESP32** – This table keeps the information about the ESP32 and the linkage between ESP32 and actuators or sensors. We use ESP32 as the microcontroller that listens to the MQTT request to command the sensors and actuators.
    a. **ESP** – This table stores the information of each ESP32 including the status and the availability of the ESP32.
    b. **MQTTMap** – This table keeps the information on the linkage between each ESP32 and the actuators or sensors.

The MongoDB database is utilized to store data records with corresponding timestamps. This data is aggregated to present the current growth factor stats to the users and is also used to generate graphs that illustrate the changes in these growth factors over time.

Temperature readings, humidity levels, and carbon dioxide concentrations will be recorded in a MongoDB database, along with corresponding timestamps. This data will serve as a comprehensive repository of environmental information. The recorded data will be aggregated to provide users with real-time growth factor metrics. Additionally, the collected dataset will be used to generate graphical representations, illustrating the changes in these growth factors over time. By analyzing the trends and patterns depicted in the graphs, users can gain valuable insights into the current environmental conditions and their impact on various processes or systems.

3.2.2.3 Backend

After finalizing the entities, our team proceeded to discuss the API specification that will interact with the frontend. Based on the entities, we developed the API design categorized into six groups: Authentication, Default, Admin, Farm, Light, and AC.

1. **Authentication** - handling all authentication processes and token management.

2. **Default** - handling account-related requests such as password changes or user creation.

3. **Admin** - handling all actions on the admin portal platform, which is the platform for admins to administrate the overall system.

4. **Farm** - handling all farm-related actions from creation to making changes.

5. **Light** - handling all farm light requests from querying, and creating, to making any updates to the setting.

6. **AC** - similar to light but handling all AC requests instead.

To facilitate development and ensure seamless frontend and backend integration, we documented the API in Swagger. Swagger is a platform for API documentation that allows developers to keep a record of the API path, input, and output for each API. This documentation helps in developing the application when multiple developers are working on the same project and is essential for the proper integration of the frontend and backend.

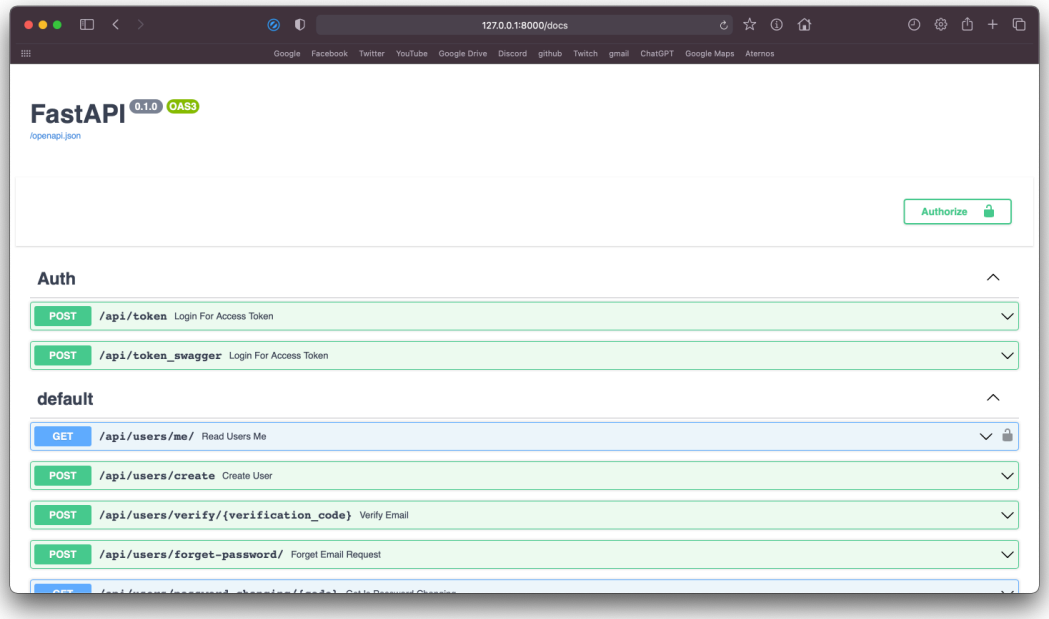

Figure 3.9 The swagger documentation for API.

Our project has a total of 57 APIs, and each API includes a process of data checking and analysis before adding or updating data to the corresponding database.

These APIs also adjust the data format retrieved from the database and send them back to the front end. This means that all the business logic occurs in the backend.

The backend service serves as a vital link between the frontend, databases, and other services. In addition to managing data, it also handles scheduling tasks by sending requests to the backend-worker. Some APIs directly send requests to the MQTT bridge, which then translates them into MQTT requests and forwards them to the MQTT broker. The broker then relays these requests to the ESP32s, which are connected to the actuators and sensors. This ensures that the backend is responsible for managing all the communication between the frontend and the hardware components in the system.

3.2.3 Backend-worker

The backend-worker service plays a critical role in automating farm-related tasks. This service is primarily responsible for scheduling activities like turning on/off the lights and air conditioner and watering the plants.

Upon initialization, the backend-worker retrieves all scheduled tasks stored in the database. The scheduler maintains a record of the timing and payload needed to be sent to the MQTT when the time arrives. The payload consists of the activation status and metadata like temperature, natural light strength, ultraviolet light strength, and infrared light strength. These payloads are then forwarded to the ESP32, which controls the actuators as per the received payload.

Users can create, update, and delete tasks from the user interface. The request is then passed down through the primary backend service and eventually reaches the backend-worker service. The scheduler and database are updated accordingly, to ensure synchronization between the database and the scheduler.

The execution of the task is based on the server's local time, while the time stored in the database is in GMT+0. To ensure our implementation can be installed globally, we implemented an appropriate time zone conversion.

3.2.4 Hardware

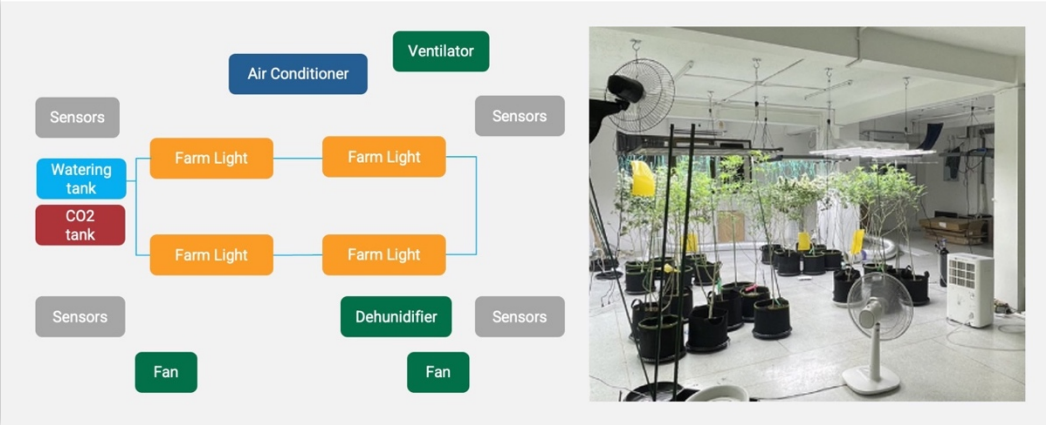

Figure 3.10 Layout of the Farm

The figure above represents the layout of the farm, including the installed hardware. There are four batches of plants in the farm, and lights are installed to provide illumination for each batch. Sensors are placed at every corner of the farm to monitor various parameters.

The dehumidifier and the fan are positioned at the front of the farm to create airflow directly towards the ventilator at the back. This setup helps in reducing the humidity level inside the farm. The Carbon Dioxide emitter and the watering system controller are installed at the side of the farm. Additionally, the watering system is connected to every plant in the farm through hoses for efficient watering.

In order to accurately track the various variables that can affect plant growth on your farm, it is essential to use three types of sensors: a Carbon Dioxide (CO2) sensor, a humidity sensor, and a temperature sensor. These sensors operate continuously, collecting real-time data on the CO2 levels, humidity, and temperature within the farm. This information is then updated and stored in a database that can be accessed by the user at any time.

To control the electrical appliance within the farm, each appliance will have specific requirements. For instance, the lighting system must be able to turn on and off, as well as adjust the light intensity. On the other hand, the air conditioning system only

needs to turn on and off. To control the watering system, dehumidifier, fan, and ventilator, a relay will be used. Similarly, a solenoid and relay will be used to control the CO2 emitter. The air conditioning system, meanwhile, will be controlled using an infrared transmitter that is attached to the system itself.

As for the lighting system, a servo motor will be used to adjust the light intensity and to turn the lights on and off by adjusting the light intensity to zero.

All of the sensors and actuators mentioned above will be controlled using an ESP32. This device will receive MQTT requests, which can trigger an action, such as activating a piece of equipment or updating the value of a variable that is being tracked within the farm.

3.2.5 Administration platform

The design of the admin portal was based on the methodology of ensuring comprehensive administration of the Smart Farm. It was identified that a system was required to facilitate the management of user accounts, farms, and sensors, and to configure the installation of each sensor and actuator. As such, the admin portal needs to be developed to provide the necessary functionality for the administration of the system.

The admin portal needs to consist of three main sections: the list of users, the list of farms, and the list of ESP32s. Each of these sections serves a specific purpose to enable efficient management of the Smart Farm.

3.2.5.1 User Administration

The list of users serves as a fundamental tool for managing user accounts, providing administrators with the ability to track the status of each account. Additionally, this feature grants access to the list of farms owned by each account, which proves beneficial in the event of user-reported issues requiring troubleshooting.

3.2.5.2 Farm Administration

The list of farms provides a means for managing the Smart Farm, including the ability to view the details of each farm, sort and search the list of farms, and assign farm keys to users for the purpose of connecting to the farm. The farm key functionality allows for the seamless integration of new users and ensures the security of the system.

3.2.5.3 ESP32 Administration

The list of ESP32s is a critical component of the admin portal as it provides the means to configure the installation of each sensor and actuator. This page lists all the sensors and actuators categorized by the type of devices they are connected to and provides the functionality to sort and search the list based on ESP32 ID. Admins can also create new ESP32s and assign them to specific devices.

Overall, the design methodology of the admin portal prioritizes the comprehensive management of the Smart Farm through the provision of essential functions for user account management, farm management, and sensor configuration. This approach ensures the seamless operation of the system and facilitates ease of use for all users3.2.6 Main feature sequence diagram.

3.2.6.1 Monitoring

When it comes to monitoring, the farm has two components. The first component is focused on monitoring the current status of the farm. The frontend sends an HTTP request to the backend to initiate this process. The backend then queries the aggregated sensor data from MongoDB, aiming to find the average data across all sensors. Additionally, it retrieves all the farm data from the MySQL database. After gathering the necessary information, the backend generates a response and sends it back to the frontend.

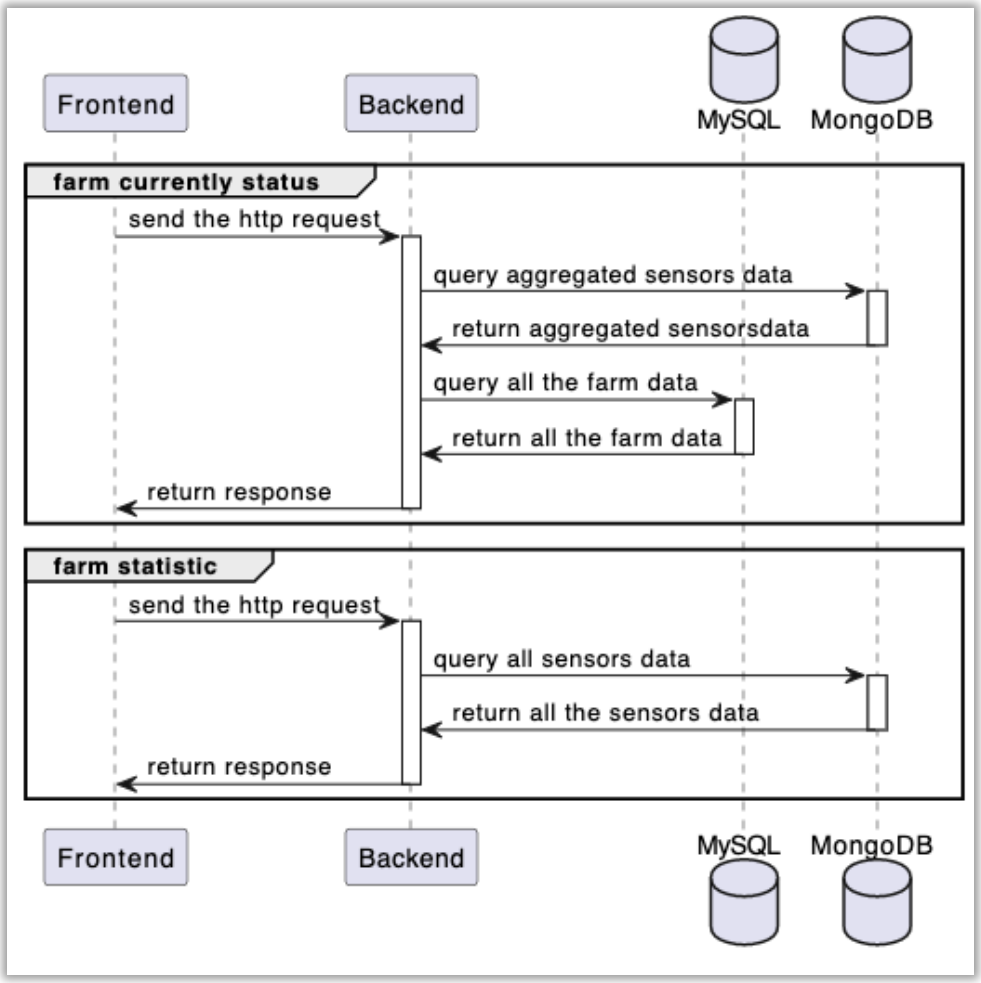

Figure 3.11 Monitoring sequence diagram

The second component involves monitoring the statistical data related to the farm's status. Similar to the previous scenario, the frontend sends an HTTP request to the backend to trigger this process. The backend, in turn, queries all the aggregated sensor data from MongoDB to obtain the historical status of the farm. Once the backend collects the relevant data, it constructs a response and sends it back to the frontend.

Subsequently, the frontend takes charge of organizing and presenting the retrieved data in a visually appealing manner. This involves rearranging and decorating the data to create graphs and tables that facilitate better visualization and understanding of the farm's status.

3.2.5.2 Manually Control

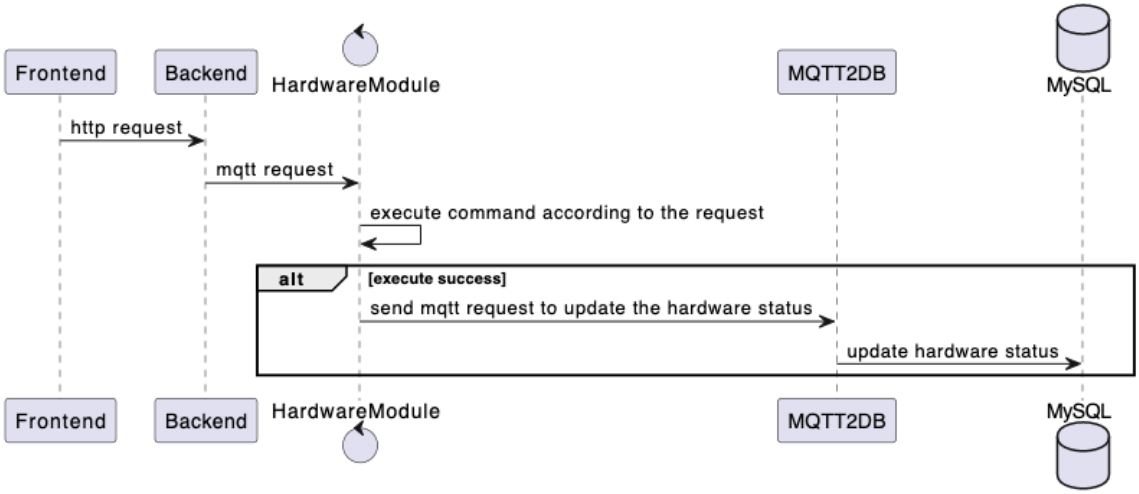

Figure 3.12 Manually Control sequence diagram.

In this system architecture, the frontend is responsible for sending HTTP requests to the backend. The backend, upon receiving the request, utilizes MQTT (Message Queuing Telemetry Transport) to send a command to the hardware module. The hardware module receives the command and executes it based on the payload provided in the request.

Once the hardware module has executed the command, it sends an MQTT request to MQTT2DB (MQTT to Database) to update its own status. MQTT2DB acts as an intermediary, receiving the request from the hardware module and facilitating the update process. Finally, MQTT2DB updates the status into a MySQL database, ensuring that the latest status information is stored persistently.

This architecture allows for manual control of the hardware module, where commands can be initiated from the frontend and executed by the hardware module. The MQTT protocol facilitates efficient and lightweight communication between the backend, hardware module, and MQTT2DB, enabling real-time updates and data synchronization with the MySQL database.

3.2.5.3 Time Automation

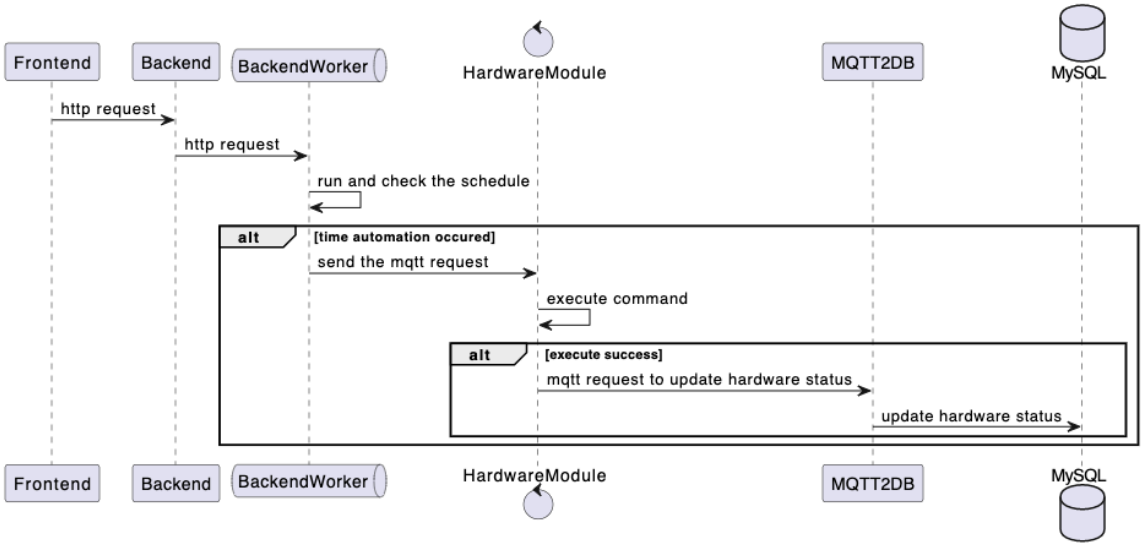

Figure 3.13 Time Automation sequence diagram

For time automation, the frontend sends an HTTP request to create a time automation schedule within the system and sends it to the backend. The backend, upon receiving the request, sends another HTTP request to the Backend Worker to create the scheduler. The Backend Worker is responsible for generating the schedule for executing the command and storing it in the database.

Once the scheduled time is reached, the Backend Worker sends an MQTT request to the Hardware module. The Hardware module, upon receiving the request, executes the command based on the payload provided in the request. It diligently follows the instructions.

After executing the command, if everything goes smoothly, the Hardware module sends an MQTT request to MQTT2DB, requesting an update of its own status. MQTT2DB serves as the intermediary and receives the request from the Hardware module, ensuring the status update takes place.

Finally, MQTT2DB updates the status in the MySQL database, ensuring that the latest status information is securely stored for future reference.

So, in this time automation process, the frontend initiates the creation of the schedule, the Backend Worker handles the scheduling and execution, the Hardware module performs the command execution, and MQTT2DB and the MySQL database handle the status updates and storage.

3.2.5.4 Threshold Trigger.

Whenever the sensor module is initialized, it sends an MQTT request to the MQTT2DB service. The MQTT2DB service then queries the database to retrieve the threshold values for carbon dioxide and humidity levels and sends them back to the sensor module. The temperature sensor, carbon dioxide sensor, and humidity sensor in the module capture data every second. Every five minutes, the sensor module sends the data to the MQTT2DB service to update the MySQL and MongoDB databases.

Upon capturing data, the sensor module compares it with the threshold values. If the carbon dioxide level reaches the threshold, the sensor module sends an MQTT request to the MQTT2DB service to activate the Carbon Dioxide controller in the farm. The MQTT2DB service then sends an MQTT request to the Carbon Dioxide controller to activate the Carbon Dioxide emitter. After activating the emitter, the Carbon Dioxide controller sends a response MQTT request back to the MQTT2DB service to update its status in the database.

Similarly, if the humidity level reaches the threshold, the sensor module sends an MQTT request to the MQTT2DB service to activate the Dehumidifier controller in the farm. The MQTT2DB service then sends an MQTT request to the Dehumidifier controller to activate the dehumidifier. After activation, the Dehumidifier controller sends a response MQTT request back to the MQTT2DB service to update its status in the database.

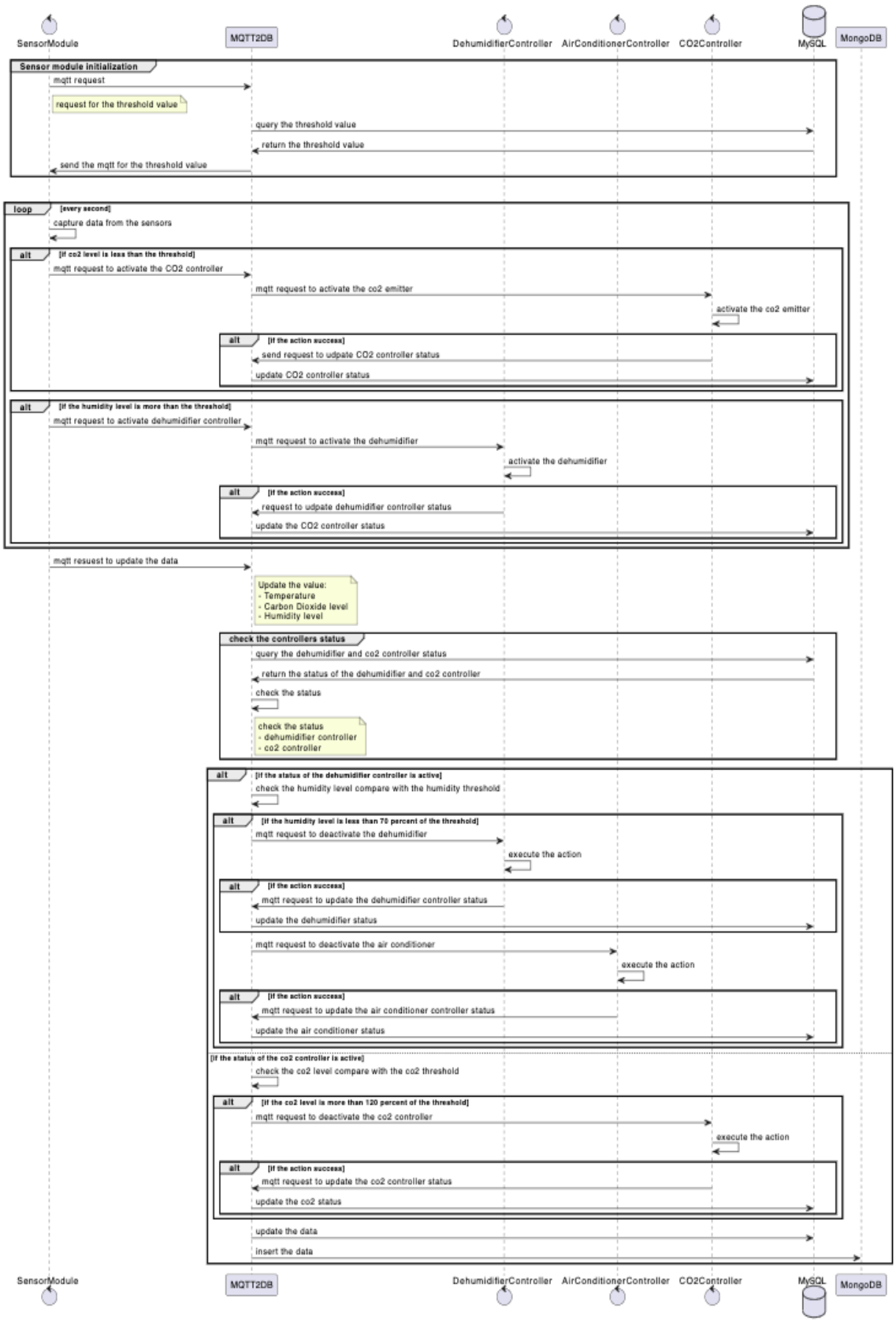

Figure 3.14 Threshold Trigger sequence diagram

During the 5-minute data transmission intervals, the MQTT2DB service first checks the status of the Carbon Dioxide controller and the Dehumidifier controller. If they are activated, the MQTT2DB service compares the current Carbon Dioxide level and humidity level with their respective thresholds. If the difference exceeds twenty percent, the MQTT2DB service sends an MQTT request to deactivate the controllers.

3.3 Interesting Problems

The interesting problem we are facing involves the ability to override commands in a system, particularly in the context of controlling a dehumidifier. One specific scenario arises when a user wants to turn off the dehumidifier even if it has been automatically turned on by a threshold trigger. It is important to ensure that the system does not allow any command, except for the reset threshold command, to override the user's input.

Furthermore, we encounter difficulties in finding hardware components that align with the specific requirements of the problem. We need to identify suitable hardware options that are compatible with the ESP32 platform. This entails considering factors such as voltage requirements, required amplitudes, and the availability of relevant libraries for each hardware component.

Another challenge we face relates to the accuracy of the sensor data. It is crucial to acknowledge that sensors may not always provide perfectly accurate readings. In this case, the sensors capturing data on carbon dioxide and humidity levels may exhibit limitations in accuracy.

In addition, there is a lack of an indicator to signify the proper functioning and connection status of the ESP32 module. Consequently, we may not be immediately aware if the ESP32 becomes disconnected from the internet or the MQTT broker.

Testing the threshold trigger system in real scenarios poses another challenge. Given the slow pace at which carbon dioxide and humidity levels change, it becomes difficult to assess the system's performance in realistic environments.

The ultimate objective of the problem is to maintain optimal levels of carbon dioxide and humidity in a farm setting. However, due to the limitations and uncertainties surrounding the threshold trigger system, it becomes challenging to guarantee its effectiveness in real-life situations.

3.4 Proposed Solution

3.4.1 User platform

At the login page, the user will initiate their journey by entering their credentials. They can either enter their username or email along with the password to access their account. If the user has forgotten their password, they can click on the "forgotten password" option to navigate to the password reset page. Similarly, if they want to create a new account, they can click on the "register" option to navigate to the registration page.

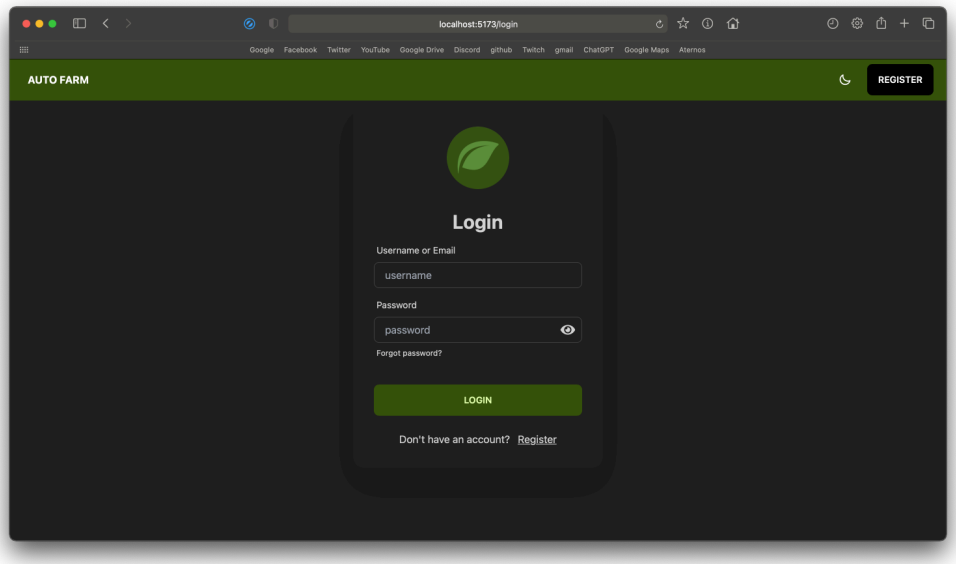

Figure 3.15 The user login page

Users are required to provide valid information to register on the platform. The registration page requires users to fill in their username, email, and password. To avoid duplicates, both the username and email must be unique. If users enter duplicate

information, they will be prompted to provide unique details. The password must contain at least one capital letter and one numerical letter to be considered valid.

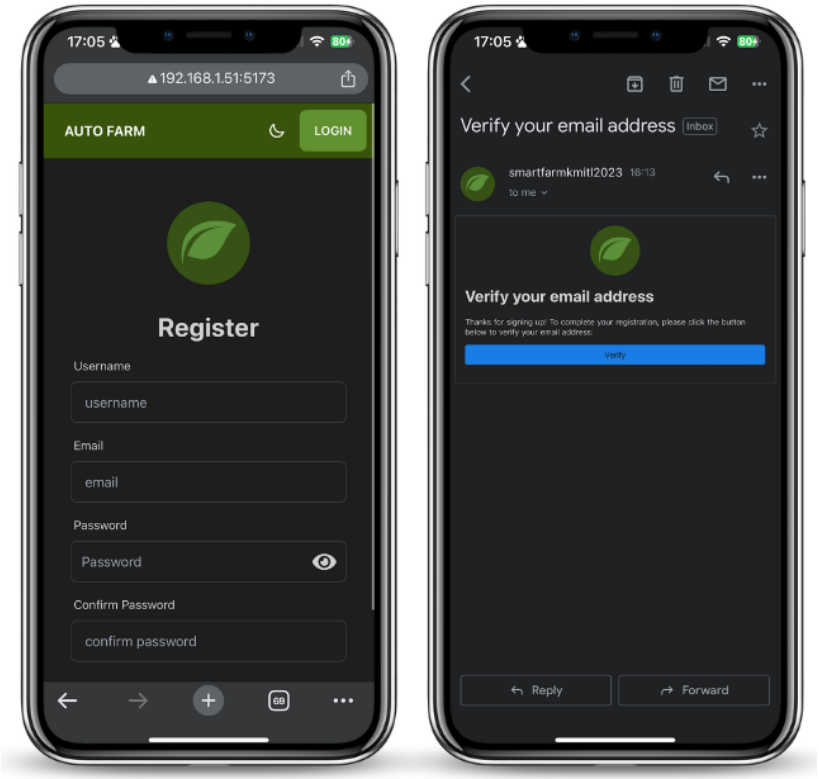

Figure 3.16 The registration page and verification email

Once the user submits the registration form, the system will send a verification email to the registered email address. The user must click the confirmation button in the email to verify their account. If the account is not verified, the user will not be able to log in. Once the verification is complete, the user can sign in using their registered username and password.

If a user forgets their password, they can request a password change by clicking on the "forgot password" button on the login page. They will need to enter their registered email for the password change. If the email is registered in the system, the password change request will be sent. Otherwise, nothing will happen. This is to prevent attackers from getting valid email addresses.

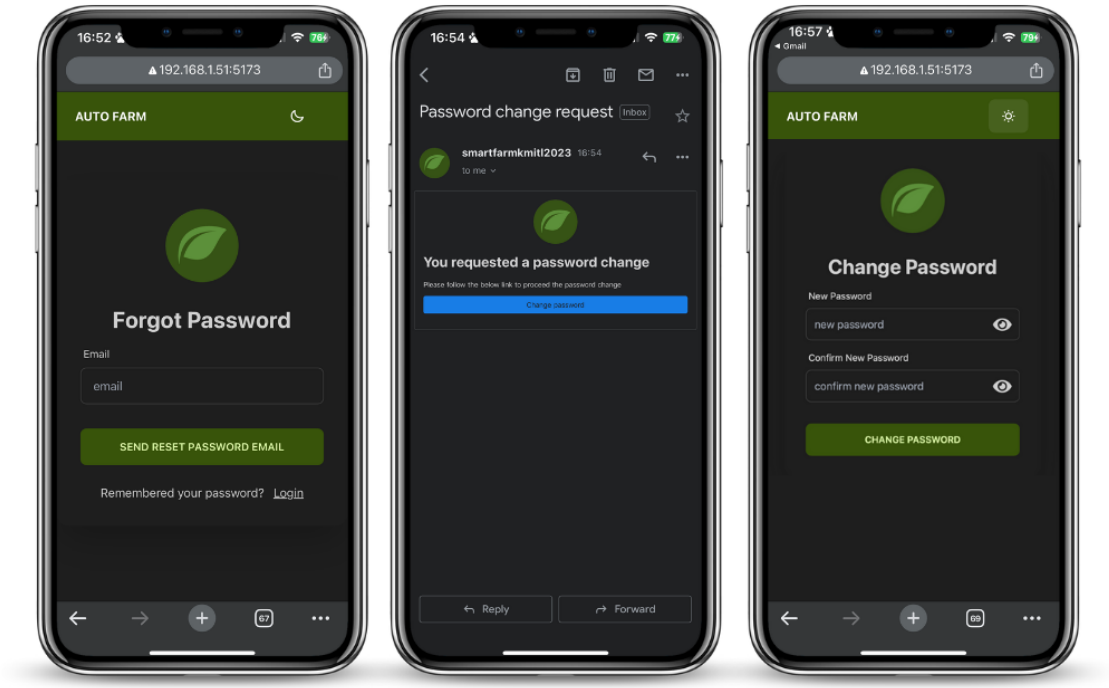

Figure 3.17 The password change process

After clicking the "change password" button in the email, the user will be redirected to the "change password" page. Here, they can enter their new password which must meet the same requirements as when registering. The password must contain at least one capital letter and one numerical letter to be considered valid. Once the password has been changed, the link will become invalid so the user cannot change the password using the same link again. Finally, the user can log in with the new password.

Next in the user journey, the user will be redirected to the list farm page upon login. In case the user has no farm linked to the account, they can contact the admin and request the farm key. Once the user adds the farm to the account by entering the farm key, the farm panel will be displayed on the farm list page. Users can always add a new farm to an account after the new farm installation has been completed. For the farm list page, this enables users to monitor the status of each much more easily. When the user wants to see more detail or wants to control that farm, the user can click on the farm panel to navigate to the farm information page.

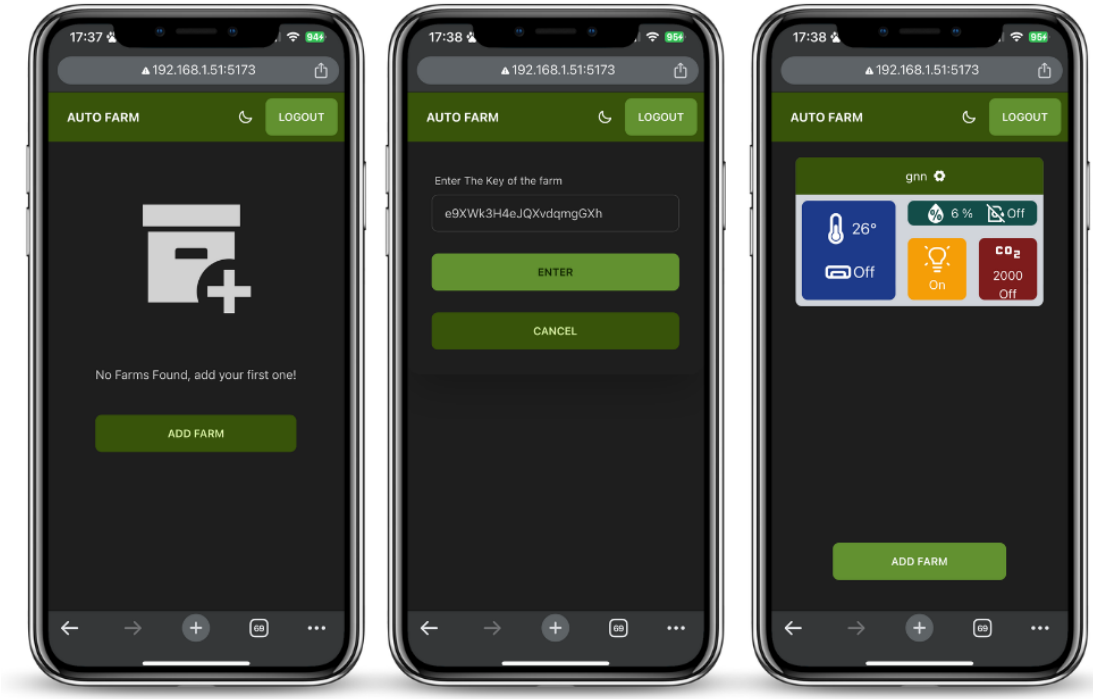

Figure 3.18 The farm linking process.

If a user wants to make changes to a farm, they can do so by clicking on the farm panel, which will take them to the farm settings page. On this page, the user will be provided with a panel that displays the status of the farm, similar to the panel on the farm listing page. The status on the panel includes:

1. Status of the Air conditioner [on/off]
2. Room temperature in degrees Celsius
3. Status of the dehumidifier [on/off]
4. The humidity level in percentage
5. Status of the farm light [on/off]
6. Status of the carbon dioxide tank [on/off]
7. The carbon dioxide level in parts per million (ppm)
8. Farm name

The application uses color coding to help users categorize data more easily. The specific details about the colors can be found in section 3.2.2.1 of this report. These colors are consistently used throughout the application to enhance its usability.

Figure 3.19 The panel showing farm status.

The farm setting page offers three essential features to users. In addition to monitoring the farm status, this page allows users to manually control the actuators, schedule tasks for certain devices, and set thresholds for the carbon dioxide and humidity levels that the system should maintain.

Figure 3.20 The manual control buttons.

1. **Control the actuator manually.**

On this page, users will be presented with a series of buttons that display the current status of each actuator and enable them to switch them on or off. After a user initiates a request, it will be sent via the MQTT bridge and converted into an MQTT request, which will then be relayed to the MQTT broker to execute the command on the ESP32s linked to the relevant actuators. The following is a list of the actuators available:

a. **Farm light** – Upon clicking this button, a request will be sent to all farm lights currently set to be automated on the farm. The combination of Natural light, Ultraviolet light, and Infrared light strength will be adjusted based on the settings previously set by the user.

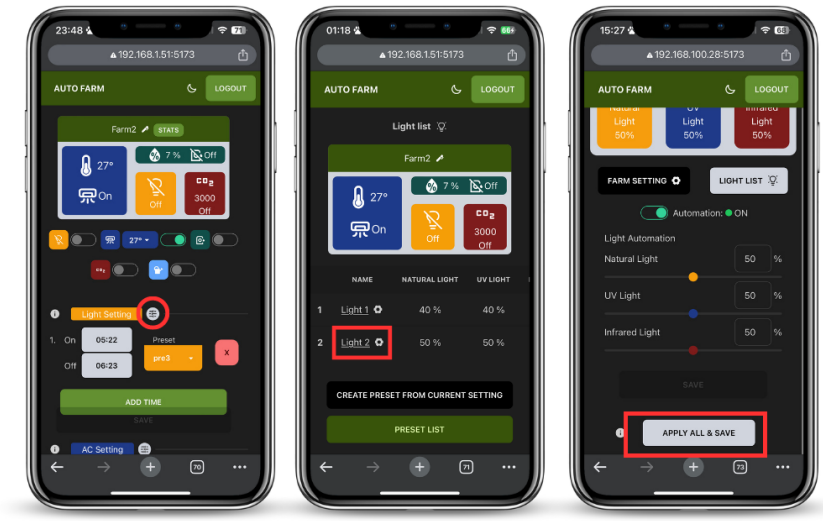

Figure 3.21 The light strength combination setting.

To access these settings, users can click on the settings button located next to the light topic section. This will redirect them to the list of lights in the farm, where the combination of light strength is also displayed. By clicking on a specific light, the user will be directed to the light strength combination setting page, where they can set the percentage of Natural, Ultraviolet, and infrared light strengths. The percentage can be adjusted in increments of 10 from 0 to 100, which corresponds to the light fixture on the actual farm light. Users can also enable or disable the automation of the light

through the button on this page. When automation is disabled and the user turns on the light, it will not be turned on. Here on this page, the user can also change the name of the light to the user's preferences. Users can also choose to apply the current setting to every light in the farm as well.

b. Air conditioner – Just like with the farm lights, turning on the air conditioner will send a request to any automated air conditioner. Users can also adjust the temperature through the interface next to the on/off button. The new temperature will be sent to the AC to adjust accordingly.

Figure 3.22 The air conditioner setting.

When the user turns on the air conditioner the current temperature setting will also be sent to ensure the set temperature is maintained. To toggle the air conditioner automation, users can click the setting button next to the air conditioner section topic, which will direct them to the air conditioner setting page. From there, users can enable or disable any air conditioner in the farm as desired. The user can also choose to view the list of air conditioners in the form of cards or tables by clicking the toggle button preset on the middle of the screen.

c. **Dehumidifier** – When the user activates the dehumidifier, multiple appliances will be activated, including the dehumidifier itself, fans, and a ventilator. These appliances are used by the farm owner to ventilate the room, as we aim to lower the humidity level as much as possible. Every appliance will be turned off after receiving the request from the user.
d. **Carbon dioxide tank** – Similarly, when the user activates the carbon dioxide tank, the attached relay will be activated and the valve will open, releasing the gas and filling the room. However, the valve is designed to release only a small amount of carbon dioxide to prevent any potential health hazards for both humans and plants. When the carbon dioxide level in the room is over the threshold level that the user has previously set, the carbon dioxide tank will stop releasing the gas automatically, but the user can also turn the tank off manually as well.
e. **Watering system** – After the user requests the activation of the watering system, the relay attached to the water pump will be activated. The water from the tank will get pumped through the hose, which is laid down networking over every plant in the farm to ensure a thorough watering of each plant. The hose is then connected to the water-dripping head, which is directed to the plant's stem. This will prevent overwatering while ensuring that every plant receives sufficient water. The watering system can be turned off manually through the same button.

2. **Scheduled tasks.**

The collaboration between the main backend service and the backend-worker service enables the scheduling feature. When the user creates a scheduled task in the farm settings page, the request is validated by the main backend service and then forwarded to the backend-worker service. The backend-worker service is responsible for executing the scheduled task at the specified time. The user will be able to override the command from the scheduled task, if the light is turned on from the scheduled task, the user can always manually turn off the light. There are three types of requests that can be made: creating a scheduled task, updating the time or content of a scheduled task, or canceling a scheduled task.

Each scheduled task in the backend-worker service corresponds to a record in the database, allowing the user to keep track of all scheduled tasks. However, the system prevents users from creating or updating scheduled tasks with overlapping time periods and prompts them accordingly.

Figure 3.23 The scheduled tasks setting.

a. **Farm light Scheduling** – To automate the farm lights, the user must specify the start and end time of the automation and ensure that there is no overlap between tasks. Each scheduled task for a farm light is associated with a specific light preset. The preset is a collection of settings for the light's strength, including natural light, ultraviolet light, and infrared light in each fixture throughout the farm. When the scheduled time arrives, these settings are sent to each light.

Users have the option to create new presets based on the current light settings in the farm by clicking the button from the light setting page. The user can choose

to create a new default preset that sets all light strengths to 50 percent from the preset list page as well.

To adjust the light strength of each light in the preset, users can access the preset setting page from the preset list page. The preset setting page is similar to the light setting page, with all the combinations of light strength for each light displayed, and the user can adjust each combination or enable/disable the automation of each light or choose to apply the setting to every light in the preset. On the preset setting page, the user can choose to change the name of the preset or delete the preset as well.

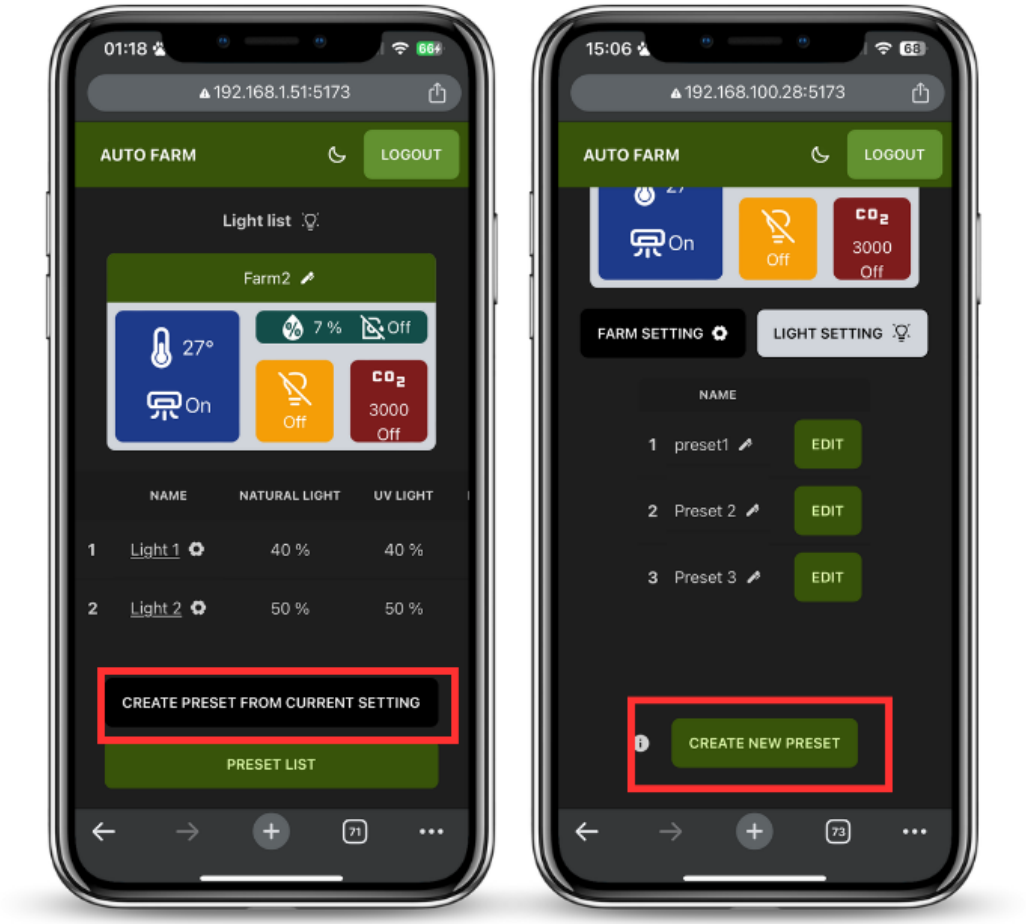

Figure 3.24 Process of creating a new preset.

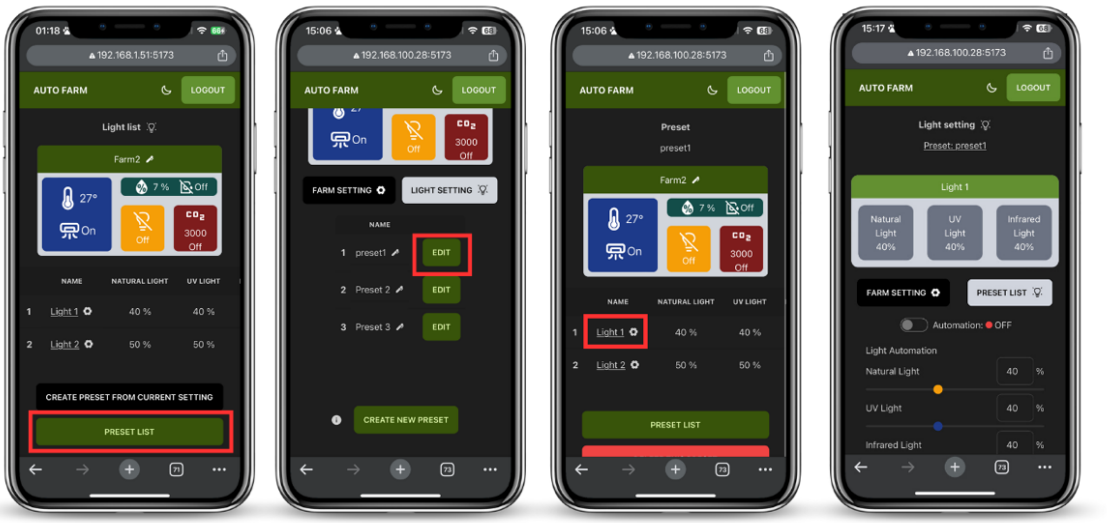

Figure 3.25 Adjusting light strength combination in the preset.

b. **Air conditioner scheduling** – Similar to the farm light scheduler task, the air conditioner scheduled tasks require users to set the start and end time of each task, along with the desired temperature. When the scheduled time arrives, an activation request including the temperature will be sent to each air conditioner in the farm that has automation enabled. Users can turn on and off the automation of each air conditioner in the air conditioner setting page, as shown in Figure 3.4.1.7. As with all scheduled tasks, users can also override the scheduled task by manually turning the air conditioner on or off.

c. **Watering scheduling** – The watering schedule works differently from the farm light and air conditioner automation. It only requires the user to set the start time for the scheduled task. When the scheduled time arrives, the activation request is sent to the relay attached to the water pump, which will start pumping water through the network of hoses around the farm. After 5 minutes, the backend-worker will send another activation request to the same relay to turn off the water pump. This is to prevent water from overflowing into the farm area. The watering automation is limited to a maximum of 5 minutes. Users can also turn off the watering automation here on the farm setting page. If the automation is set to be off, both manual and scheduled control will be disabled.

3. Set the threshold.

Maintaining appropriate levels of humidity and carbon dioxide is crucial in the farming process. To prevent any mold from growing on the plants, the humidity level must be kept as low as possible. Plants release water through a process called transpiration. During transpiration, water vapor escapes from small pores on the leaves called stomata. This process helps plants regulate their temperature, move nutrients and minerals, and maintain their shape and structure. Controlling the humidity level in the farm is crucial due to the increase in humidity caused by both the transpiration process and evaporation of any water sources in the farm.

Carbon dioxide is essential for the photosynthesis process of plants. Since the farm is located in a closed environment, the carbon dioxide level must be maintained during the time the light is on to ensure photosynthesis. Therefore, users need to set the threshold for these two factors to maintain optimal levels.

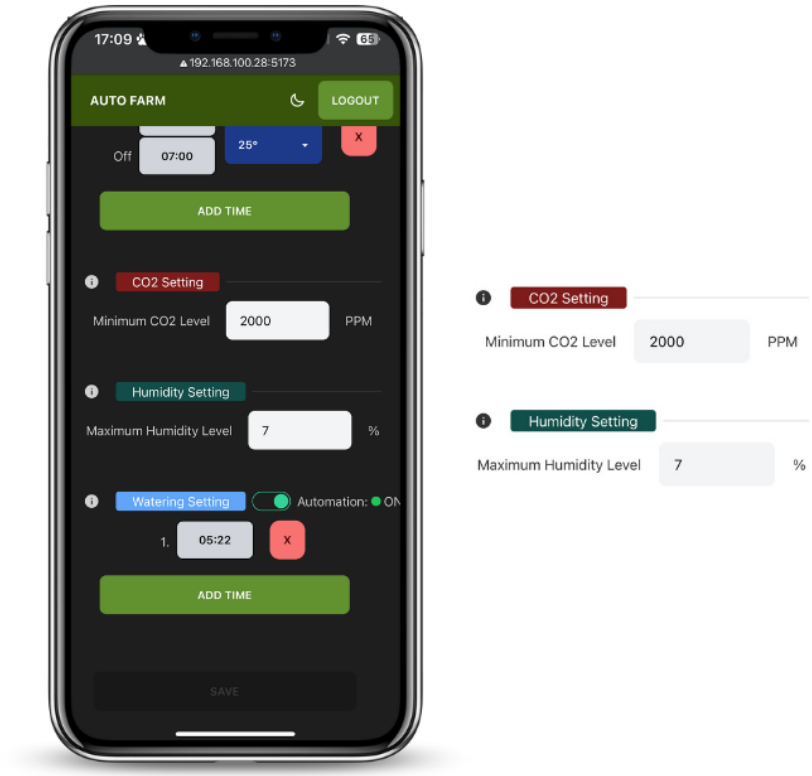

Figure 3.26 The threshold setting.

a. **Humidity level** – To prevent molds, it is important to keep the humidity level as low as possible. Users can set a threshold between 0 to 100 percent for humidity. When any sensors detect humidity higher than the threshold level, the dehumidifier is activated. The turn-off threshold is calculated as 70 percent of the set turn-on threshold. The dehumidifier will continue running until none of the sensors detect any humidity higher than the turn-off threshold. At that point, the system will send a request to turn off the dehumidifier to ensure that the humidity level remains within the set threshold. Suppose the user sets the humidity threshold to 50 percent. When any sensor in the farm detects a higher humidity level, the dehumidifier is triggered. It will continue to operate until none of the sensors detect a humidity level above 35 percent, at which point the dehumidifier will automatically turn off.

b. **Carbon dioxide level** – In order to maintain the proper level of carbon dioxide for photosynthesis, users can set a minimum threshold for the carbon dioxide level, which is measured in parts per million (ppm). Typically, in a closed environment with adequate ventilation, the carbon dioxide level ranges from 400-1000 ppm. If any of the sensors detect a carbon dioxide level lower than the threshold, the carbon dioxide tank will be activated and release carbon dioxide into the environment. The turn-off threshold is set at 120% of the turn-on threshold. If none of the sensors detect a carbon dioxide level lower than the turn-off threshold, the carbon dioxide tank will be turned off. For example, if the carbon dioxide threshold is set to 1000 ppm, and none of the sensors detect a value lower than 1200 ppm after the carbon dioxide is released, the tank will be turned off.

3.4.2 Hardware

According to the stakeholder requirements, the proposition of the project is to track all plant growth variables in the farm, maintain critical plant growth factors, and control all electrical appliances in the farm wirelessly. Additionally, all the hardware included in this project needs to be compatible with the stakeholder's default appliances.

To achieve the stakeholder requirements, the hardware design needs to be flexible enough to be implemented with the stakeholder's appliance. It should also be capable of controlling and maintaining plant growth variables in the farm.

Furthermore, this project requires the control of various plant growth factors, including light, temperature, humidity, carbon dioxide levels, and watering. Therefore, the project will involve the use of actuators to assist in maintaining these variables and controlling all electrical appliances in the farm.

Communication and control for all the sensors and actuators in the project will be handled by ESP32 through the MQTT protocol. Each module will consist of one ESP32 and the corresponding sensor or actuator based on its purpose.

To track the data from the sensors and control all the hardware modules, the responsibility falls on the MQTT2DB service. This service acts as a control center, assisting all the hardware modules. It will update the sensor data into the MySQL database as current data and store the sensor data in the Mongo Database as time series data for statistical analysis. Additionally, it will update the status of each hardware module after activation or deactivation.

The hardware modules in the project will consist of the following components: light controller module, air conditioner controller module, dehumidifier controller module, carbon dioxide controller module, sensor module, and watering module.

3.4.2.1 Sensor Module

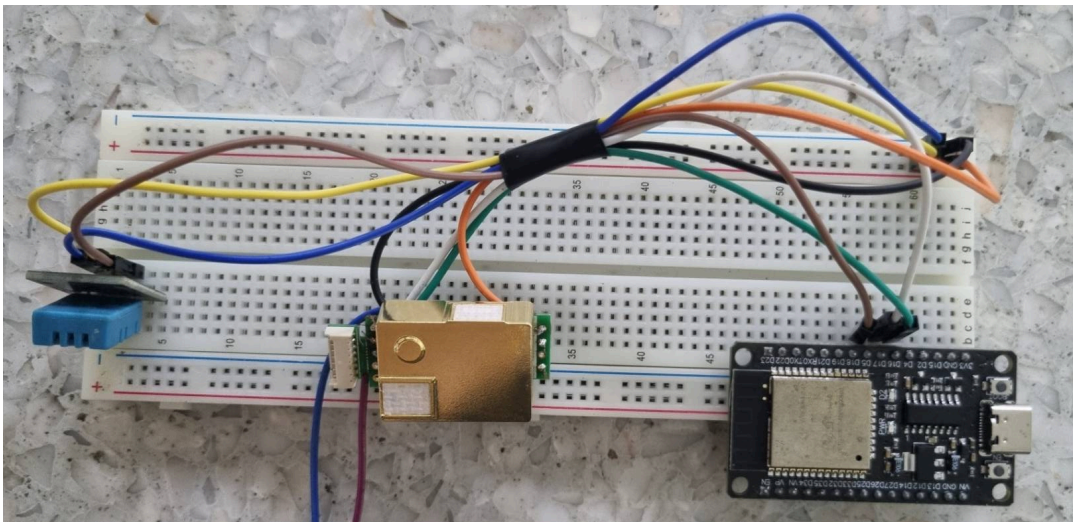

Figure 3.27 Sensor module

The main variables that need to be tracked are carbon dioxide level, temperature, and humidity. To monitor the farm's environmental status, the project will utilize three types of sensors: carbon dioxide sensor, temperature sensor, and humidity sensor. To ensure accuracy and correctness, the project will include four sensor modules, each consisting of one sensor of each type and an ESP32 as the controller. Each module will be installed in each corner of the farm to capture variables from every edge. The displayed data provided to the user will be the average of the data captured by these four sensors.

Every five minutes, the Sensor Module will send humidity level, carbon dioxide level, and temperature data to the MQTT2DB service. The MQTT2DB service will update this data in the MySQL database as current data and store it in the Mongo Database for statistical analysis as time series data.

The critical plant growth variables in this farm are humidity level and carbon dioxide level. To maintain these variables, the project will require additional hardware modules such as a dehumidifier and carbon dioxide emitter. These modules will help maintain the desired levels in case the density reaches critical thresholds. The user will be able to set the threshold values, and the ESP32 attached to the Sensor Module will

send MQTT requests to query the threshold data whenever it is updated in the database or during module initialization.

For the carbon dioxide level threshold, a minimum carbon dioxide value will be set, while for the humidity level threshold, a maximum humidity value will be set. These variables are highly sensitive to the plants in the farm, so each sensor will regularly check the threshold conditions every one to two seconds. When a variable value reaches the threshold, the Sensor Module will send an MQTT request to the MQTT2DB service to activate the corresponding Dehumidifier Module or Carbon Dioxide Emitter Controller Module.

Once the activation command is sent to the MQTT2DB, the MQTT2DB service will continuously monitor the value. If the triggered value is the humidity level, the Dehumidifier Module will remain activated until the value is below the threshold by thirty percent. Then, the MQTT2DB service will send an MQTT request to deactivate the Dehumidifier Module.

If the triggered value is the carbon dioxide level, the Carbon Dioxide Emitter Controller Module will be activated until the value exceeds the threshold by twenty percent. Then, the MQTT2DB service will send an MQTT request to deactivate the Carbon Dioxide Emitter Controller Module.

After each action, the Dehumidifier Module and the Carbon Dioxide Emitter Controller Module will send MQTT requests to the MQTT2DB service to update the module's status in the database.

3.4.2.2 Dehumidifier Controller Module

Figure 3.28 Dehumidifier and Fan

Figure 3.29 Air Ventilator

Humidity is a critical variable for plant growth in this project. To control and maintain humidity, several electrical appliances such as fans, dehumidifiers, and ventilators will be used. All the electrical appliances will be connected to a trailer plug attached to a relay. The ESP32 will control the relay via MQTT commands.

The Dehumidifier Module can be activated through three methods: user input, time automation, and threshold triggers. If the module is activated by the user or through time automation, it cannot be overridden by the system (triggered by threshold). However, if the module is activated due to a threshold trigger, the user can override the command to deactivate it.

If the Dehumidifier Module is activated due to a threshold trigger, it will automatically deactivate when the humidity level is below the threshold by thirty percent. The MQTT2DB service will send the command to deactivate the module.

After each MQTT request is sent to the Dehumidifier Module, if any actions are performed by the module, it will send an MQTT request to the MQTT2DB service to update the module's status in the database.

3.4.2.3 Carbon Dioxide Emitter Controller Module

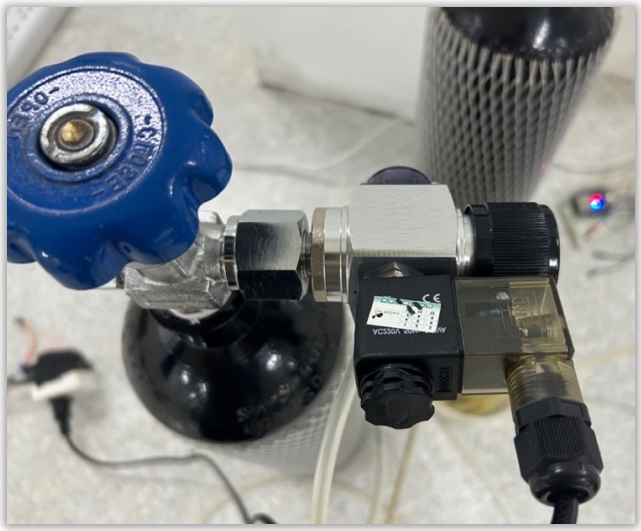

Figure 3.30 Carbon Dioxide Controller

Carbon dioxide is another critical variable for plant growth in this project. To control and maintain the carbon dioxide level in the farm, the stakeholder will use a carbon dioxide gas tank to increase the carbon dioxide concentration. The valve of the

carbon dioxide tank will be controlled by a solenoid connected to a relay. The relay will be controlled by the ESP32 via MQTT commands.

The Carbon Dioxide Emitter Controller Module can be activated through two methods: user input and threshold triggers. If the module is activated by the user, it cannot be overridden by the system (triggered by threshold). However, if the module is activated due to a threshold trigger, the user can override the command to deactivate it.

If the Carbon Dioxide Emitter Controller Module is activated due to a threshold trigger, it will automatically deactivate when the carbon dioxide level exceeds the threshold by thirty percent. The MQTT2DB service will send the command to deactivate the module.

After each MQTT request is sent to the Carbon Dioxide Emitter Controller Module, if any actions are performed by the module, it will send an MQTT request to the MQTT2DB service to update the module's status in the database.

3.4.2.4 Air Conditioner Controller Module:

The temperature in the farm will be controlled by the air conditioner, considering the closed environment. The air conditioner will be controlled using an infrared transmitter. The data signal from the air conditioner's remote control will be stored in the ESP32 and emitted to control the air conditioner based on the data received via MQTT.

Figure 3.31 Air Conditioner Controller

The Air Conditioner Controller Module can turn the air conditioner on, off, or adjust the temperature based on MQTT requests. The requests will be categorized into two methods: user input and threshold triggers. If the module is activated by the user, it cannot be overridden by the system (triggered by threshold). However, if the module is activated due to a threshold trigger, the user can override the command to deactivate it.

After each MQTT request is sent to the Air Conditioner Controller Module, if any actions are performed by the module, it will send an MQTT request to the MQTT2DB service to update the module's status in the database.

3.4.2.5 Light Controller Module:

The lighting system in the farm consists of specialized agricultural lights, including natural light, ultraviolet light, and infrared light. The light bar used in the project has already been provided by the stakeholders. Our responsibility is to control the lights without interfering with the existing circuit system. Each type of light in the light bar will be controlled individually.

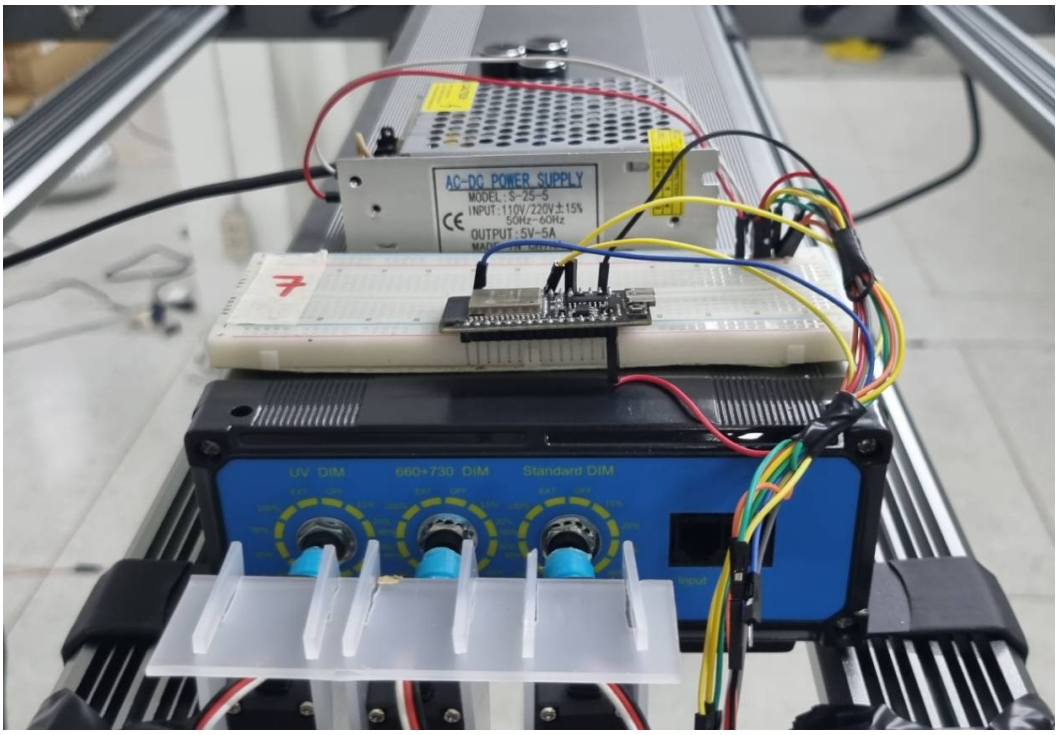

Figure 3.32 Light Controller

To achieve remote control, a 3D-printed connector will be used to connect the knob of the light bar to a servo motor. The servo motor will be responsible for turning the knob based on the desired light intensity commanded by the user through the ESP32 using the MQTT protocol.

The Light Controller Module can turn the lights on, and off, or adjust the intensity of each type of light in the light bar based on MQTT requests. The requests will be categorized into two methods: user input and time automation. After each MQTT request is sent to the Light Controller Module, if any actions are performed by the module, it will send an MQTT request to the MQTT2DB service to update the module's status in the database.

3.4.2.6 Watering Module:

The watering system in this farm operates by delivering water drop by using the dropper head. The Watering Module functions by pumping water from the water storage using a water pump motor and delivering it through mini hoses that are already attached to each plant pot in the farm. The water pump is controlled by a relay connected to the pump motor, which receives commands through MQTT requests via the ESP32.

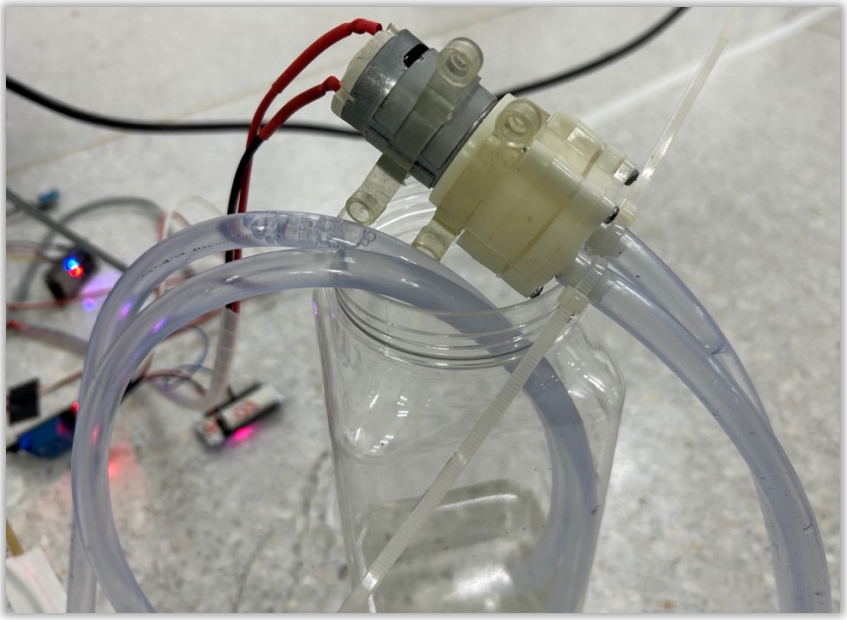

Figure 3.33 Watering system

The Watering Module can be activated through two methods: user input and time automation.

After each MQTT request is sent to the Watering Module, if any actions are performed by the module, it will send an MQTT request to the MQTT2DB service to update the module's status in the database.

The Watering Module is responsible for ensuring that plants receive the appropriate amount of water. Delivering water drops by using the dropper head will allow for precise control and efficient water usage. The module operates by pumping water from the storage source and distributing it through the mini hoses to each plant pot, ensuring that every plant receives an adequate water supply.

Manually or Time Automation	Threshold Trigger	Result
Activate	**Activate**	**Activate**
Activate	Deactivate	**Activate**
Deactivate	**Activate**	**Activate**
Deactivate	Deactivate	Deactivate

Table 3.1 Override Command Table

To prevent overriding commands. Once the hardware has been activated by the user, whether through manual control or time automation, it cannot be deactivated by the threshold system. However, when the hardware is activated by the threshold system, it can be deactivated by both the user and the threshold system.

3.4.3 Administration platform - Admin Portal

The administration of the Smart Farm service is handled by Admins, who use the Admin Portal as the designated platform. This platform was developed to facilitate efficient management of the service and comprises three primary sections: the user list, the farm list, and the ESP32 list.

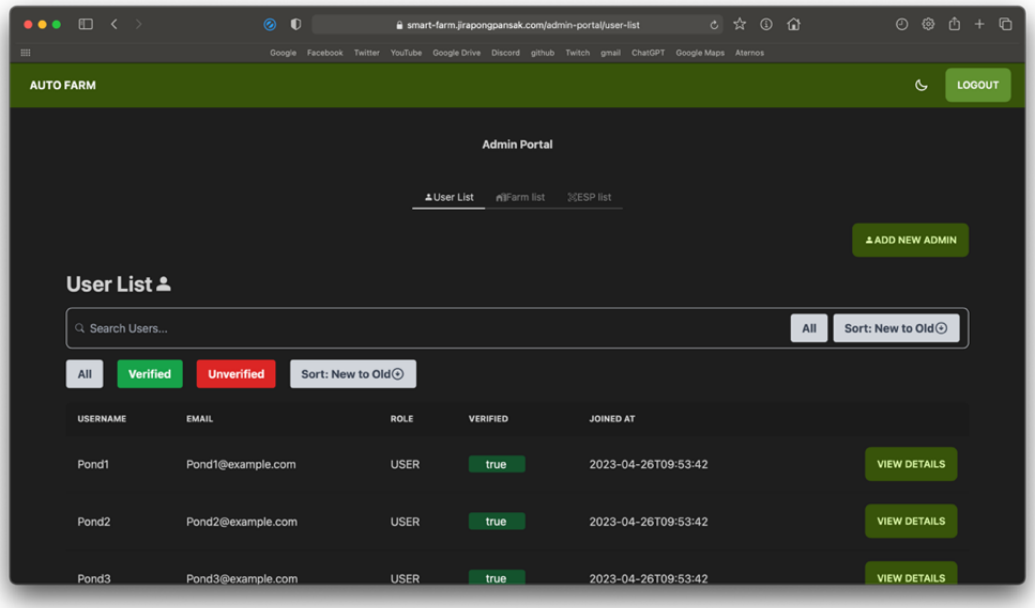

Figure 3.34 The User list of the Admin Portal page

To access this platform, the admin can log in through the same user login page. The account with the admin role will be redirected to the designated Admin Portal platform. To create a new admin account, the pre-existing admin account (or super admin account) must first register the new email with the system. To do so, the admin will have to click the 'Add new admin' button on the top of the user list page in the Admin portal. There will be a prompt show up for the user to enter the email.

Once finished, the system then sends an email to the registered address with the link that redirected the user to the registration page. The link can only be used once. If the registration is completed, the link will be put to expire, and the user can no longer access the registration page again.

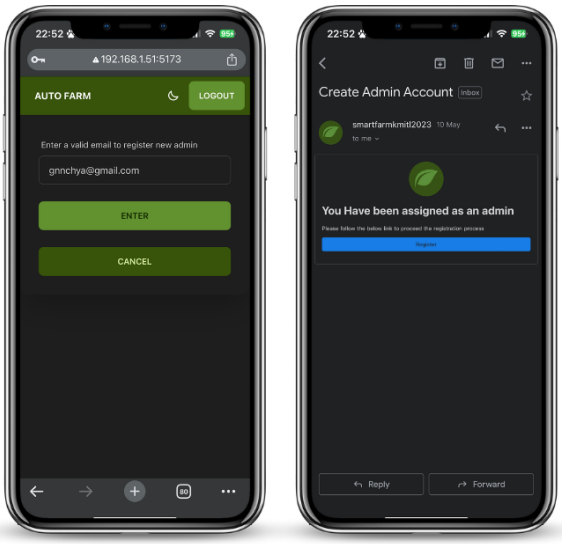

Figure 3.35 Email registration prompt and received Email.

The new admin can then set their username and password to complete the registration process. The protocol is the same as the user registration. The username must not duplicate in the system and must be at least 3 characters long. Password must be at least 6 characters long and must contain at least 1 capital letter and 1 numerical character. Once finished, the confirmation email will be sent to the same email and the new admin can log in to the system using their registered username and password.

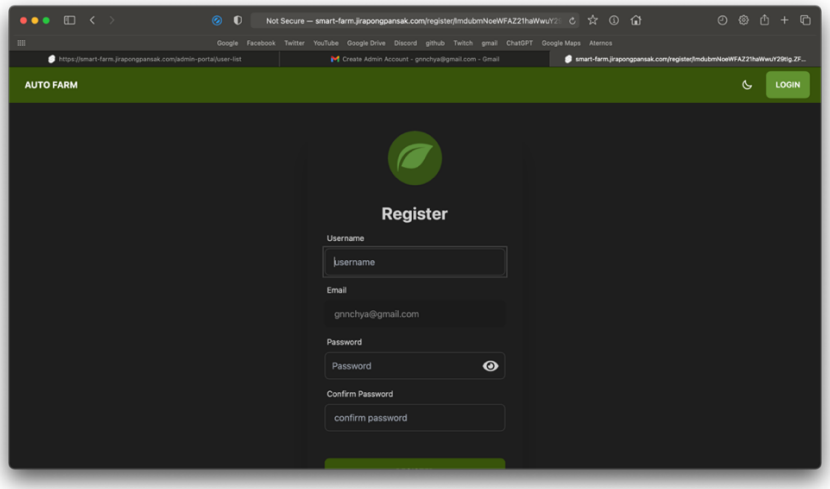

Figure 3.36 The Admin registration page

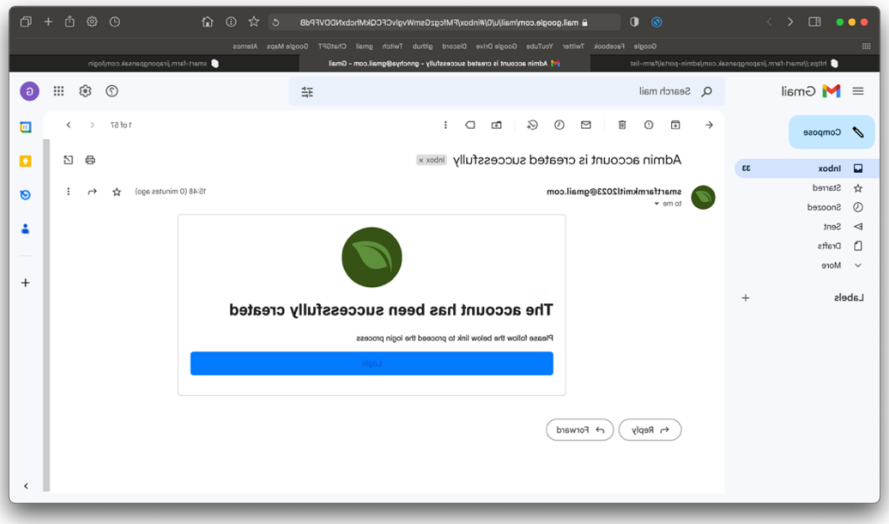

Figure 3.37 The Admin registration confirmation email

The user list section is where all the users that register to the system are displayed. The basic information of that user is listed including username, email, verification status, and the registration timestamp. This allows the admin to address the problem when they get contacted. The list can be sorted from the newest to oldest and vice versa and can be listed only the verified or unverified accounts. Moreover, users can also search for the account's username or email as well. To list all the farms that the user owns, the user can click the 'view detail' button.

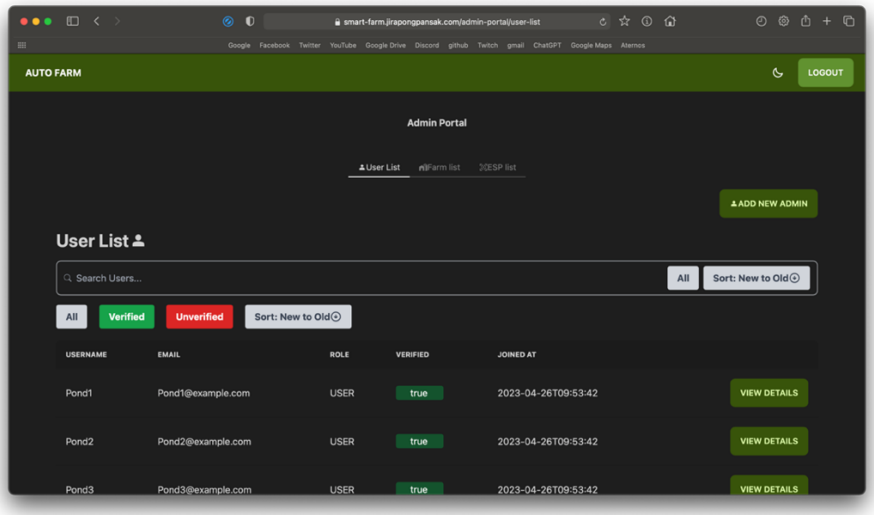

Figure 3.38 The User list of the Admin Portal page

Similar to the list of users, the Admin Portal platform also allows admins to sort the farm list in ascending or descending order based on the date of creation. Users can also search for a specific farm by name using the search feature provided. The farm key is a crucial component of this page, as it is required for users to add a post-setup farm to their account. During the installation process, admins may provide the farm key to the farm owner, who can then share it with authorized users to enable them to connect to the farm. The farm key can be shared with authorized users again if requested.

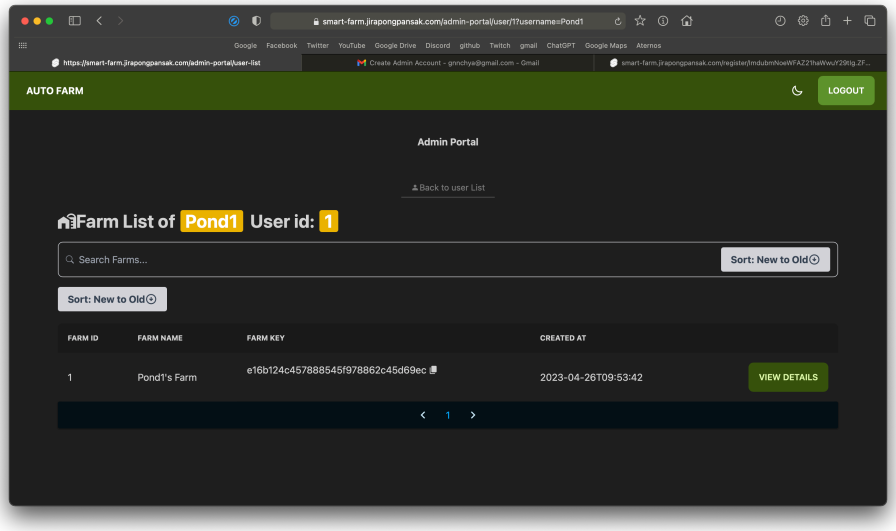

Figure 3.39 The listing of farms owned by a certain account.

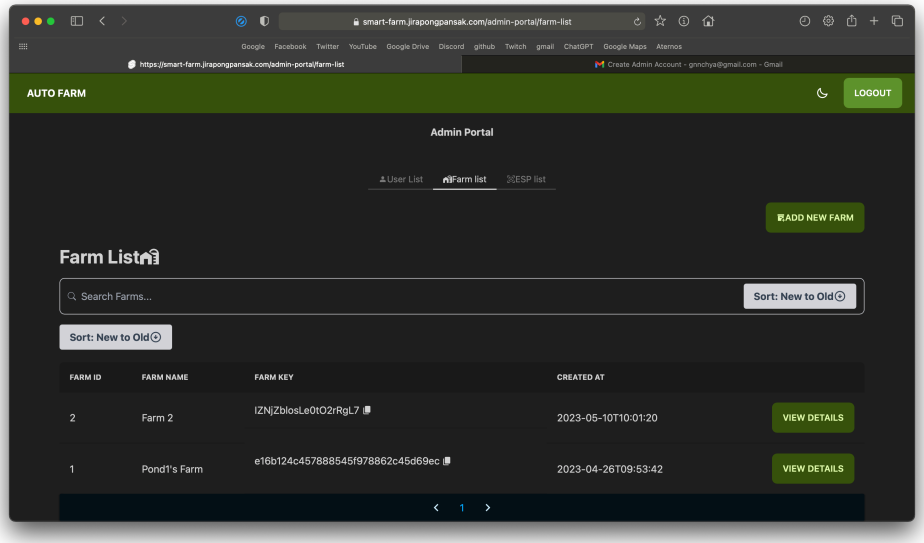

Figure 3.40 Farm list section

The second section is similar to the listing of the farm owned by the user. But the list includes all the farms from every user. The sorting and searching are also identical. But on this page, the admin can create a new farm. The farm key will be automatically generated. This will be used in the case of new farm installation. The farm key will then be passed down to the farm owner once the installation is completed.

To configure the installation of ESPs for each sensor and actuator, the admin can navigate to the farm list page and click on the "view detail" button to access detailed information about each farm. This action redirects the user to a list of ESP32s associated with the selected farm, each with a unique ID linked to specific sensors or actuators. The page categorizes all connected sensors and actuators by type of device, and users can sort the list in ascending or descending order based on the ESP32 ID. Additionally, users can search for a specific ESP32 ID using the search function provided.

This page is essential for the installation process, as admins will use it to link ESP32s with specific devices to ensure proper system functionality. Admins can add more devices to the system and subsequently assign them to an available ESP32.

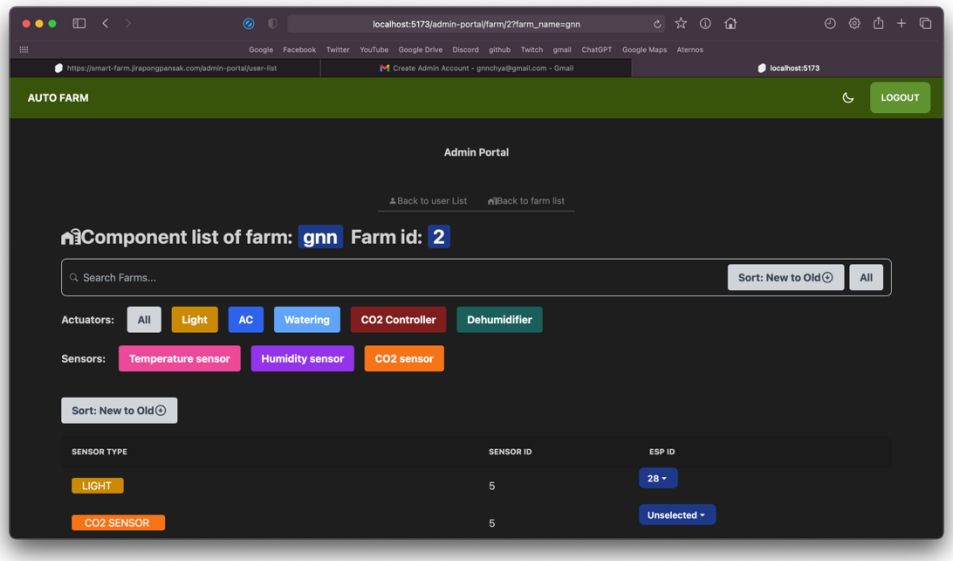

Figure 3.41 ESP list based on a farm.

The final section of the Admin Portal is dedicated to the list of ESP32s. Here, all ESP32s are displayed along with relevant metadata, including their usage status (in-

use or not in-use) and availability. The page also provides sorting functionality based on the creation date in both ascending and descending order. ESP32s can be categorized as in-use or not in-use, and the page includes a search function based on ESP32 ID. Admins can create a new ESP32 by selecting the "create new ESP" button, which generates a new ESP32 labeled as "Not in use."

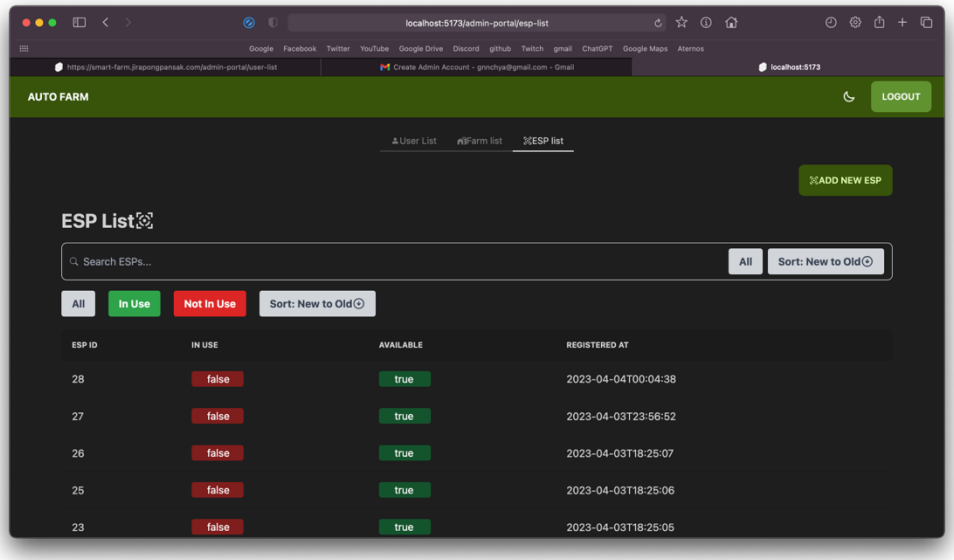

Figure 3.42 ESP list page in Admin Portal.

3.5 Chapter Summary

This chapter described the high-level requirements and design of a system able to maintain, monitor and automate the farm. The chapter started by describing how the system works at a high level. The system design was then discussed in section 3.2.1 followed by discussion of the user platform in 3.2.2

Furthermore, we cover the interesting problems in 3.3, this covers the problems that we faced in the implementation of the project. Finally, we discuss the proposed implementation of the system in 3.4.

CHAPTER 4
EXPERIMENTAL RESULT

4.1 Introduction

Chapters 3 and 4 described the design and implementation of the farm automation system, a system that automates farming. In this chapter, we present a testing method and its results that show how the system works. The chapter is organized as follows: Section 4.2.1 introduces manual control and describes the expected outcome. Next, section 4.2.2 presents the scheduling system, Section 4.2.3 presents the threshold system and finally, 4.2.4 presents the monitoring solution.

The results of the tests are summarized in section 4.3 before the solution is evaluated in the following paragraphs.

4.2 Testing Summary

This section provides an overview of the testing process and methods used to ensure the proper functionality of our system. The testing phase encompasses four main features: Manual Control, Scheduling, Threshold, and Monitoring. Each feature was thoroughly tested to validate its effectiveness and reliability.

4.2.1 Manual control

As highlighted throughout the report, the system offers users the ability to manually control various actuators via the web application. This includes managing farm lights, air conditioners, dehumidifiers, carbon dioxide tanks, and the watering system. Users have full control over these components, allowing them to adjust settings and operations according to their specific requirements and preferences.

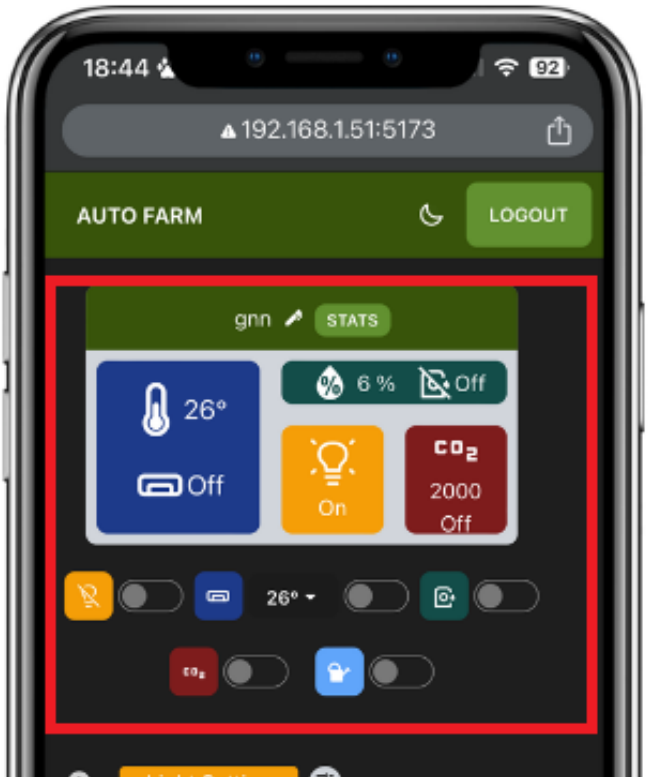

Figure 4.1 Manual control panel

The testing process involved verifying the system's response by toggling the actuators on and off. Additionally, we experimented with adjusting parameters to ensure their functionality. For instance, we modified the light intensity in the request sent to the actuators, allowing us to verify if the changes were accurately reflected. This type of testing not only ensured proper transmission and updating of user settings but also helped evaluate the precision of components like the servo, confirming if the desired degree of rotation was achieved.

4.2.1.1 Manual control – Light

We performed a test on the manual control of the farm light. Upon turning on the light, it emitted a specific combination of light strengths that had been previously set: 50% natural light, 50% UV light, and 50% infrared light. This action is carried out by the turning servos on the light dials. You can observe this result in the accompanying picture. Subsequently, we proceeded to turn off the farm light and check the result.

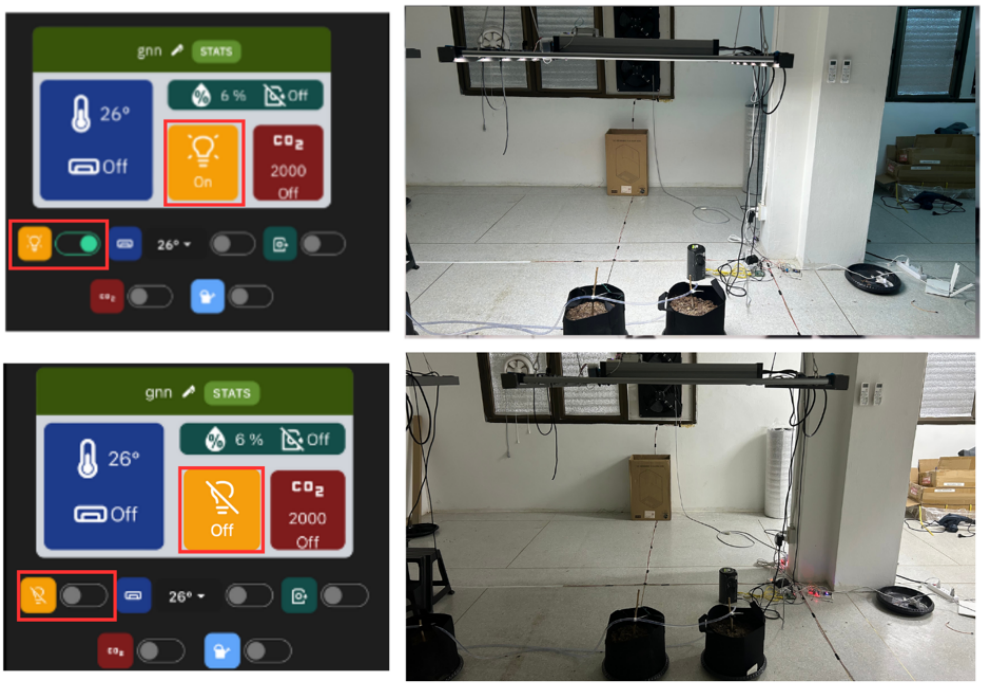

Figure 4.2 Turning on/off the farm light.

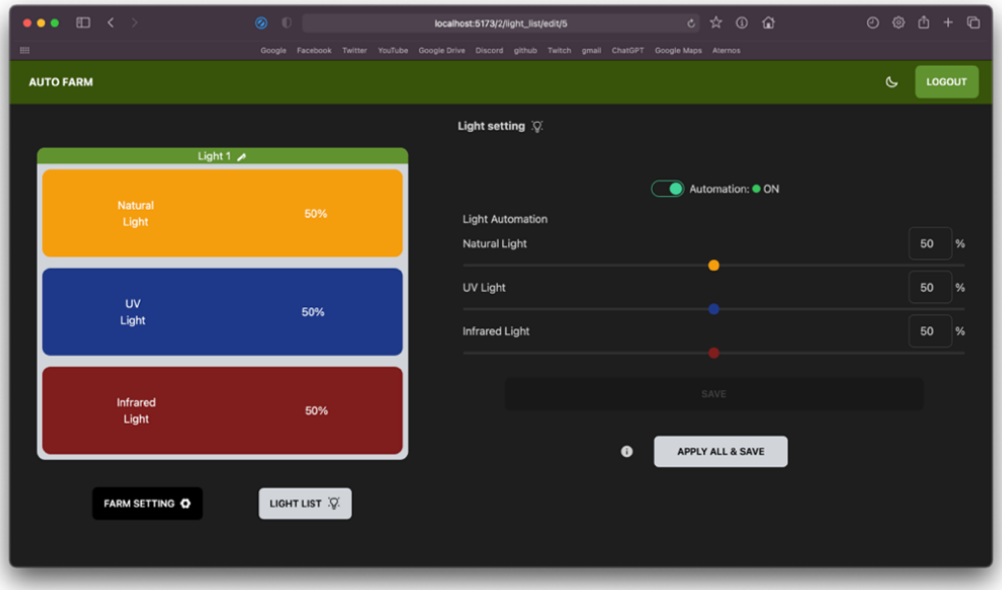

Figure 4.3 Light strength setting.

After we test the I/O functionality of the farm light, we then proceed with the light strength combination testing. To do that, we try adjusting the light strength of each dial and check if the actuators react accordingly. We test by these 3 light combinations.

1. **50 percent Infrared light** – In this particular configuration, the infrared light intensity is set to 50%, while the other components are set to 0%. Once the settings are saved, we proceed to turn on the light and examine the outcome. The resulting image clearly displays the infrared light being activated at a strength of 50%.

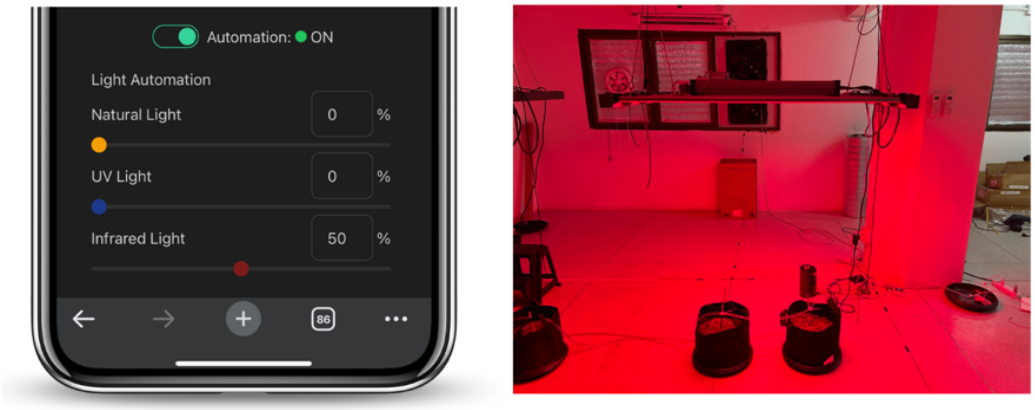

Figure 4.4 Turn on the Infrared light.

2. **50 percent Infrared light** – We repeated the same test, but this time the setting was adjusted to 50% UV light, while the remaining components were set to 0%. The outcome of this test is illustrated in the accompanying figure.

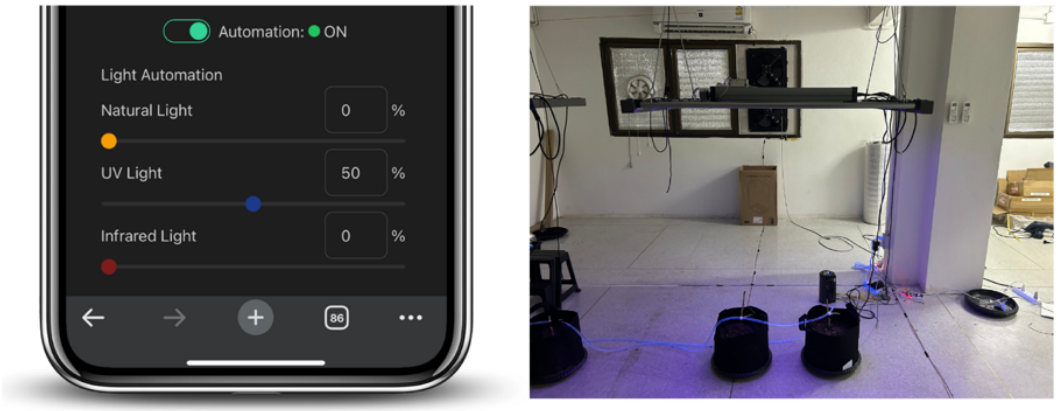

Figure 4.5 Turn on the Ultraviolet light.

3. **50 percent Natural light** –For the final test, we repeated the process using natural light as the setting. The outcome of this test is displayed in the figure below.

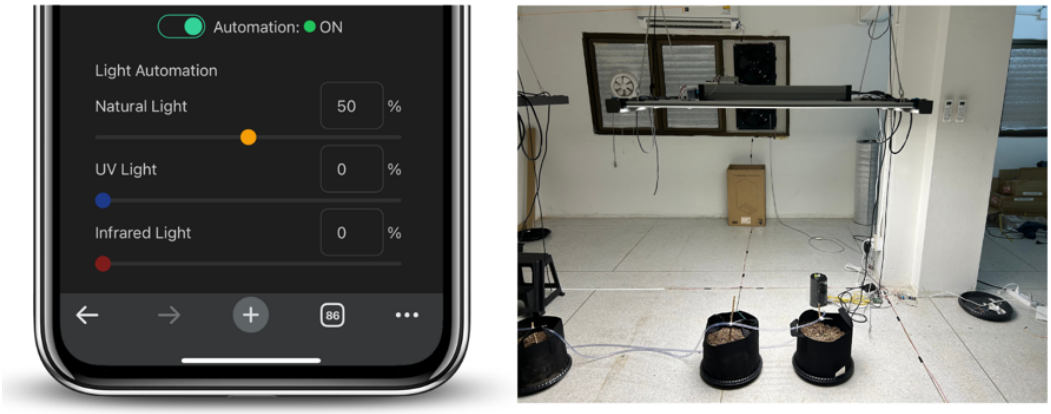

Figure 4.6 Turn on the Natural light.

To broaden the scope of our testing, we explored different scenarios by adjusting the light strength settings while the light was being turned on. We expected the light to adapt and reflect the changes immediately upon clicking the save button. After we turn on all the light to 50 percent, we decrease the light strength of the Natural light and the Ultraviolet light to 0 percent and observe the result. The outcome of this test can be observed in the provided figure, which demonstrates the desired behavior of the system.

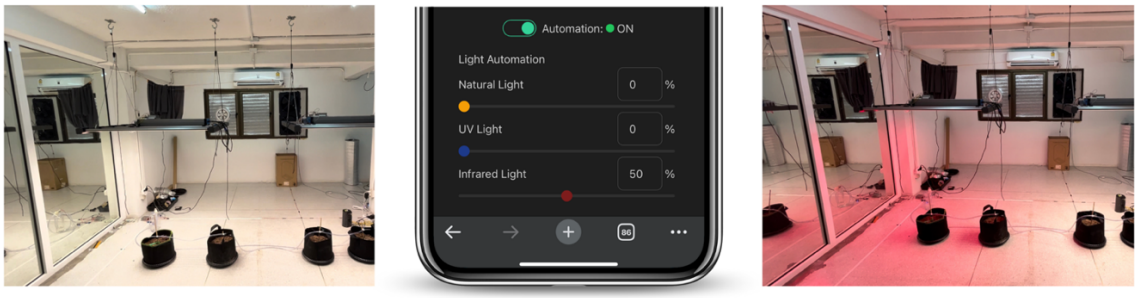

Figure 4.7 Change the light strength setting during the light is on.

To accommodate user preferences, our system allows users to customize the automation settings for individual lights. In situations where users do not want specific

lights to be turned on when using the manual control button, they can navigate to the light setting page and disable the automation feature for those lights. During our test, we followed the mentioned procedure, manually turning on all lights in the farm while having the automation turned off for light number 1. As expected, the non-automated lights remained off, as depicted in the provided figure.

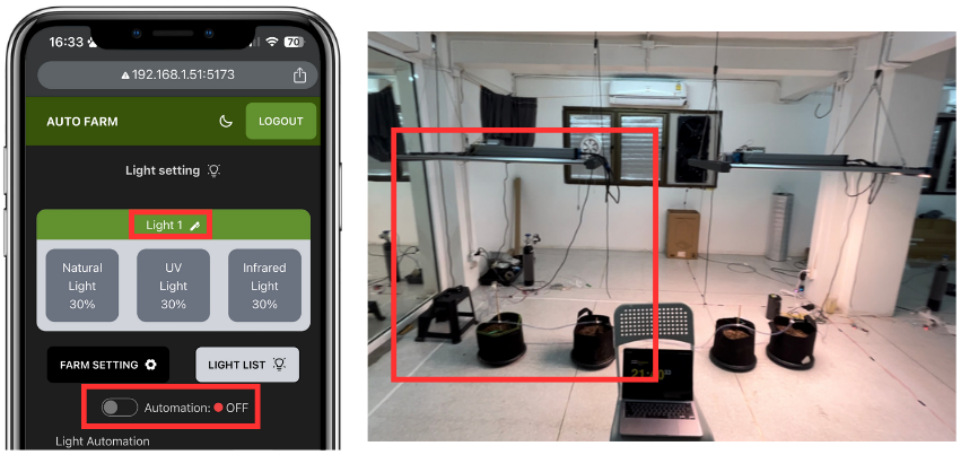

Figure 4.8 Turn off the light automation.

4.2.1.2 Manual control – Air-conditioner

In the case of manual control over the air conditioner, we conducted a comprehensive test that involved turning the AC on and off, as well as adjusting the temperature settings. When the user activates the infrared transmitter from the control panel, a signal is sent to the air conditioner, resulting in an audible beep and the actual activation of the AC unit. However, it is important to note that the air conditioner does not display a temperature indicator. To test the temperature adjustment functionality, we relied on the subsequent beep sound that accompanied changes made to the temperature settings in the interface. The temperature readings obtained from the temperature sensor were used to estimate the effectiveness of the temperature control. However, it is important to note that the temperature measured by each temperature sensor in the room may not precisely match the temperature set on the air conditioner. This discrepancy is due to the size of the room and the placement of the sensors in different corners. However, we observed that the temperature readings did exhibit

noticeable changes in response to the temperature adjustments made through the interface. This correspondence between the interface settings and the observed temperature changes allows us to infer the effectiveness of the temperature control feature.

Figure 4.9 Turn the air conditioner on and off.

After verifying the I/O of the air conditioner, we conducted a temperature control test. Users can adjust the temperature through the interface by selecting their desired setting from the dropdown menu. The air conditioner promptly responded with an audible beep, confirming the receipt of the command. We monitored the temperature sensors to validate the changes, noting a consistent correlation between the interface adjustments and the detected temperature fluctuations. While individual sensor readings may vary due to room size and placement, the overall alignment between user input and temperature changes demonstrates the reliable functionality of the temperature control feature.

Figure 4.10 Changing the temperature of the air conditioner.

Our system extends the automation control feature to the air conditioning units as well. Users have the flexibility to enable or disable the automation setting for each air conditioner independently. During our test, we specifically turned off the automation for air conditioner number 1 and proceeded to manually turn it on. As expected, the air conditioner did not activate, demonstrating the successful integration of the automation settings with manual control. This test confirms that when automation is disabled for a specific air conditioner, it remains unaffected by manual commands, providing users with precise control over their desired cooling preferences. The result can be seen in the following figure.

Figure 4.11 Turning off the air conditioner automation.

4.2.1.3 Manual control – Dehumidifier

We conducted a comprehensive test on the system controlling the dehumidifier, fan, and ventilator. By pressing the green toggle button, the system promptly executed the corresponding command. In the image below, you can observe the successful operation of the dehumidifier system. The fans, ventilator, and dehumidifier are all functioning as intended.

To prepare for the test, we initially powered on all the appliances and connected them to the electrical socket linked to the relay. This setup ensures that when a command is received, the relay is triggered, allowing electricity to flow into the respective appliance, enabling its operation.

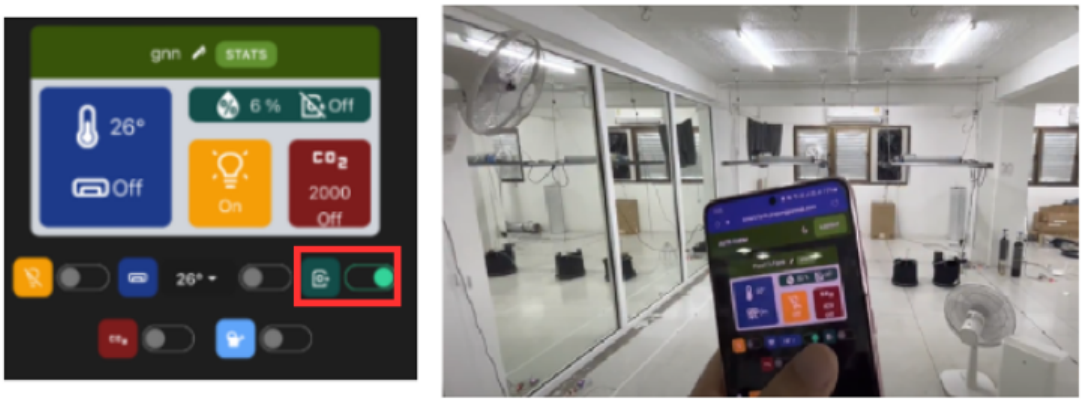

Figure 4.12 Turn on the dehumidifier on the interface.

Figure 4.13 The dehumidifier system is on.

4.2.1.4 Manual control – CO2 controller

For the Carbon Dioxide emitter controller, the user can control it by toggling a button on the web application. Upon receiving the request, the system commands the Carbon Dioxide emitter controller to activate or deactivate the relay that controls the solenoid attached to the Carbon Dioxide gas tank. When the Carbon Dioxide emitter is activated, Carbon Dioxide gas is released into the farm, causing the light indicator on the electric solenoid to appear, as depicted in the figure. Please note that the Carbon Dioxide concentration level on the web application may not immediately increase due to factors such as the size of the farm and the pressure of the Carbon Dioxide tank. The gradual increase in Carbon Dioxide levels within the farm can be monitored through the Carbon Dioxide sensors positioned throughout the room. These sensors provide regular updates to the user, allowing them to observe the progressive changes over time.

On the other hand, when the Carbon Dioxide emitter is deactivated, the emission of Carbon Dioxide gas from the tank ceases. As a result, the light indicator on the electric solenoid disappears, following the pattern shown in the figure.

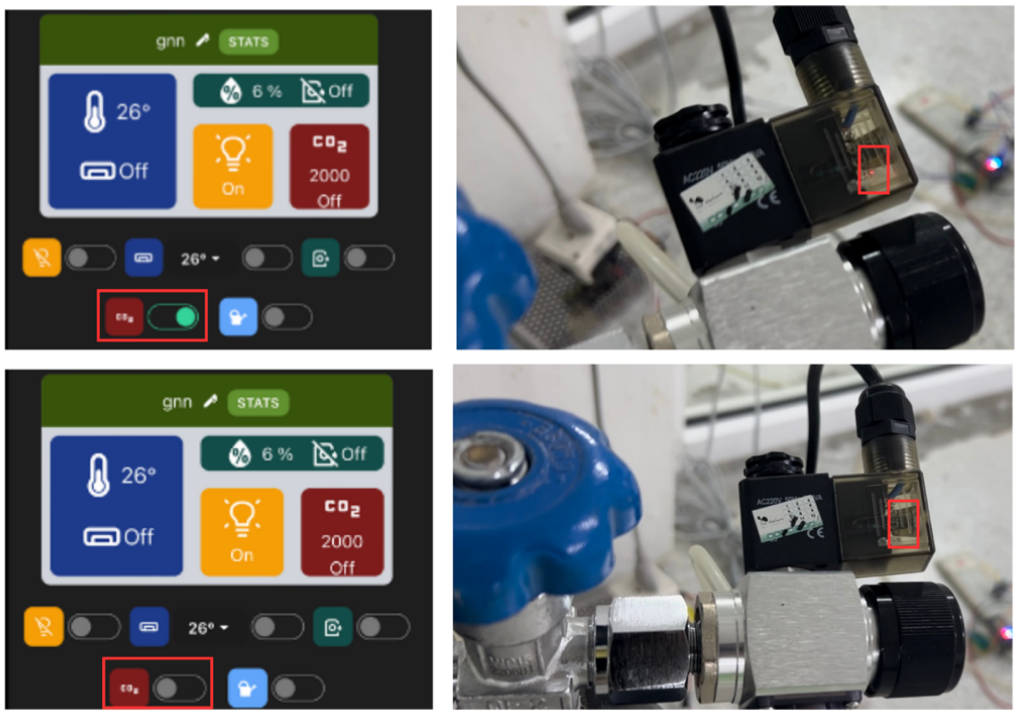

Figure 4.14 Turn the Carbon Dioxide on and off

4.2.1.5 Manual control – Watering controller

For the watering controller, the user can control the watering system by toggling a button on the web application. Upon receiving the request, the system sends the command to the watering controller. The watering controller then executes the command to activate or deactivate the electric solenoid attached to the watering pump.

When the watering system is activated, the watering pump starts pumping water from the container. The water is delivered through the hose and is distributed using a dripping head to water the plants. The dripping head allows controlled and precise water delivery to each plant.

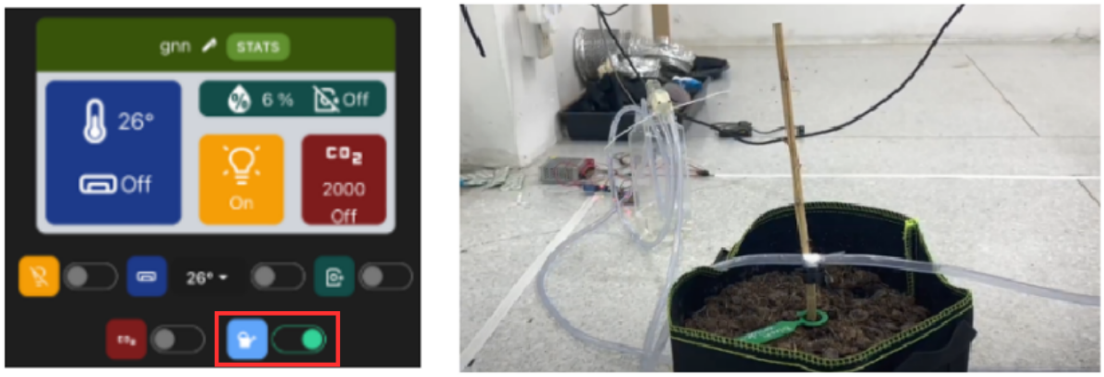

Figure 4.15 Watering slide button and turning on the watering system.

4.2.2 Scheduler

In addition to manual control, our system offers a scheduling feature that enables users to schedule tasks for lighting, air conditioning, and watering. Users can define the start and end times of the tasks, as well as specify parameters such as air conditioner temperature or light presets. To validate the scheduling functionality, we conducted tests by setting the task end time to be 5 minutes after the start time, as the scheduling allows for time intervals in multiples of 5 minutes. This section will provide an overview of the conducted tests in this regard.

4.2.2.1 Scheduler - Farm

In order to test the farm light scheduling functionality, we needed to set the start time and end time for the task, along with selecting a specific farm light preset. As previously explained, a light preset consists of a combination of light strengths for each light fixture, along with automation settings that determine whether the light should be automated during task execution. To ensure the accuracy and correctness of this feature, we conducted several tests to verify both the timing of task execution and the parameters we had set.

1. **Time and parameter test**

   As part of our testing, we configured the task to start at 21:20 and end at 21:25, using Preset 1. Preset 1 entails automating all lights and setting the light strength combination to 50 percent for light number 1 and 30 percent for light number 2. This test aimed to verify the execution of the task at the designated time and ensure the accuracy of the specified parameters.

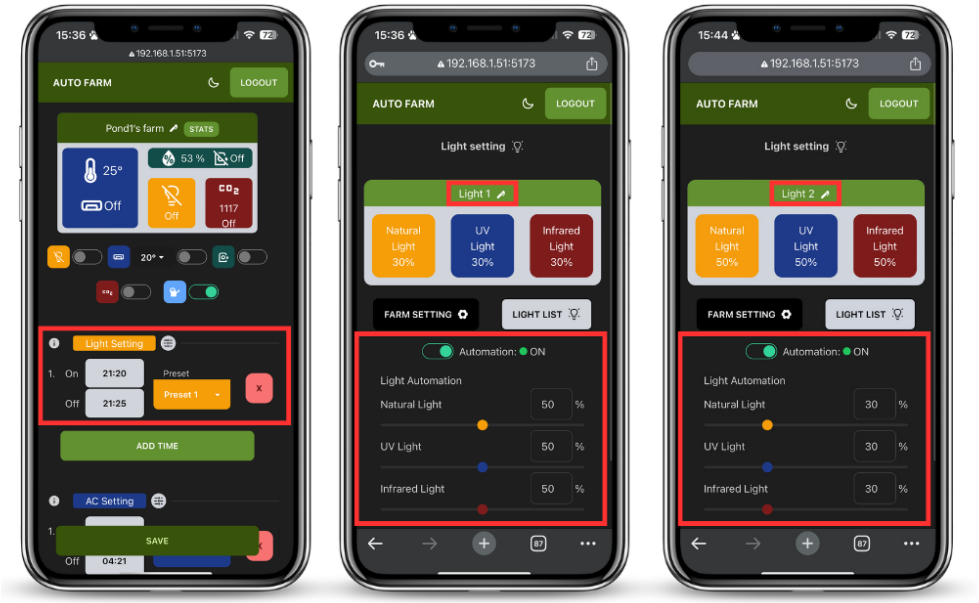

Figure 4.16 Light scheduling and the light strength combination

During the test, at 21:20, the scheduled task was executed as expected. The lights were turned on, with light 1 set at 30 percent and light 2 at 50 percent, matching the assigned parameters. Subsequently, at 21:25, the lights were successfully turned off.

This test confirms the proper functioning of the time scheduler, as well as the successful assignment and execution of parameters for scheduled tasks. The result can be observed in the following figure (Figure 4.17).

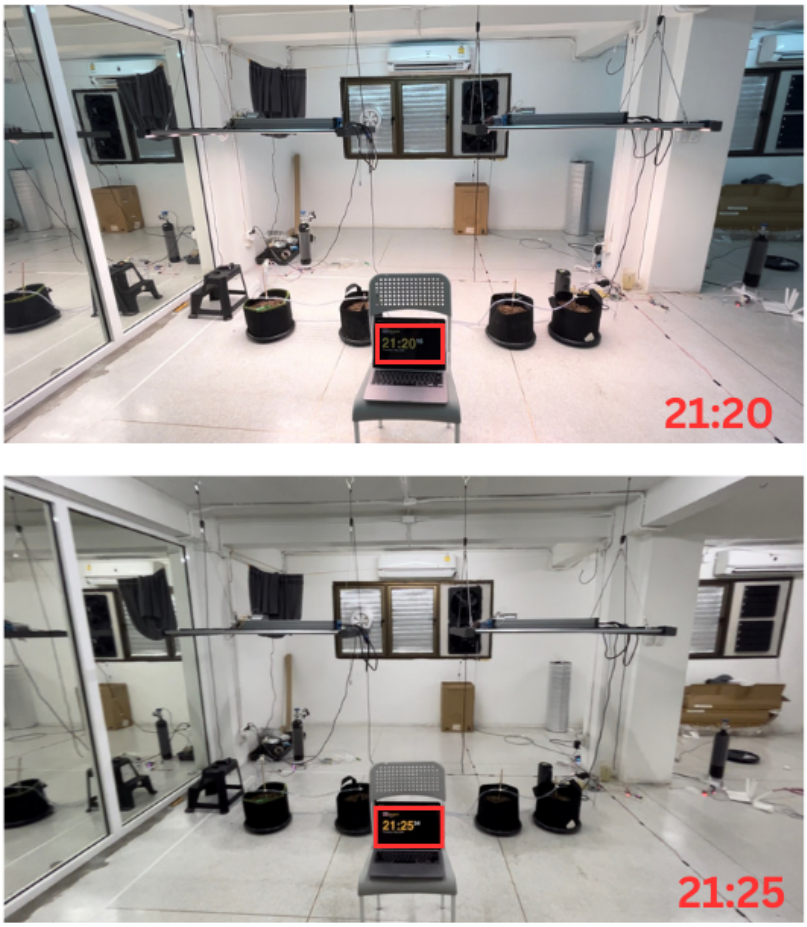

Figure 4.17 The scheduled task is executed at the specified time.

2. **Farm light automation setting test.**

In this test, we focused on verifying the automation setting for each light. For the updated task, light preset 1 was modified to have a light strength combination of 50 percent for light number 1, while the automation setting was turned off. Light number 2 retained the same settings, with all lights set at 30 percent. The task was scheduled to start at 21:40 and end at 21:45. During the scheduled task initiation at 21:40, the expected result was that light number 1 would remain off, as its automation setting was disabled. Conversely, light number 2 was expected to

respond according to its settings. By observing the system's behavior during this test, we aimed to confirm that the automation settings were correctly applied to each light during the scheduled task execution.

Figure 4.18 The scheduled task with disabling automation setting.

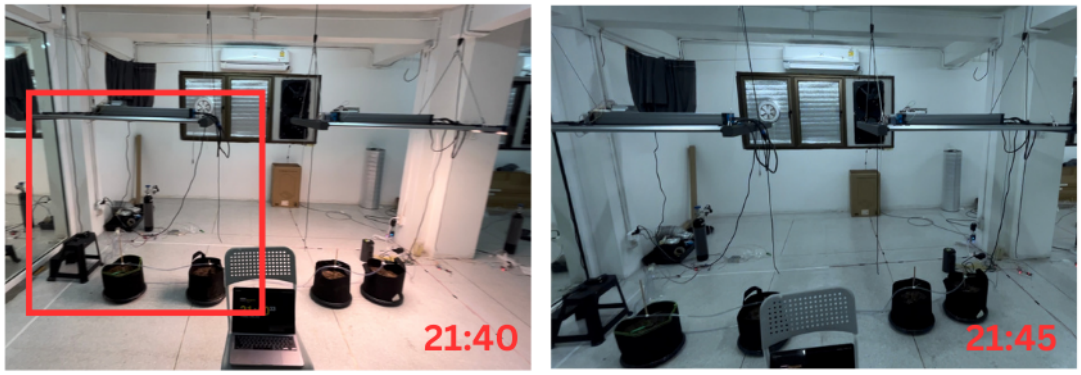

Figure 4.19 The result from turning off the light automation of light number 1.

3. **Override the scheduled task.**

In our system, users could manually override commands from scheduled tasks. For example, if the light is turned on by the scheduler, users can manually turn it off. When the end time of the scheduled task is reached, no further action will be taken, and the light will remain off. However, if the user manually turns the light off and then back

on within the duration of the scheduled task, the light will adhere to the user's manual setting rather than the preset. Nevertheless, once the end time of the scheduled task is reached, the light will be automatically turned off. This feature provides flexibility and control to the user, allowing them to customize the lighting according to their preferences.

In this test scenario, we build upon the previous scenarios to explore the interaction between manual commands and scheduled tasks. At 21:40, when the scheduled task activates the light, we manually turn it off through the user interface, and indeed, the light is successfully turned off. Subsequently, when we attempt to turn the light back on, it responds accordingly. However, when the scheduled task's end time is reached, the system automatically turns off the light once again, as depicted in Figure 4.16. This test demonstrates the seamless integration of manual commands and scheduled tasks, providing users with flexibility and control over the lighting system.

4.2.2.2 Scheduler – Air Conditioner

The scheduler for the air conditioner in the smart farm system allows users to set specific activation times and temperature levels. Users can configure multiple scheduled tasks for the air conditioner throughout the day and adjust the temperature level for each task.

Since each farm can have multiple air conditioners installed, the user has the ability to toggle the automation setting for each air conditioner individually in the web application. This means they can choose which air conditioner in the farm to be activated during the scheduled tasks based on their preferences.

1. **Time and parameter test**

To conduct the test, the first scheduler is set for 21:20 (9:20 PM) to activate the air conditioner and set the temperature to 25 degrees Celsius. In this case, all the automation of the air conditioner is turned on so when the clock reaches 21:20, the air conditioner will activate and adjust the temperature level to 25 degrees Celsius. It will remain active until 21:25 (9:25 PM) when it will automatically deactivate.

Figure 4.20 The scheduled task with disabling automation setting.

2. **Air conditioner automation setting test.**

However, if the user toggles off the air conditioner automation in the air conditioner setting page, that certain air conditioner will not activate or deactivate during the scheduled task. To conduct the test, the automation setting for air conditioner number 1 is turned off. After that, the task was scheduled to turn on the air conditioner at 21:40 and turn it off at 21:45. As a result, during the scheduled task at 21:40 and 21:45, air conditioner number 1 did not turn on or off, aligning with the automation setting. This behavior is consistent with the manual control scenario. The figure 4.9 illustrates the outcome of this test, confirming the successful integration of automation settings with the air conditioning units.

3. **Override the scheduled task.**

Similar to the farm light, manual control of an air conditioner can override the command from the scheduled task. From test number 1, when the air conditioner is turned on at 21:20. Test number 3 can be conducted by turning the air conditioner off manually. The air conditioner reacted accordingly, indicating that the system works properly.

4.2.2.3 Scheduler – Watering controller

The watering system scheduler allows users to set specific times for activating the watering system to water their plants. Each time the watering system is activated, it will water the plants for a duration of 5 minutes, delivering approximately 200 ml of water. Users can set multiple watering times throughout the day and can use the toggle button in the web application to activate or deactivate these scheduled watering events. The same set of tests were conducted to test the functionality.

1. **Time scheduling test**

To conduct the test, the first scheduler is set for 21:20 (9:20 PM) with the automation set to be on. In this case, the watering system will begin watering the plants at precisely 21:20 and continue until 21:25 (9:25 PM) as shown in the provided figure.

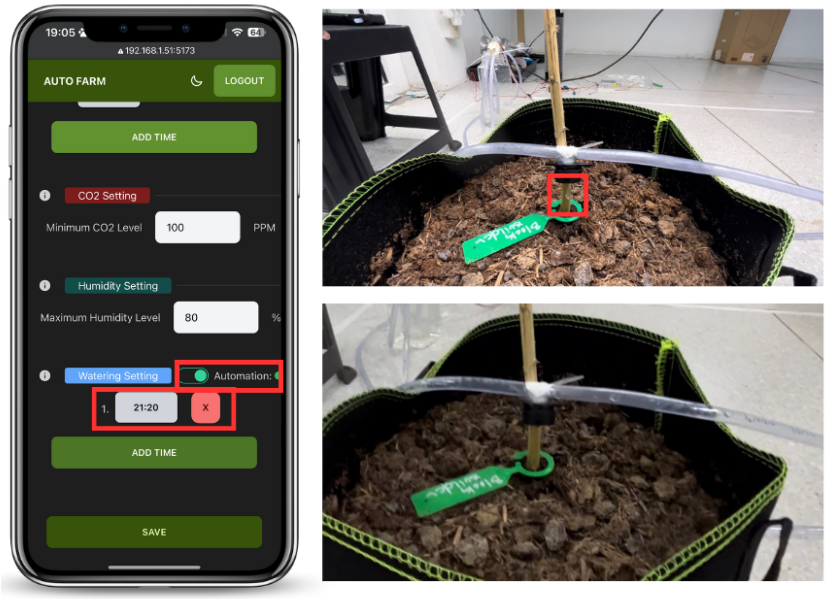

Figure 4.21 The scheduled task for watering

2. **Watering system automation setting test.**

But if the user chooses to toggle off the watering system automation in the web application, the watering system in the farm will not be activated according to the predefined time automation set by the user in the web application until the scheduler is toggled on again. The test is conducted by setting the schedule at 21:20 with the

automation set to be off. As a result, when it is at 21:20, the watering system is not activated. The automation setting can be seen in the provided figure.

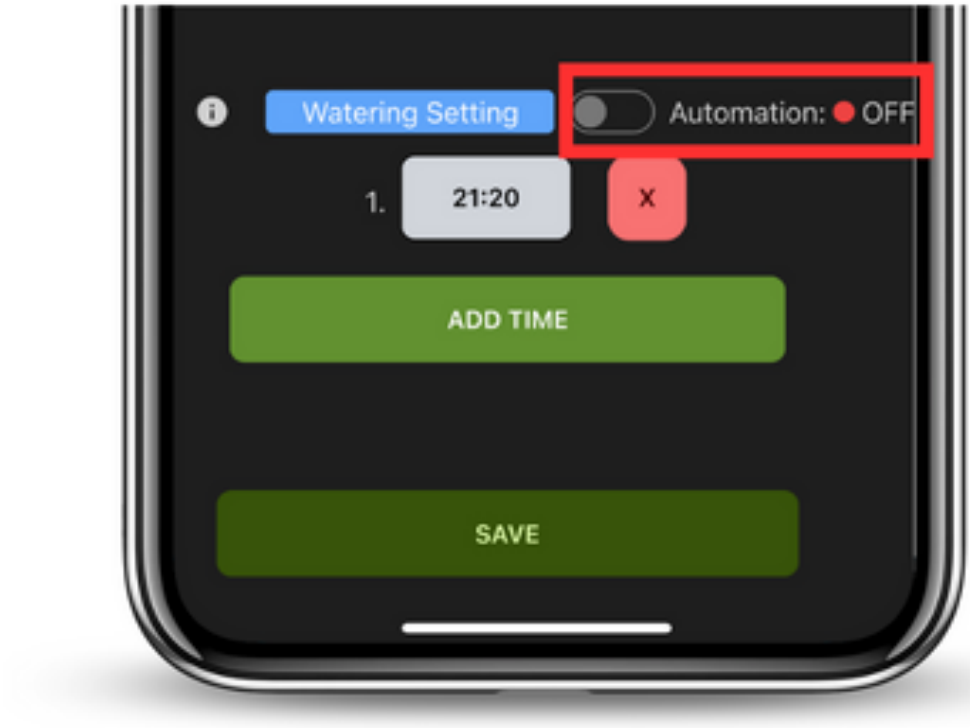

Figure 4.22 The watering automation is set to be off.

3. **Override the scheduled task.**

Similar to the farm light and the air conditioner, manual control of an watering system can override the command from the scheduled task. From test number 1, when the watering system is turned on at 21:20. Test number 3 can be conducted by turning the watering system off manually. The watering system reacted accordingly, indicating that the system works properly

4.2.3 Threshold Triggering System

4.2.3.1 Humidity Threshold

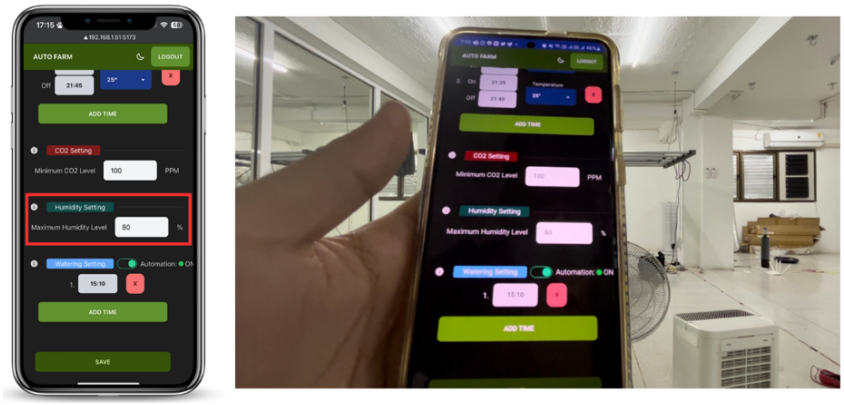

Figure 4.23 Set Humidity Threshold

Initially, the maximum humidity level is set to eighty percent. If any of the sensors detect a humidity level that exceeds this threshold, the dehumidifier controller is automatically activated. This is because some plants are highly sensitive to humidity, and an excessive level of humidity in the farm can result in the development of mold. Suppose we spray water all over the farm, and the humidity level subsequently increases beyond eighty percent. In that case, the dehumidifier controller will activate the fan, dehumidifier, and air ventilator to regulate the humidity level. However, the humidity level displayed on the screen shows only sixty-seven percent. This is because the displayed humidity level is the average of all the sensors.

Figure 4.24 Dehumidifier activated.

Suppose we spray water all over the farm, and the humidity level subsequently increases beyond eighty percent. In that case, the dehumidifier controller will activate the fan, dehumidifier, and air ventilator to regulate the humidity level. However, the humidity level displayed on the screen shows only sixty-seven percent. This is because the displayed humidity level is the average of all the sensors.

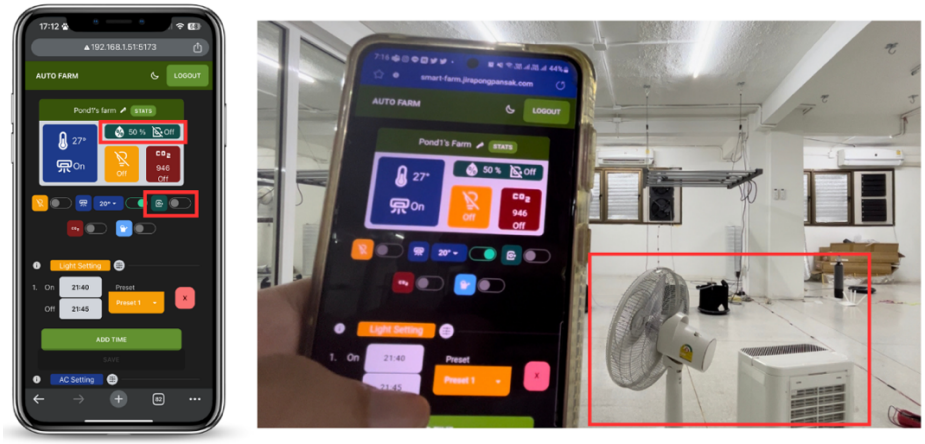

Figure 4.25 Dehumidifier deactivated.

Once the dehumidifier controller is activated, the system will automatically deactivate it when the humidity level drops below thirty percent of the threshold. The humidity level continues to decrease until it reaches fifty percent, at which point the dehumidifier is deactivated.

4.2.3.2 Carbon Dioxide Threshold

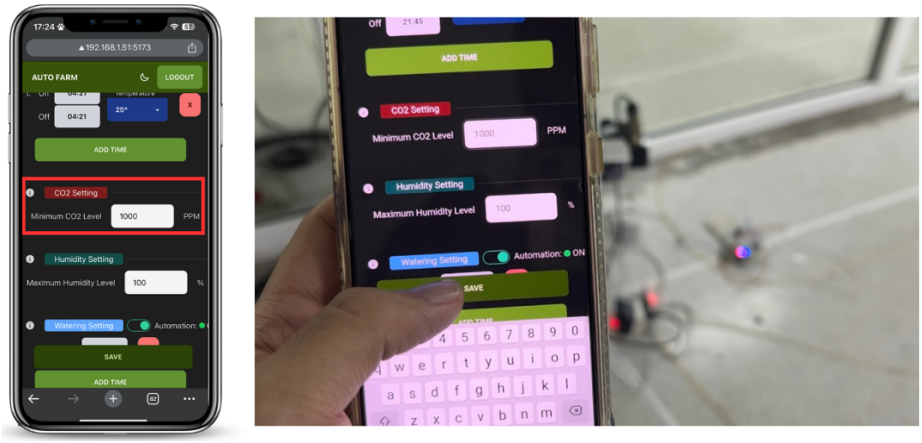

Figure 4.26 Set Carbon Dioxide Threshold

Initially, the Carbon Dioxide level in your farm is below 1000 ppm (parts per million). Since you are unable to decrease the Carbon Dioxide level manually, you decide to set the minimum threshold to be higher than the current Carbon Dioxide level in the farm. Thus, you set the minimum Carbon Dioxide level at 1000 ppm. As per your setup, if any of the sensors capture a Carbon Dioxide level below 1000 ppm, the Carbon Dioxide Controller will activate automatically.

Figure 4.27 Carbon Dioxide emitter activated.

Upon setting the minimum Carbon Dioxide level at 1000 ppm and given that the current Carbon Dioxide level in the farm is 824 ppm, the Carbon Dioxide Controller activates immediately.

Figure 4.28 Carbon Dioxide emitter deactivated.

Once the Carbon Dioxide controller is activated, the system operates in a way that automatically deactivates the controller when the Carbon Dioxide level in the farm exceeds 20 percent of the set threshold. The Carbon Dioxide level continues to rise until it reaches 1117 ppm. At this point, the Carbon Dioxide controller is deactivated.

4.2.4 Monitoring

The sensor transmits readings to our MQTT2DB service via MQTT. As its name suggests, the MQTT2DB service retrieves the data from MQTT and stores it in MongoDB. The data stored in MongoDB is utilized to generate charts depicting historical readings spanning several years.

When a user requests to view the data, the backend system retrieves the data from MongoDB and delivers it to the frontend.

The frontend displays the data to the user with selectable time frames such as by day, by week, and by month. The system can show the graphs of data over time for each item.

Figure 4.29 History by Day Week Month

Furthermore, users have the option to view individual graphs for each growth factor or a combined graph displaying all factors. This can be easily achieved by selecting the desired factor at the bottom of the graph interface. By providing this

flexibility, users can focus on specific growth factors of interest or analyze the overall trends by viewing all factors simultaneously.

Figure 4.30 Historical Data of each growth factor.

Additionally, the system offers the capability to export data in CSV format or display it in a tabular form, allowing for further analysis. Furthermore, users have the option to export the graph as either a JPG or PNG image file.

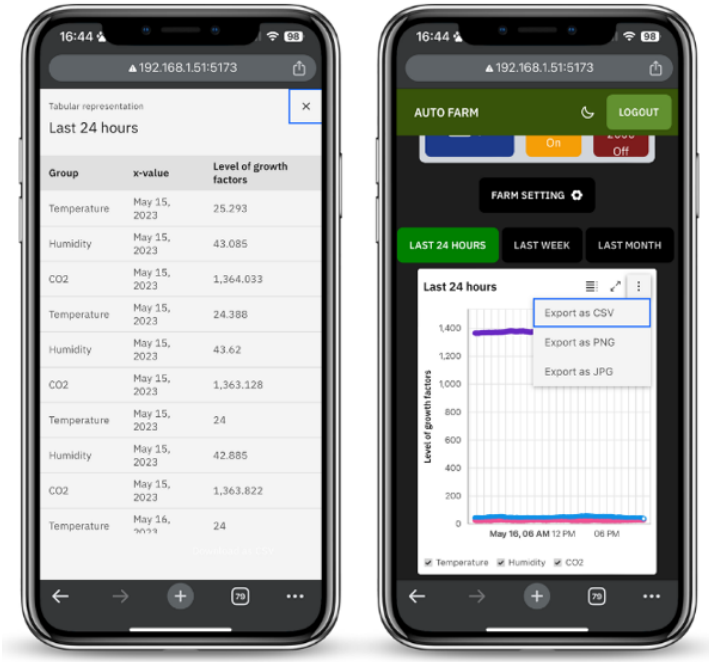

Figure 4.31 Exporting Data as CSV

4.3 Evaluation

For this project, users are able to track the growth of plants in a farm by monitoring various variables such as carbon dioxide concentration, humidity level, and temperature. These variables will be displayed along with the status of the hardware modules in the farm on a web application. The historical data of the plant growth variants will also be available to demonstrate statistical information about the farm.

Using IoT technology implemented in the farm, users can remotely control the electric appliance through the web application. The light system can be activated or deactivated, and the density of each type of light, such as ultraviolet or infrared light, can be adjusted. Users can set preset combinations for each type of light and also schedule the automation of the lights to activate and deactivate at specific times. The air conditioner can be turned on or off, and the temperature in the farm can be adjusted through the web application. Users can also set timers for the air conditioner at specific times. The watering system can be activated or deactivated remotely, and users can schedule the activation and deactivation of the watering system through time automation.

Additionally, users can maintain the optimal growth conditions for plants by setting thresholds for variables such as carbon dioxide concentration and humidity level. The system will automatically control actuators based on the sensor data captured at any given time and the defined threshold values. For example, if the carbon dioxide concentration falls below the minimum threshold or the humidity level exceeds the maximum threshold, the system will activate the appropriate actuators to adjust the conditions accordingly.

4.4 Testing and Evaluation Summary

In summary, the manual control feature for operating all the electric appliances on the farm has been successfully tested. Each electrical appliance can be toggled on or off through a web application, allowing the user to remotely and wirelessly control the system. Furthermore, the time automation feature for the watering system, light system,

and air conditioner has also been tested. This feature enables the user to set predetermined activation and deactivation times for the actuators on the farm.

Although the threshold trigger system could not be physically tested due to time constraints, it is designed to prevent critical levels of humidity and carbon dioxide concentration in the farm. The user has the ability to define threshold values for each variable, and when these values are reached, the actuators will be triggered accordingly. This system ensures that the farm remains within the desired parameters for optimal conditions.

CHAPTER 5
CONCLUSION

5.1 Introduction

In this Chapter, we first summarize the work described in this report (section 5.2). Then we draw several conclusions about key parts of the work undertaken in section 5.3, and finally, in sections 5.3.1,5.3.2, and 5.3.3 we discuss future work and how we see improvements to the project could be done. These changes could allow our customers to have a better user experience.

5.2 Summary

In summary, this project involves conducting research on the Internet of Things (IoT) and its application in a basic farm. It covers various aspects such as hardware, software, UX/UI design, and more. The web application has been designed as a responsive platform to cater to both desktop and mobile users, based on the stakeholder's requirements.

The hardware solutions are specifically designed to control the stakeholder's existing electrical appliance, ensuring compatibility with their older devices. The system architecture has been meticulously planned to support the entire IoT system, incorporating protocols such as MQTT and HTTP.

To assist the stakeholder in managing their farm, an IoT system has been implemented. This system enables remote control of various electronic appliances within the farm, including lights, watering systems, dehumidifiers, carbon dioxide emitters, air conditioners, air ventilators, and more. Through the web application, powered by IoT technology, the stakeholder can effortlessly monitor and adjust these devices.

Furthermore, the IoT system collects and stores valuable data, encompassing the status of all electronic appliances on the farm and the growth variations of the plants. This data is presented in real-time through the web application, allowing the stakeholder to have up-to-date information about the farm's current status. Additionally, historical data is accessible, providing valuable statistics and insights for farm analysis.

To ensure optimal functionality, the project has been designed and implemented appropriate algorithms that handle various use cases. Each aspect of the system, including the hardware, software, and user interface, has been meticulously designed, assembled, implemented, and tested within the defined scope. This comprehensive approach ensures that all modules are functional and user-friendly, meeting the stakeholder's requirements.

In terms of the backend logic, the web application interacts with the IoT devices with APIs and protocols such as MQTT and HTTP. The backend logic processes incoming requests from the user interface and translates them into commands comprehensible by the respective IoT devices. It also retrieves data from the devices and stores it in a database for real-time and historical data visualization.

The sequence diagram provides an overview of the interactions between the different components in the system. It illustrates the flow of events and communication between the web application, backend, and IoT devices, showcasing the step-by-step process of controlling and monitoring the farm devices.

Overall, this project encompasses comprehensive research, hardware, and software design, UX/UI development, and algorithm implementation to create an efficient IoT system for managing a farm.

5.3 Conclusions

In conclusion, this project has successfully fulfilled the requirements set by the stakeholders. It has provided the stakeholder with the ability to manually control the

electric appliance in the farm, implement time automation for the appliances, ensure the optimal growth conditions for the plants, and monitor the farm's status and statistics. However, due to constraints such as limited time, budget, and resources, there are areas where the project can be further improved. These improvements may include enhancing the accuracy of sensor data capture, implementing real-time monitoring capabilities, and exploring additional features to enhance the overall functionality of the system.

5.3.1 Monitoring

To enhance the project, you can implement real-time monitoring by integrating a real-time database like Firebase. This allows for immediate updates without a 5-minute delay. By setting up data listeners on the backend, you can track changes in the relevant data and trigger events when updates occur.

To notify the frontend about data updates, you can establish a socket connection between the frontend and backend. When a data update event is detected, the backend emits a socket event, providing real-time notifications to the frontend.

Upon receiving the socket event, the frontend can promptly send a request to the backend to fetch and query the updated data. This ensures that the frontend always has the latest information at its disposal.

By implementing these steps, you enable real-time monitoring in the project, significantly improving its responsiveness and ensuring that the frontend remains synchronized with the most up-to-date data in real time.

5.3.2 Notification

To address the current issues, you can implement RabbitMQ Message Queue and Socket functionality in the project. Currently, when a user performs an action in the web application, it doesn't execute the corresponding action in the farm, and the database status doesn't update. There is also a lack of error notifications to the user.

By integrating RabbitMQ Message Queue, the user's actions can be sent as messages to the queue instead of executing them directly on the farm. Backend workers can listen to the queue, retrieve the messages, and execute the actions on the farm. If

any errors occur during execution, the workers can send notification messages back to the RabbitMQ queue.

To notify the user about errors, a socket connection can be established between the frontend and backend. When an error notification message is received on the RabbitMQ queue, the backend sends a socket event to inform the frontend.

Upon receiving the socket event, the frontend can display a notification or toast message to the user, providing immediate feedback about the error that occurred during the action execution.

Implementing RabbitMQ Message Queue and Socket functionality ensures that user actions are properly executed on the farm, database statuses are updated, and users are promptly notified of any errors, improving the overall user experience and error handling in the project.

5.3.3 Maintaining the critical variants of the plant growth.

To ensure the effectiveness of the threshold trigger system for maintaining plant growth variables like carbon dioxide concentration and humidity level, it is crucial to conduct real-world testing. Although a desk check has been performed, it lacks real evidence of the system's performance in practical scenarios. Given more time, collecting sufficient farm data and conducting tests in an actual farm environment would provide valuable insights into the system's reliability and performance.

By tracking farm data and testing the threshold trigger system under real conditions, you can assess its functionality, accuracy, and responsiveness. This real-world testing will provide concrete evidence of how well the system performs in maintaining the desired plant growth conditions.

With this additional testing, any potential issues or limitations of the system can be identified and addressed. It will also enable you to fine-tune the thresholds and make necessary adjustments to ensure optimal plant growth and system performance.

Investing time in gathering farm data and conducting real-world testing will ultimately contribute to the system's reliability and confidence in its ability to maintain the desired plant growth variables effectively.

5.3.4 AI & ML Integration

An enhancement that can be implemented in this project is the integration of Artificial Intelligence (AI). By leveraging the available data, the system can learn from the current conditions and autonomously adjust the parameters. This AI capability empowers farmers to optimize crop yields. Furthermore, the data can be shared with other farmers through networking, leading to improved yields for the entire farming community.

If we market this product, gathering user feedback will enable us to enhance the models further. These improvements will not only benefit existing users but also provide a better experience for new users as well.

REFERENCES

1. *Dht11–temperature and humidity sensor*. Components101. (n.d.). https://components101.com/sensors/dht11-temperature-sensor
2. *MH-Z19C IR infrared CO2 Sensor Module Carbon Dioxide Gas Sensor Ndir for CO S6Y1 4894864000759*. eBay. (n.d.). https://www.ebay.com/itm/114848219092
3. *Futaba S3003 Servo (360 degree)*. Arduitronics. (n.d.). https://www.arduitronics.com/product/2462/futaba-s3003-servo-360-degree
4. *KY-005 38KHz modulating infrared IR transmitter sensor module durable transmitter sensor for Arduino FA: Walmart Canada*. Walmart.ca. (n.d.). https://www.walmart.ca/en/ip/KY-005-38KHz-Modulating-Infrared-IR-Transmitter-Sensor-Module-Durable-Transmitter-Sensor-For-Arduino-FA/PRD0YFOULW1P584
5. *R385 DC 6V-12V diaphragm based water pump*. rareComponents.com. (n.d.). https://rarecomponents.com/store/r385-mini-dc-12v-diaphragm-water-pump
6. *BESTEP 1ช่อง3.3V รีเลย์โมดูล Optocoupler Isolation terminal สำหรับ arduino*. Bestep 1ช่อง3.3v รีเลย์โมดูล Optocoupler Isolation Terminal สำหรับ Arduino - Buy Low Level Trigger Relay Module,Low Level Trigger Relay Module Product on Alibaba.com. (n.d.). https://thai.alibaba.com/product-detail/BESTEP-1-Channel-3-3V-Low-62290831568.html
7. /cidbioscience. (2022, March 16). *Transpiration in plants: Its importance and applications*. CID Bio-Science. https://cid-inc.com/blog/transpiration-in-plants-its-importance-and-applications/
8. *38KHz infrared IR transmitter sensor module for Arduino, operating voltage: 5V, forward current: 30 ~ 60 ma. REES52. (n.d.). https://rees52.com/38khz-infrared-ir-transmitter-sensor-module-for-arduino-ab034-997-optical-sensors/*
9. *Agarwal, T. (2019, August 6). DHT11 sensor definition, working and applications. ElProCus. https://www.elprocus.com/a-brief-on-dht11-sensor/*
10. *Carbon dioxide*. Wisconsin Department of Health Services. (2023, March 29). https://www.dhs.wisconsin.gov/chemical/carbondioxide.htm#:~:text=The%20levels%20of%20CO2,of%20drowsiness%20and%20poor%20air.

11. *How do CO2 sensors work? (n.d.). https://www.processsensing.com/en-us/blog/how-CO2-sensors-work.htm#:~:text=NDIR%20CO2%20sensors%20monitor%20and,through%20the%20filter%20wavelength%20choice.*
12. *Mendizábal, I. de. (2022, June 6). IOT communication protocols-IOT data protocols - technical articles. All About Circuits. https://www.allaboutcircuits.com/technical-articles/internet-of-things-communication-protocols-iot-data-protocols/*
13. Open Connectivity Foundation (OCF). (2022, February 7). https://openconnectivity.org/
14. *Red Hat. (2022, November). Cloud vs. edge. Red Hat - We make open source technologies for the enterprise. https://www.redhat.com/en/topics/cloud-computing/cloud-vs-edge#:~:text=A%20cloud%20is%20an%20IT,the%20source%20of%20the%20data.*
15. *Samuel. (2022, November 15). 5V relay module - how it works and application. GEYA Electrical Equipment Supply. https://www.geya.net/5v-relay-module-how-it-works-and-application/*
16. *Team, D. D. (2022, November 27). What are the main components of IOT?. Digital Directions. https://digitaldirections.com/main-components-of-iot/*
17. *Woodstream, W. (2020, June 11). What is a plant's vegetative stage? Safer® Brand Home, Garden & Lawn. https://www.saferbrand.com/articles/faq-vegetative/*
18. *BV, B. (n.d.). Introduction to the flowering phase. What is flowering and how can you make sure your plants flower well? https://www.plagron.com/en/hobby/grow-topics/introduction-to-the-flowering-phase#:~:text=In%20the%20flowering%20phase%2C%20a%20plant%20makes%20flowers%20and%20fruit. /*

APPENDICES

APPENDIX A

Source Code

The source code for the project is available to be developed further / forked at the github repository https://github.com/MaoMaoCake/Smart-Farm