Alexandria University Faculty of computers and Data Science

Computing and Data Science Department

Muthmir

Supervised by:

Dr. Mohamed El-habrouk

Contributors

Team 13	
Ahmed Samir Abdelfattah	20221450304
Ahmed Samir Abdulazim	20221373780
Ahmed Salah Abdelkader	20221445695
Ahmed Nasser Abdelfadeel	20221449224
Ali Mahmoud Ali Mahmoud	20221370972
Hazem Mohamed Ahmed Bakr	20221441990
Khalid Abdulhamd Hamdy	20221041599
Marwan Ashraf Mohamed	20221442040
Mustafa Saleh Mustafa	20221453750
Nour-Eldeen Anwar Mohamed	20221150099

Acknowledgments

We would like to extend our deepest gratitude to Dr. Mohamed Elhabrouk, our esteemed supervisor, whose support and expertise have been pivotal throughout the course of this project. His valuable insights, constructive feedback, and unwavering encouragement have played a significant role in guiding us to the successful completion of this work.

We also wish to express our sincere appreciation to Dr. Hesham, CEO of HST, for his generous contributions to this project.

Their support and involvement have been vital in advancing our research, and we are truly grateful for their collaboration.

Table of figures

Table of Contents

Chapter 1: Introduction

1.1 Agricultural Challenges and the Need for Smart Solutions

Agriculture is one of the most vital sectors for national development and food security. However, it faces increasing pressure due to limited natural resources, environmental degradation, and rising global demand for food. In Egypt, these pressures are particularly acute. The country is confronted with severe limitations in water availability, shrinking arable land, and low agricultural productivity per unit area. These challenges are compounded by the growing population, urban expansion, and climate variability.

To address these issues, Egypt has taken practical and strategic steps toward optimizing the use of its available resources. Key initiatives include:

- The adoption of greenhouse agriculture to improve crop yield and allow year-round production.
- Lining irrigation canals to reduce water loss and improve water use efficiency.
- Expanding the use of modern irrigation techniques, such as drip and sprinkler systems, to replace traditional flood irrigation.

These efforts are not exclusive to Egypt. Globally, nations are striving to strike a balance between increasing food production and reducing resource consumption. The push toward smart, sustainable agriculture is becoming a worldwide imperative to secure food for future generations.

Smart Agriculture as a Solution

Our project emerges from this urgent need for sustainable innovation. It focuses on developing a smart agricultural system capable of adapting to various agricultural contexts, including:

- Open-field farming
- Greenhouse cultivation
- Aromatic and medicinal plant production
- Vertical farming and high-efficiency systems

For the initial phase, we chose greenhouse agriculture as a prototype environment. Greenhouses offer a controlled and measurable setup, which makes it easier to design, test, and refine smart technologies such as automated control systems, environmental sensors,

and data analytics. Once the system is validated in greenhouses, it can be scaled and adapted to work in more complex environments like open-field farming and aromatic plant cultivation.

1.2 Agricultural greenhouse vs open field

Aspect	Open-Field Farming	Greenhouse Agriculture
Environment	Crops are grown outdoors in natural spaces like fields, meadows, and forests.	Crops are grown inside enclosed structures such as plastic tunnels, glasshouses, or polyethylene-covered structures.
Environmental Control	Exposed to varying weather conditions and environmental factors such as temperature fluctuations, rainfall variability, and winds.	Provides a controlled environment inside the greenhouse, allowing better monitoring and control of temperature, humidity, lighting, soil quality, and nutrient levels.
Resource Usage	Requires large land areas for crop cultivation, relying on natural resources such as water, soil, and sunlight.	Utilizes less land space as crops can be grown vertically or horizontally, and relies on resource control techniques like controlled irrigation, artificial lighting, and regulated ventilation.
Productivity and Quality	Productivity may be affected by environmental conditions and natural factors. Crops may be more susceptible to diseases and pests.	Can achieve higher productivity and better quality crops due to improved environmental control and protection from harmful environmental factors and pests.
Resource Sustainability	Open-field farming may consume natural resources more significantly and be vulnerable to environmental degradation factors like desertification, water, and soil pollution.	Greenhouse agriculture can achieve better resource sustainability through techniques such as recycling, efficient water use, and controlled pesticide usage.

The Egyptian agricultural sector faces several fundamental challenges that hinder its ability to meet local needs or develop exports to address the continuous trade deficit. The key problems include:

- **Limited Water and Land Resources**: The scarcity of water and arable land poses significant constraints on agricultural productivity and expansion.
- **Low Per-Acre Productivity:** The productivity per unit of land (per acre) remains relatively low, impacting overall output and efficiency.

- **Production-Harvest Mismatch:** The agricultural production and harvest cycles are primarily linked to climatic conditions, often not aligning with suitable export periods characterized by intense global market competition.

Significance of Greenhouses: Greenhouse farming plays a crucial role as a modern agricultural technique, facilitating increased production, efficient land use, high-quality and pollutant-free crop yields, even off-season, and conservation of water resources compared to traditional farming methods.

Ministry of Agriculture's Project: In response to these challenges, the Ministry of Agriculture and Land Reclamation has recently embarked on a national project to **establish 100,000 agricultural greenhouses** over 100,000 acres.

جدول رقم (3) : التوزيع الجغرافى للصوب الزراعية فى مصر خلال متوسط الفترة (2016-2019)

المحافظة	المساحة ($م^2$)	%	الإنتاج (طن)	%	العدد (صوبة)	%
الدقهلية	13160338	49.0	105849	41.1	39705	54.5
الجيزة	1965940	7.3	23600	9.2	6024	8.3
الغربية	1546598	5.8	20833	8.1	4752	6.5
النوبارية	1718750	6.4	25174	9.8	3438	4.7
الشرقية	1975462	7.4	14462	5.6	3171	4.3
الإسماعيلية	1187028	4.4	13792	5.4	3083	4.2
المنوفية	890764	3.3	7423	2.9	2579	3.5
دمياط	874340	3.3	6876	2.7	2244	3.1
كفر الشيخ	527772	2.0	6983	2.7	1661	2.3
السويس	566210	2.1	3869	1.5	1507	2.1
البحيرة	1059538	3.9	19923	7.7	1307	1.8
باقى المحافظات	1384364	5.2	8891	3.5	3449	4.7
إجمالى	**26857101**	**100**	**257675**	**100**	**72918**	**100**

المصدر: جمعت وحسبت من : وزارة الزراعة وإستصلاح الأراضى، قطاع الشؤن الإقتصادية ،نشرة مشروعات الأمن الغذائى ، أعداد مختلفة.

Figure 1.1 The Geographical Distribution of Greenhouses in Egypt between 2016 and 2019

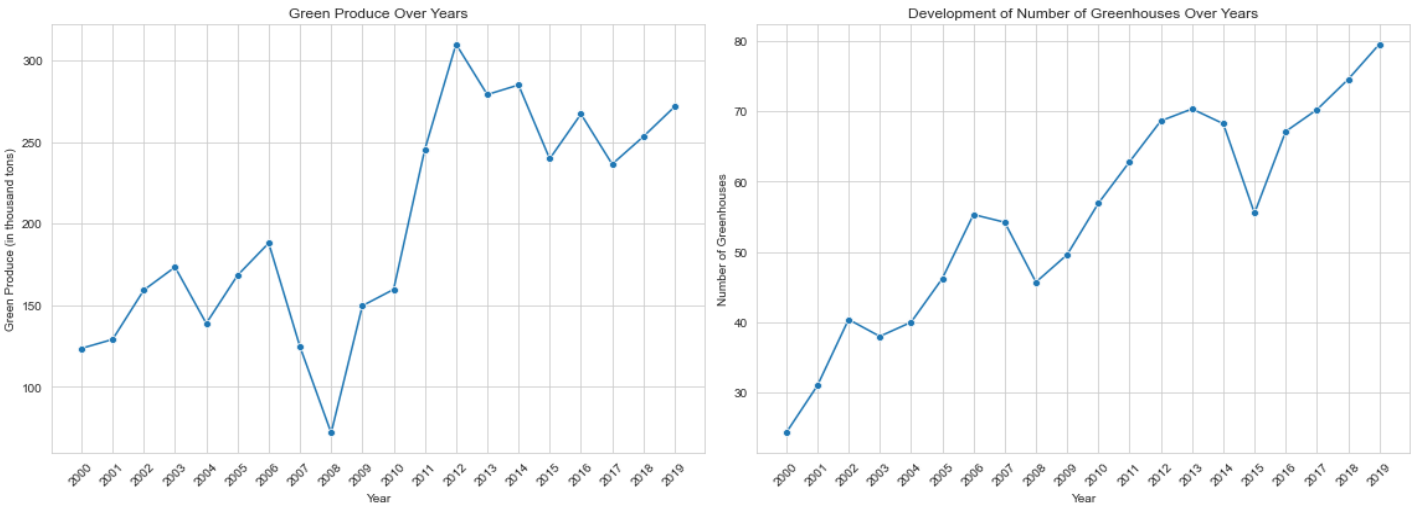


Figure 1.2 greenhouses over years

جدول ٨. إنتاجية وحدتى الأرض الزراعية ومياه الرى لمحاصيل الخضر موضع الدراسة بالصوب الزراعية بعينة الدراسة الميدانية للموسم الزراعي ٢٠١٦/٢٠١٧.

البيان	الخيار الشتوى	الطماطم الشتوى	الفلفل الشتوى
نظام الرى:			
حقل مكشوف:	غمر	غمر	غمر
إنتاجية الفدان بالطن	٩،٠	١٧،٥	٦،٦
الاحتياجات المائية للفدان م^{3}	١٧٧١	١٨٠٢	١٦٣٧
صوبة زراعية:	تنقيط	تنقيط	تنقيط
إنتاجية الفدان صوب بالطن	٥٠،٠	٨٠،٠	٣٠،٠
الاحتياجات المائية للفدان صوب م^{3}	١٦١٠	١٦٤٠	٢٢٠٠
الاحتياجات المائية (م٣ / م٢):			
حقل مكشوف	٠،٤٤	٠،٤٥	٠،٤١
صوبة زراعية	٠،٤٠	٠،٤١	٠،٥٥
مقدار الفرق: م^{3}	٠،٠٤ –	٠،٠٤ –	٠،١٤
٪	٩،١	٨،٩	٣٤،١
إنتاجية وحدة الأرض (كجم /م٢):			
حقل مكشوف	٢،٣	٤،٤	١،٧
صوبة زراعية	١٢،٣	١٤،٢	٨،٣
مقدار الزيادة: كجم /م^{3}	١٠	٩،٨	٦،٦
٪	٤٣٥	٢٢٣	٣٨٨
إنتاجية وحدة المياه (كجم / م٣):			
حقل مكشوف	٥،١	٩،٧	٤
صوبة زراعية	٣١،١	٤٨،٤	١٣،٦
الفرق: كجم /م^{3}	٢٦	٣٩،١	٩،٦
٪	٥٠٨،٨	٤٠٣،١	٢٤٠

المصدر: جمعت وحسبت من بيانات:

١. عينة الدراسة الميدانية بمنطقة النوبارية للموسم الزراعي ٢٠١٦/٢٠١٧.
٢. معهد بحوث البساتين، قسم بحوث الزراعة المحمية.
٣. الجهاز المركزي للتعبئة العامة والإحصاء، نشرة الموارد المائية والري، موقع الجهاز على شبكة الانترنت.

Figure 1.3 Productivity of agricultural land units and irrigation water for vegetable crops under study in greenhouses

1.3 Types of Greenhouse:

Greenhouses vary according to:

- Its geometric shapes
- Type of structure
- on the covers
- Heating sources
- Cooling sources
- Ventilation sources
- Irrigation systems

Greenhouses consist of the following main parts:

-1 The structure

-2 covers

-3 irrigation systems,

-4 ventilation, cooling and heating

Here's a breakdown of the main parts of a greenhouse and how they can vary:

1- **Structure:**
   Geometric Shapes: Greenhouses can have various shapes such as traditional rectangular or square designs, as well as more specialized shapes like domes, A-frames, or even multi-span structures.

   Type of Structure: The structure can be made of materials like glass, polycarbonate panels, plastic film, or fiberglass, each offering different levels of insulation, light transmission, and durability.

2- **Covers:**
   Covers can include materials like glass, which provides excellent light transmission and durability but can be expensive. Polycarbonate panels are another option known for their strength and insulation properties. Plastic film is more affordable but may need replacement more frequently.

3- **Irrigation Systems:**

Irrigation systems vary from simple hand-watering methods to automated drip irrigation systems or overhead sprinklers. The choice depends on factors like water availability, crop water requirements, and labor considerations.

4- Ventilation, Cooling, and Heating:

Ventilation Sources: Greenhouses require adequate ventilation to control humidity, temperature, and carbon dioxide levels. Ventilation can be achieved through natural methods like vents or fans and automated systems.

Cooling Sources: Cooling methods can include shading systems, evaporative cooling pads, or misting systems to maintain optimal temperatures during hot weather.

Heating Sources: Heating systems like gas heaters, electric heaters, or geothermal systems are used to maintain temperatures during colder months or in regions with harsh winters.

5- type of greenhouse control system:

Basic Timer System: A system that controls the timing of watering or irrigation in a greenhouse, typically offering simple scheduling options for one or a few zones.

Advanced Timer System: A more sophisticated version of a timer system that can control multiple zones, integrate with sensors for automated adjustments, and provide more flexibility in programming watering schedules.

Basic Greenhouse Controller: A control panel that goes beyond just watering, allowing for control of additional devices like fans and ventilation systems, and often integrating with environmental sensors for automated adjustments.

Professional Greenhouse Controller: An advanced control system designed for maximum control over a wide range of greenhouse devices and systems, supporting advanced features like Vapor Pressure Deficit Irrigation, scalability for large operations, and integration with PC or smartphone for remote monitoring and control

When choosing a greenhouse design and components, it's crucial to consider the local climate, environmental conditions, crop requirements, budget constraints, and long-term sustainability goals. Consulting with experts in greenhouse design and agriculture can help in selecting the most suitable type of greenhouse for specific needs and environmental conditions.

positive of using greenhouse	negative of using greenhouse
Enhanced Crop Yield	Requires a Significant Initial Capital
Reduced Exposure to Weather Risks	Requires Precise Design
Greater Profitability	Higher Production Costs (Operating Costs)
Better Pest and Disease Management	Demands Higher Skill Levels
Year-Round Cultivation	Ideal Conditions for Diseases
Lower Pesticide Usage	Requires Stable Marketing Operations

Table 1.1 advantages and disadvantages of greenhouse

1.4 Advanced Greenhouse control systems

The advanced greenhouse control systems market offers a promising future Here are some factors:

- **Growing Demand:** Global demand for advanced greenhouse control systems is rising due to increased focus on sustainable agriculture and efficient crop production.

- **Technological Advancements:** Rapid progress in IoT, AI, Mobile and data analytics has enhanced the capabilities of greenhouse control systems, offering competitive advantages.

- **Environmental Focus:** There's a significant emphasis on systems that optimize resource usage and minimize environmental impact, aligning with industry and regulatory priorities.

1.5 Project objectives

Turning challenges into opportunities, our product not only enhances existing advantages but also solves critical problems, setting a new standard for excellence in the market so our objective:

- **Optimized Resource Management:** our system for efficient greenhouse management can help reduce operating costs by optimizing resource usage, such as energy and water. This addresses the challenge of higher production costs and ensures sustainable practices.
- **Remote Management Flexibility:** Through advanced remote monitoring and control functionalities, our system facilitates year-round cultivation. This capability ensures mitigating the impact of labor shortages and enhancing overall productivity and profitability

- **Data-Driven Pest and Disease Management:** By utilizing data analytics and monitoring capabilities, our system can enhance pest and disease management strategies. This improves crop health and reduces the risks associated with ideal conditions for diseases, contributing to better pest management within the greenhouse.
- **Efficient Water and Pesticide Usage:** our system's data analysis and automation can optimize water and pesticide usage, reducing overall consumption. This aligns with efficient water management and lower pesticide usage, promoting environmentally friendly practices.
- **Market Intelligence and Decision Support:** our system can provide market intelligence through data science, helping farmers make informed decisions about crop selection and market demand. This supports stable marketing operations and profitability.

Chapter 2: Prototype

2.1 Determine the type of plant in our prototype

We have selected tomato plants for our greenhouse prototype, recognizing their importance for cultivation in Egypt. Tomatoes are not only a staple crop but also one of the most popular choices for greenhouse cultivation due to their high demand and versatile usage. By focusing on tomatoes, we aim to address a critical agricultural need while harnessing the potential of greenhouse technology to optimize growth conditions and ensure consistent production year-round. This initiative underscores our commitment to bolstering food security and promoting sustainable agricultural practices in the region.

2.2 Key Environmental Factors for Efficient Farming

1. Temperature

- Impact on Plants:
  Controls growth rates, photosynthesis, and nutrient uptake.
  Extreme temperatures can stress plants, reducing productivity.
- Monitoring and Control:
  Sensors: Measure real-time temperature.
  Heating Systems: electric heaters for maintaining optimal warmth in colder conditions.
- Cooling Methods: evaporative coolers to prevent overheating.

2. Humidity

- Impact on Plants:
  High humidity promotes fungal diseases; low humidity may cause dehydration.
  Balancing transpiration rates is critical for healthy crop growth.
- Monitoring and Control:
  Sensors: Monitor relative humidity levels.
- Ventilation: vents to regulate airflow.
- Misting Systems: Increase humidity during dry conditions.

3. Soil Moisture

- Impact on Plants:
  Essential for nutrient transport and root health.
  Overwatering leads to root rot; under-watering causes wilting.

- Monitoring and Control:
  Moisture Sensors: Detect water levels in the soil.
  Irrigation Systems: Automated systems for precise watering.
- Ai: Use IoT tools to predict irrigation needs based on weather and plant growth stages.

4. Light

- Impact on Plants:
  Drives photosynthesis and affects flowering and fruiting cycles.
  Seasonal changes and cloudy days can limit light availability.
- Monitoring and Control:
  Grow Lights: Supplement natural light with LED systems.
- Light Sensors: Detect light intensity .
- Automation: Systems that adjust lighting duration and intensity based on crop needs.

2.3 Overview

Our prototype integrates cutting-edge technology to optimize farming efficiency and sustainability. It is divided into four interconnected components:

- IoT System
- AI Integration
- Mobile Application
- Data Analysis and Reporting

1- <u>IoT System</u>

The IoT system is the backbone of the prototype, divided into two main functionalities:

Monitoring (Sensors and Cameras)

Sensors: Measure key environmental factors such as:

- Temperature
- Humidity
- Light intensity
- Soil moisture

Cameras: Capture real-time images of plants for visual analysis, supporting AI-driven disease detection.

Data Flow: These devices upload real-time data to Google Cloud for processing and storage.

Control (Action Devices)

Automated devices that take actions based on sensor feedback:

- **Grow Lights:** Maintain optimal light conditions.
- **Irrigation System:** Delivers water based on soil moisture levels.
- **Heaters:** Regulate temperature during colder conditions.
- **Vents:** Control airflow and reduce humidity.
- **Evaporative Coolers:** Lower temperature during hot conditions.

All control actions are logged and uploaded to Google Cloud for system monitoring and analysis.

2- AI Integration

The AI component, hosted on Google Cloud, adds advanced intelligence to the system.

Plant Health Monitoring:
Analyzes images from greenhouse cameras to detect signs of plant diseases.
Identifies early-stage issues and generates alerts.
Notifications:
If a disease is detected, the AI sends an alert to the mobile application for action.

3- Mobile Application

The mobile application bridges the gap between users and the greenhouse system.

Monitoring:

Retrieves real-time data from Google Cloud, displaying temperature, humidity, light, and soil moisture levels.

camera footage for visual inspection.

Control:
Allows users to override automated controls and manually operate devices such as grow lights, irrigation systems, and vents.

4- Data Analysis

Data analysis is a crucial component for decision-making and performance optimization.

Data Processing:

Retrieves historical and real-time data from Google Cloud.

Analyzes trends, such as temperature fluctuations, water usage, and plant growth conditions.

Reporting:

Generates detailed reports with insights and recommendations.

Sends these reports to the mobile application for user review.

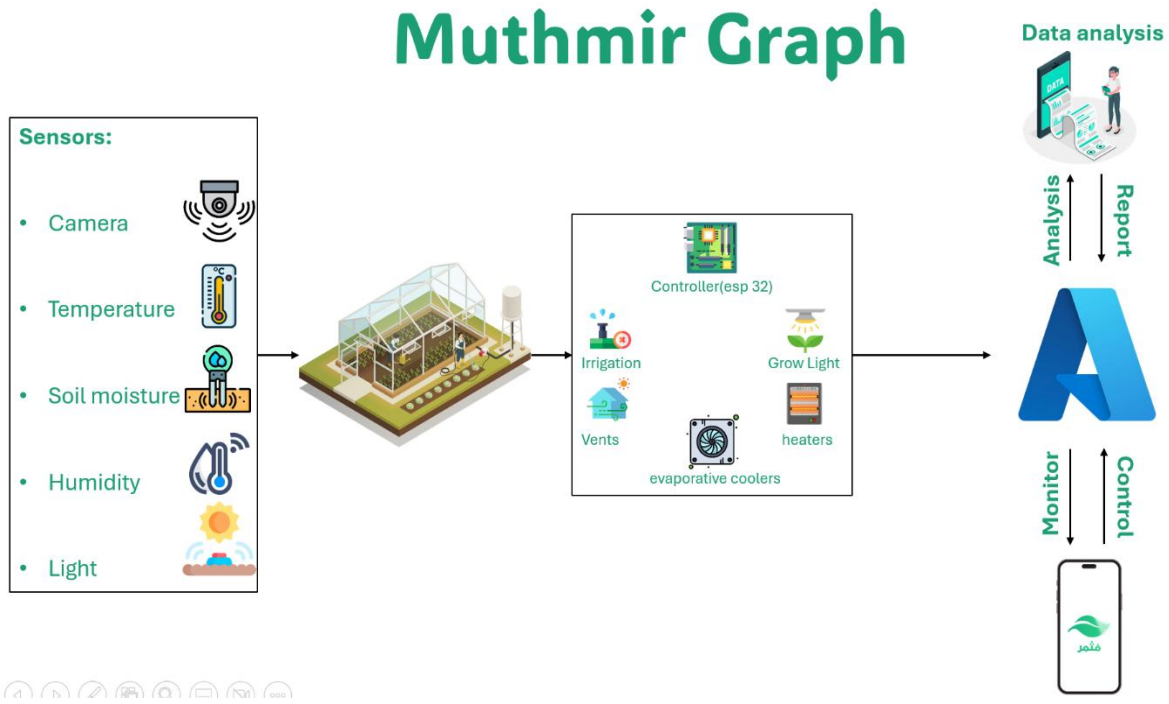


Figure 2.1 block diagram

Chapter 3: IOT

3.1 Introduction

3.1.1 The Need for IoT in 2025

The Internet of Things (IoT) has emerged as a cornerstone of technological advancement, revolutionizing the way devices communicate, interact, and automate processes. In 2025, IoT continues to play a critical role in enhancing efficiency[1], personalization, and connectivity across various domains, from smart homes and healthcare to industrial automation and urban infrastructure.

Consider a simple yet transformative use case: a smart lamp in a home. Through IoT, the lamp can gather and analyze historical usage data, such as when it's typically turned on or off. This data enables intelligent scheduling and energy optimization, reducing waste while enhancing convenience. Building on this, the integration of a motion sensor connected to the door could take automation a step further. As someone walks into the room, the sensor detects movement and seamlessly triggers the lamp to light up, creating a personalized and responsive environment.

When multiple devices and sensors begin communicating with each other, the possibilities expand even further. The interconnected system can collect a wealth of data from various sources, providing a comprehensive view of usage patterns and behaviors. This opens the door to leveraging machine learning models that analyze this data to uncover helpful insights, predict future needs, and optimize performance, making technology not just smarter but deeply intuitive.

This interconnected ecosystem exemplifies the transformative potential of IoT in 2025, enabling a future where devices work together intelligently to enhance everyday life and provide sustainable, data-driven solutions.

3.1.2 How IoT Integration is Made Possible: The Role of Microcontrollers

The seamless integration of IoT devices, sensors, and machine learning models is largely enabled by advancements in microcontrollers. A microcontroller is a compact integrated circuit designed to perform specific tasks within an embedded system. It serves as the “brain” of IoT devices, enabling them to process data, communicate with other devices, and execute automated actions based on predefined instructions or real-time inputs.

3.1.3 Microcontroller Architecture [2]

The architecture of a microcontroller is tailored to ensure low power consumption and efficiency, making it ideal for IoT applications. Its core components include:

1. **Central Processing Unit (CPU):**
   Executes instructions, processes data, and controls operations of the microcontroller. Popular microcontrollers use 8-bit, 16-bit, or 32-bit processors, with 32-bit architectures (like ARM Cortex-M) being prominent for IoT due to their performance and power efficiency.
2. **Memory:**
   - **Flash Memory:** Stores firmware and programs.
   - **RAM (Random Access Memory):** Temporary storage for active processes.
   - **EEPROM:** Non-volatile storage for persistent settings or small datasets.
3. **Input/Output (I/O) Ports:**
   Connects the microcontroller to sensors, actuators, and other devices, supporting both digital and analog signals.
4. **Communication Interfaces:**
   Includes UART, SPI, I2C, and wireless protocols like Wi-Fi, Bluetooth, Zigbee, and LoRa, enabling IoT connectivity.
5. **Timers and Counters:**
   Provide timing for operations like triggering sensors or scheduling tasks.
6. **Power Management:**
   Optimizes energy usage with low-power modes, essential for battery-powered IoT systems.

Examples of Popular Microcontrollers

1. **ATmega (used in Arduino devkits):**
   - **Description:** An 8-bit AVR microcontroller, widely used for prototyping and education.
   - **Strength:** User-friendly with extensive community support.
2. **ESP8266 and ESP32 (by Espressif Systems):**

- **Description:** A dual-core 32-bit microprocessor with integrated Wi-Fi and Bluetooth capabilities.
- **Strength:** Cost-effective, energy-efficient, and ideal for wireless IoT applications.

3. **Broadcom BCM2837 (used in Raspberry Pi):**
   - **Description:** A quad-core 64-bit ARM processor that runs a full operating system (e.g., Linux). Used in more complex IoT systems requiring advanced processing, such as smart home hubs or machine learning applications.
   - **Strength:** High computational power, suitable for tasks requiring more processing, such as image recognition

Most Used (ESP32):

ESP microcontrollers, especially the ESP32, are favored because of their powerful dual-core processor, integrated Wi-Fi and Bluetooth modules, low energy consumption, and affordability. They simplify IoT development by reducing the need for additional communication modules, making them the go-to choice for hobbyists and professionals alike.

3.2 Protocols

3.2.1 MQTT (Message Queuing Telemetry Transport):

MQTT (Message Queuing Telemetry Transport) is a lightweight and efficient communication protocol designed for data exchange between IoT devices, widely used for monitoring and controlling devices. It follows a publish/subscribe (Pub/Sub) model, where devices (Publishers) send messages to a central broker, which then distributes them to subscribed devices (Subscribers). This enables real-time, event-driven communication, making MQTT ideal for resource-constrained devices and applications requiring reliable messaging over unstable networks. It is standardized by OASIS and ISO.

3.2.1.1 Communication model

MQTT is considered a communication protocol that follows the Publisher/Subscriber communication model. It relies on transmission protocols like TCP/IP for reliable data transport between devices.

3.2.1.2 MQTT Architecture

Publisher: Sends data to a specific topic via the broker.

Broker: Receives data from publishers and distributes it to the appropriate subscribers.

Subscriber: Receives data from the broker when subscribed to a specific topic.

3.2.1.3 What is its use in our project?

We used it in our project to monitor devices such as air conditioning, pump, LED lamp, and tungsten lamp, and through MQTT it tells us the status of each device, whether it is ON or OFF.

3.2.1.4 Comparison between MQTT and HTTP

Feature	MQTT (Message Queuing Telemetry Transport)	HTTP (Hyper Text Transfer Protocol)
Underlying Protocol	TCP/IP	TCP/IP
Architecture	Publish-Subscribe (can also use Request/Reply mode)	Request-Response
Command Targets	Topics	URIs (Uniform Resource Identifiers)
Message Size (Minimum)	2 bytes (header data can be binary)	8 bytes (header data is text - compression possible)
Message Size (Maximum)	256MB	No limit, but 256MB is beyond normal use cases
Content Type	Any (binary)	Text (Base64 encoding for binary)
Reliability	Three levels of service: 0 - Fire and forget, 1 - At least once, 2 - Exactly once	Must be implemented in the application

Table 3.1 comparison between MQTT and HTTP

NO. MESSAGES IN A CONNECTION CYCLE FOR MQTT	MQTT AVG. RESPONSE TIME PER MESSAGE (MS) (QOS 1)	HTTP AVG. RESPONSE TIME PER MESSAGE (MS)
1	113	289
10	47	289
100	43	289

Table 3.2 Response rate of MQTT

The reduction in response time as we send more messages during a single TCP connection is expected, as we do not incur the connection setup overhead for each message. HTTP does not have that ability - some approaches such as caching or connection pooling can improve response times and overhead but not to the extent to match MQTT.

3.2.1.5 Why to use MQTT

MQTT	2 bytes (header)	256MB	Fast in unstable networks using TCP, 1-100 ms in local networks	Ideal for IoT applications in resource-constrained environments, handles unstable connections
CoAP	4 bytes (header)	64KB	Faster than MQTT in low-bandwidth environments, 0.1-1 ms in local networks	Best for IoT devices with low overhead and fast communication needs
AMQP	4-8 bytes (header)	No strict limit, MBs+	Slower than MQTT and CoAP, 50-200 ms depending on message size	Enterprise applications for secure, reliable messaging between distributed systems
DDS	10-50 bytes (header)	No strict limit, practical limits in MBs	1-10 ms in real-time environments, faster with UDP, slower with TCP	Real-time systems like robotics, automotive, and military applications
HTTP	8 bytes (header)	No strict limit, 256MB beyond normal use	Slower than MQTT, 100-500 ms depending on network	Web applications, client-server communication, not suitable for real-time IoT

Table 3.3 comparison between different protocols use cases

3.2.1.6 Conclusion

MQTT is the ideal choice for IoT applications requiring fast, reliable communication, offering a balance of speed and reliability. It is faster than AMQP and DDS in most cases and provides optimal performance for real-time monitoring in resource-constrained environments. While CoAP is faster than MQTT, it is less reliable and better suited for specific low-overhead IoT use cases. HTTP, however, is better suited for web applications and client-server communication that don't require real-time responses. Based on these factors, MQTT was chosen for our project to efficiently monitor devices and check their status.

3.2.2 HTTP (Hypertext Transfer Protocol)

HTTP (Hypertext Transfer Protocol) is a protocol used for transmitting hypermedia documents like HTML. Originally designed for communication between web browsers and web servers, it can also be used for machine-to-machine communication and accessing APIs. HTTP operates on a request-response model, where a client sends an HTTP request to a server, and the server responds with an HTTP response containing the requested data

or an error message. This makes HTTP essential for web applications and data exchange across the internet.

3.2.2.1 Communication model

HTTP is a communication protocol that follows the Request/Response communication model. It relies on transmission protocols like TCP/IP for reliable data transport between devices.

3.2.2.2 HTTP Architecture

HTTP operates on a request-response model:

Client: Sends an HTTP request to a server, including the requested resource, HTTP method, headers, and optional body data.

Server: Processes the request and responds with an HTTP response, which contains a status code, headers, and the requested data or an error message.

3.2.2.3 What is its use in our project?

1-Data Collection:

The system devices collect data from sensors such as the DHT22 to measure temperature and humidity.

2-Sending Data via HTTP:

After collecting the data, the system sends it to our server using an HTTP POST request. In this process, the data is sent in JSON format over the internet, including details like temperature and humidity, along with an Authorization header for authentication. The REST API allows the ESP32 to interact with the server directly by sending and receiving data over the HTTP protocol.

3-Data Storage in MongoDB:

The server receives the data and stores it in Mongo DB. Applications or interfaces can then access this data for real-time display and analysis.

3.2.2.4 Why to use HTTP

HTTP is easy to implement compared to complex protocols like MQTT, requiring less in-depth protocol knowledge. Firebase provides HTTP-based APIs, simplifying the connection between the ESP32 device and the server. HTTP is ideal for web and mobile applications that need to interact with the server, making it a practical choice in this context. In terms of speed, HTTP is slower (100-500ms) compared to protocols like MQTT, but it is sufficient for tasks that involve periodic data transmission from ESP32 to the server.

However, HTTP may face challenges if the network is unstable, potentially increasing latency.

3.2.2.5 Conclusion

Therefore, we used HTTP in our project to securely and reliably transfer data to the server.

3.2.3 Wi-Fi (Wireless Fidelity):

Wi-Fi is a technology that allows devices to connect to the internet or local networks wirelessly using radio waves. It operates on the IEEE 802.11 family of standards and is widely used in homes, offices, and public spaces to provide wireless internet access.

MQTT	2 bytes (header)	256MB	Fast in unstable networks using TCP, 1-100 ms in local networks	Ideal for IoT applications in resource-constrained environments, handles unstable connections
CoAP	4 bytes (header)	64KB	Faster than MQTT in low-bandwidth environments, 0.1-1 ms in local networks	Best for IoT devices with low overhead and fast communication needs
AMQP	4-8 bytes (header)	No strict limit, MBs+	Slower than MQTT and CoAP, 50-200 ms depending on message size	Enterprise applications for secure, reliable messaging between distributed systems
DDS	10-50 bytes (header)	No strict limit, practical limits in MBs	1-10 ms in real-time environments, faster with UDP, slower with TCP	Real-time systems like robotics, automotive, and military applications
HTTP	8 bytes (header)	No strict limit, 256MB beyond normal use	Slower than MQTT, 100-500 ms depending on network	Web applications, client-server communication, not suitable for real-time IoT

Table 3.4 Comparison between communication technologies

3.2.3.1 Communication model

Wi-Fi is a standard (Transfer Protocol) that uses a set of protocols and technologies to provide wireless connectivity to the Internet and other devices.

Therefore, Wi-Fi is a standard that defines how data is formatted, transmitted, and received over a wireless network, using transmission protocols like TCP/IP to ensure reliable and efficient communication between devices.

Wi-Fi data rate is based on its protocol type:

Protocol	Maximum Data Rate
IEEE 802.11a	Up to **54 Mbps**
IEEE 802.11b	Up to **11 Mbps**
IEEE 802.11g	Up to **54 Mbps**
IEEE 802.11n	Up to **150 Mbps**
IEEE 802.11ac	Up to **866.7 Mbps**
IEEE 802.11ad	Up to **7 Gbps**

Table 3.5 Wi-Fi protocols

3.2.3.2 Why did we use it in our project?

We used to connect microcontrollers like ESP32 and ESP_CAM to the internet. This is due to the following features of the connectivity:

1-Supported Frequencies: subGHz, 2.4GHz, and 5GHz bands.

2-Data Transfer Speed: Ranges from 0.1 to 54 Mbps depending on the protocol used.

3-Range: Up to ~300 feet(30 meters indoors), making it suitable for indoor use.

4-Power Consumption: Moderate, making it suitable for portable devices.

3.3 prototype

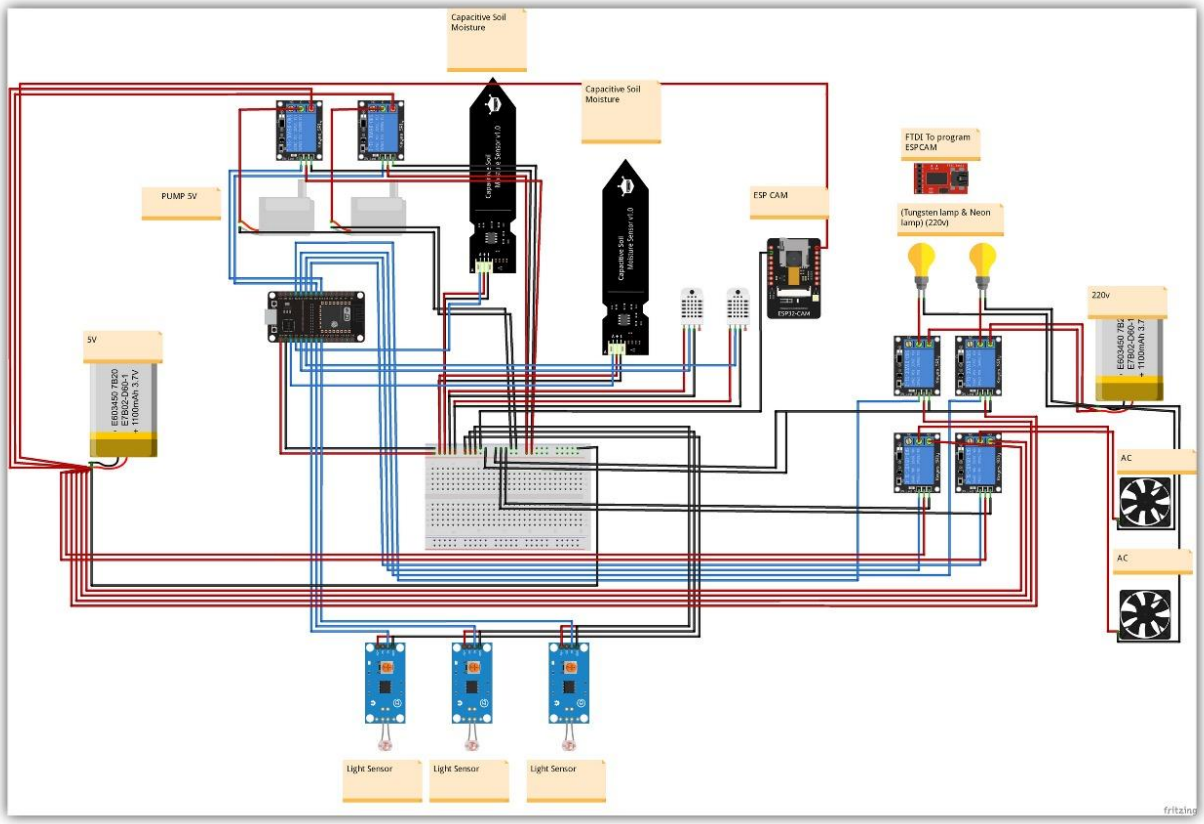


Figure 3.1 schematic diagram

3.3.1 Sensors

3.3.1.1 DHT22 [5]

is an advanced digital sensor used to measure temperature and humidity with high accuracy. Compared to other sensors like the DHT11, it offers improved precision, making it suitable for applications requiring reliable environmental monitoring.

Module Connections

The module has a 3-pin header on the assembly.
1 x 3 Header

- '+' : Connect to 3.3V.
- OUT : Digital sensor I/O – Connects to a digital pin on a microcontroller
- '- ' : Connect to system ground. This ground needs to be in common with the microcontroller.

Measurement Range:

Temperature:

- Measurement range: -40 to +80°C
- Resolution: 8-bit (0.1 °C)
- Accuracy: ± 0.5 °C
- Response time: on average – 2 seconds

Humidity:

- Measuring range: 0 – 99.9 %RH
- Resolution: 8-bit (±0,1 %RH)
- Accuracy: ±2 %RH*
- Response time: on average – 2 seconds

3.3.1.2 Capacitive Soil Moisture Sensor v1.2 [6]

is a version of the capacitive sensor used to measure soil moisture levels accurately. It works by detecting changes in capacitance between two conductive plates or electrodes inserted into the soil, which is affected by the water content in the soil.

Ideal Moisture Range for Tomatoes:

- The ideal soil moisture for tomatoes typically ranges between 60% to 70% RH (Relative Humidity).

- If the moisture is below 60%, tomatoes may experience water deficiency, leading to plant stress and stunted growth.

- If the moisture exceeds 70%, excess moisture may cause root rot or fungal diseases.

Module Connections:

There is a 3-pin JST PH2.0 type connector on the probe. One end of the supplied cable plugs into the connector and the other end is a standard Dupont style 3-pin female connector. The cable is color coded with black for ground, red for VCC and yellow for AOUT 1×3 Connector

- GND = Ground, must be common with the MCU
- VCC = 3.3V – 5.5VDC. May be powered from a digital output pin on a MCU
- AOUT = Analog output usually connected to an analog input on a MCU

3.3.1.3 LDR (Light Dependent Resistor) [7]

An LDR is a resistor whose resistance changes based on the amount of light falling on it. The more intense the light, the lower its resistance, and vice versa: the less intense the light, the higher its resistance.

Measuring Conditions

1-Light resistance:
Measured at 10 Lux with standard light A (2854K color temperature) and 2hr illumination at 400-600 lux prior to testing.
2-Dark Resistance: Measured 10 senconds after closed 10 lux.
3-Gamma Characteristic: Between 10 lux ande 100 lux and given by γ = lg(R10/R100) R10 、R100 Cell resistance at 10 lux and 100 lux. The error of γ is ± 0.1.
4-Pmax: Max. power dissipation at ambient temperature of 25 ℃.
5-Vmax: Max. voltage in darkness that may be applied to the cell continuously.

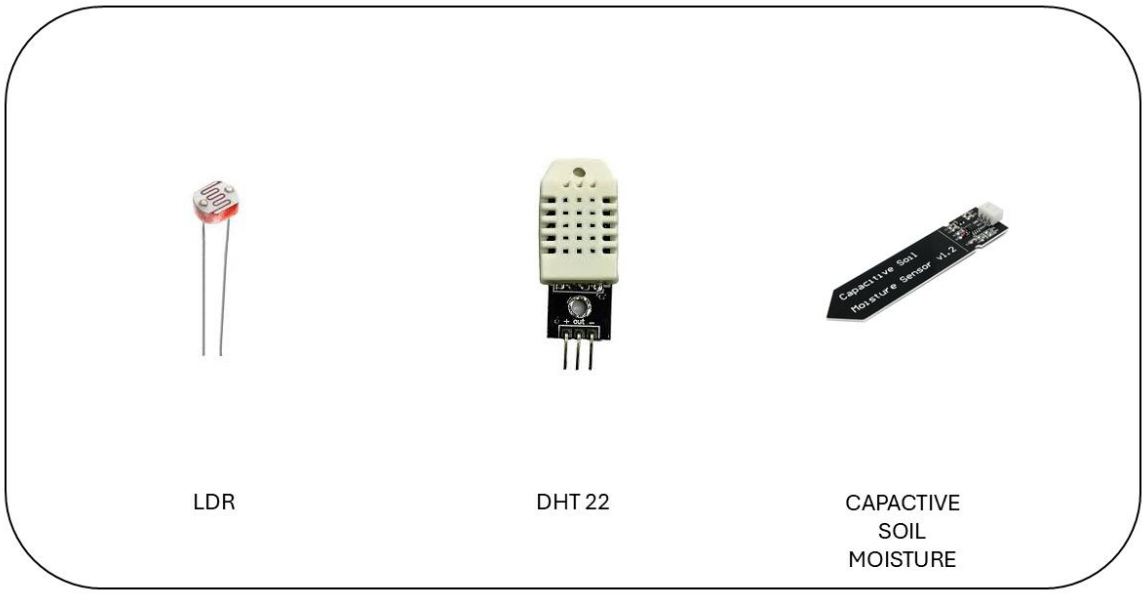


Figure 3.2 sensors used

3.3.2 Micro Controllers

3.3.2.1 ESP32 [3-4]

- Esp32 is a function controller that takes data from sensors and sends data to the cloud via the wife protocol
- It is characterized by security such as encryption and connection security

Specs:

-SRAM: 512KB
- Memory: 4MB
- Operates on voltage 3,3v or 5v from a usb
- Built-in Wi-Fi and bluetooth
- Supports UART / I2C / SPI / DAC / ADC
- 30-pin board is more than enough for most prototypes

3.3.2.2 ESP CAM

is a controller used to take pictures of the plant in the greenhouse.

Esp cam is connected to an AI model on a server that predicts the health and age of the plant to make a decision to save the plant in the event of the plant becoming sick.

Specs:

- we use the ov2643 which is a 2MP camera.
- Esp32 cam has a slot for microSD to save pictures taken on and a 4MB PSRAM

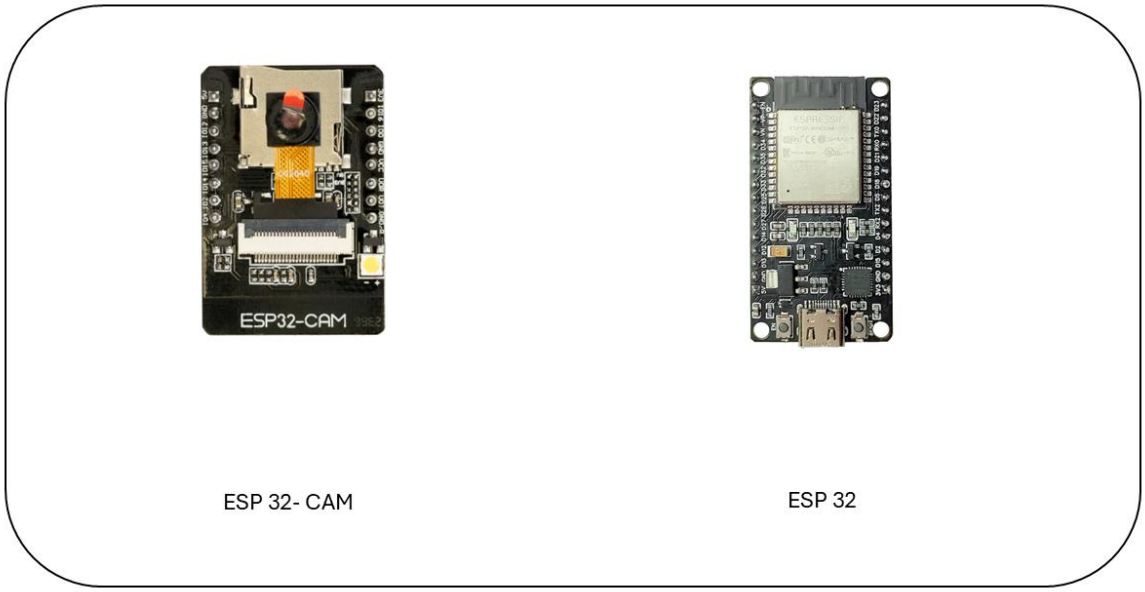


Figure 3.3 esp23

3.3.3 Devices

3.3.3.1Water Pump 5V

The pump is basically a DC motor that is powered with 5V and draws 100mA.

Specification:

1-Operating Voltage: 5V DC
2-Operating Currunt: 0.1-0.2A
3-Flow rate: 80-100L/H
4-Max. Lift: 35cm
5-Wire length: ~23cm

3.3.3.2 Relay

- The relay allows switching between different connected circuits based on control signals, enabling the user to choose whether to turn a circuit on or off.

- switch up to 30V DC or 125/250V AC

Module Connections

1x 3 Header

- GND = Connect to ground of the power supply
- IN1 = The control end, the driving current should be more than 4mA (Active Low)
- Vcc = Connect to power supply

Relay Contacts

- NC = Normally Closed side of the relay
- Center Terminal = Common terminal usually connects to the load power source
- NO = Normally Open side of the relay

3.3.3.3 Desert cooler

Its role in our smart greenhouse is to cool the greenhouse in case the temperature of the greenhouse rises above the permissible limit of 30C, and it remains operating until the temperature returns to normal.

3.3.3.4 Lamp [11]

1-Red Light (627 nm):

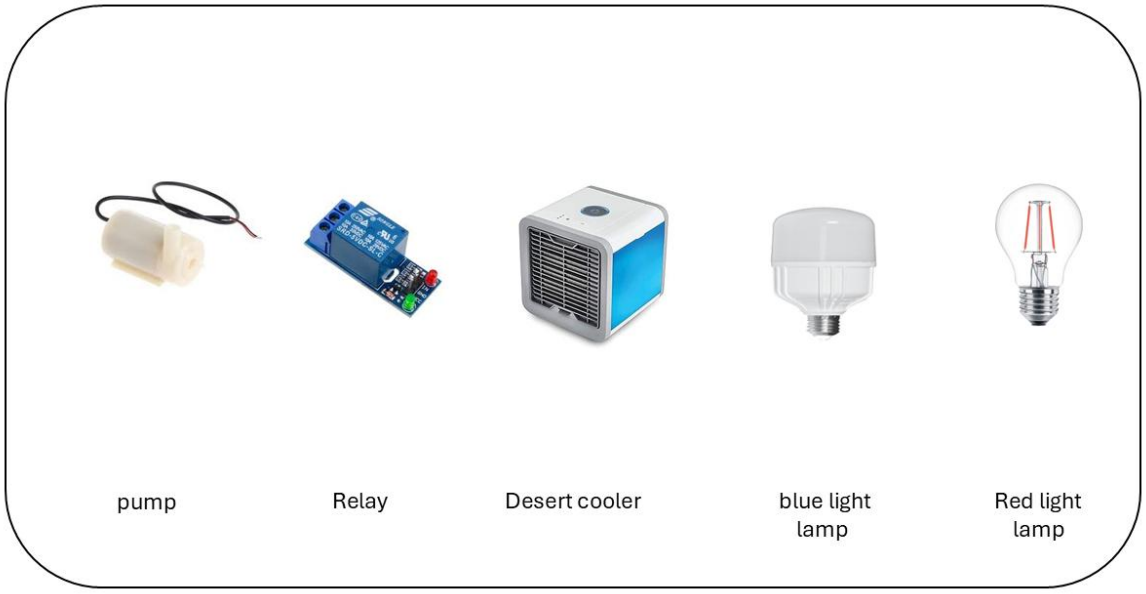
1-Crucial for flowering and fruiting stages.

2-Enhances growth and increases fruit yield.

2-Blue Light (450 nm):

1-Essential for leaf growth and overall plant development.

Figure 3.4 devices used



3.4 Industrial

3.4.1 Sensors

- **7 in 1 Integrated Soil Sensor EC PH NPK Moisture Temperature Meter:** Monitoring soil EC, PH, NPK, moisture and temperature **($265)**
- **XF500 Compact Weather Station:** Measures: Ambient Temperature, Relative Humidity, Wind Speed, Wind Direction, Atmospheric Pressure **($345)**
- **S-Light-01:** Industrial Light Intensity Sensor **($65)**

3.4.2 Industrial Connection Methods

- **Modbus:** A serial communicating protocol used to transmit data
- **RS485:** All mentioned sensors use RS485 for signal transmission as it can transmit data for up to 1200 Meters

3.4.3 RS485

- The RS485 protocol is a serial communication protocol developed for transmitting data over twisted pair cables in industrial and commercial environments.

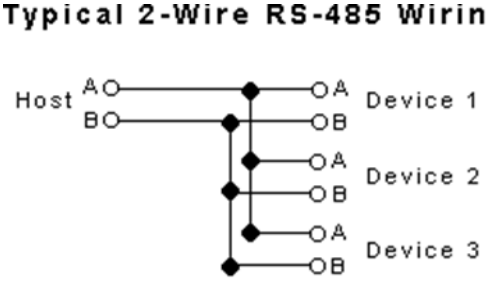


Figure 3.5 RS-485 wiring

- The RS485 protocol offers the following advantages:
- 1-Physical Layer: RS485 uses a balanced differential signaling method, which means it transmits data using two wires: one for data transmission (A) and one for data reception (B). This differential signaling helps to improve noise immunity and allows for longer cable lengths compared to single-ended signaling methods like RS232.
- 2-Voltage Levels: RS485 uses voltage levels that range from -7V to +12V for logical low and high states, respectively. However, the exact voltage levels can vary depending on the specific implementation and driver circuitry.
- 3-Topology: RS485 supports multi-drop configuration, allowing multiple devices (up to 32 or more) to be connected on the same bus. Each device on the bus has a unique address, and communication occurs in a master-slave or peer-to-peer fashion.
- EX: RS485 protocol is commonly used in industrial automation, such as industrial process control, building automation systems, security and protection systems,

3.5 system flow

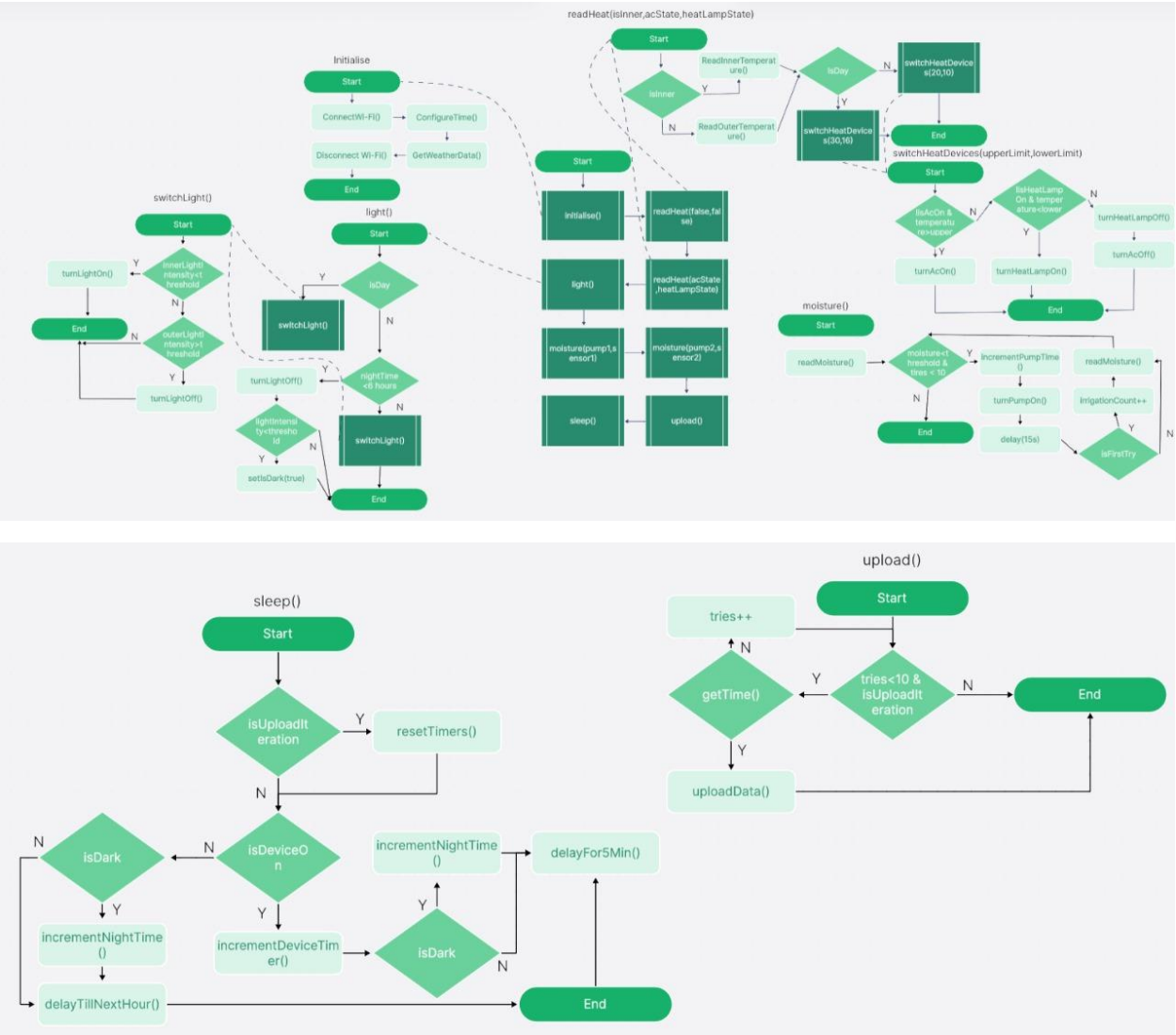


Figure 3.6 system flow

Chapter 4: AI

4.1 Tomato leaves disease detection model [12]

4.1.1 Background

Tomato crops are a vital component of global agriculture, yet they are highly susceptible to various leaf diseases. These diseases not only reduce crop yield but also incur significant economic losses. Early and accurate detection is critical for timely intervention and disease management.

4.1.2 Objectives

This study aims to develop a robust detection model capable of identifying multiple tomato leaf diseases with high precision and recall. The goal is to enhance agricultural productivity through reliable disease monitoring.

4.1.3 problem statement

In the context of computer vision, there are several tasks to choose from:

1. **Classification**: The model assigns one output from a predefined set of classes based on the input image.
2. **Classification with Localization**: Like classification, but in addition to the class prediction, the model also draws a bounding box around the detected object.
3. **Object Detection**: The model can detect multiple objects within an image, and for each detected object, it draws a bounding box around it.
4. **Segmentation**: This task involves identifying and separating individual objects within an image, highlighting the boundaries of each object.

Given that our task focuses on detecting the health status of multiple leaves in an image, **object detection** has been selected as the most suitable approach due to existing of multiple objects in the same image.

4.1.4 Dataset

The diseases selected for this study are prevalent all over the world. The dataset, which contains 9 distinct categories of plant leaves, was sourced from Roboflow website:Early Blight, Healthy, Late Blight, Leaf Miner ,Leaf Mold, Mosaic Virus, Septoria ,Spider Mites ,Yellow Leaf and Curl Virus.

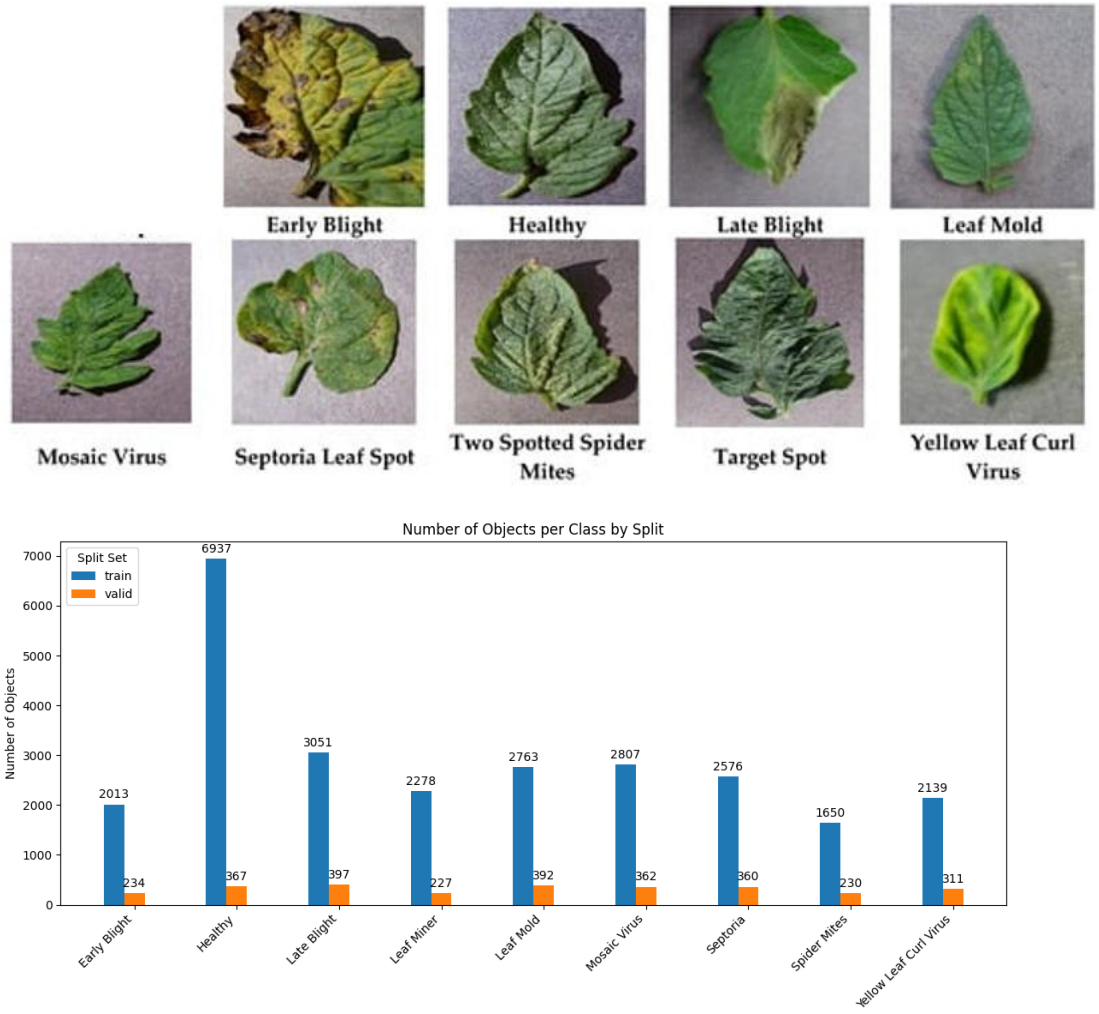


Figure 4.1 dataset classes and objects per class and set

The dataset consisted of a total of 6092 images and 29.094, including both healthy and diseased tomato leaves. The training set comprised 88% of the data,10% For validation,

Augmentation Type	Value
Outputs per training example	3
90 ° rotate	Clockwise, counter-clockwise,upside Down
Flip	Horizontal and Vertical
Saturation	Between -25% and +25%
Brightness	Between -23% and +23%
Blur	Up to 4.3px

Table 4.1 augmentation used

Component	Specification
Environment	Kaggle
Hardware	4× NVIDIA T4 GPUs, CPU

Table 4.2 Computational setup

4.1.5 Yolov8m model

In this project, the YOLOv8 (You Only Look Once, version 8) model was utilized to predict tomato diseases based on images captured within a protected agricultural environment. YOLOv8 is a state-of-the-art object detection model that offers high accuracy and real-time processing capabilities, making it an excellent choice for smart agriculture applications. The model was trained on the dataset described above, which includes various tomato diseases such as Early Blight, Healthy, Late Blight, Leaf Miner, Leaf Mold, Mosaic Virus, Septoria, Spider Mites, and Yellow Leaf Curl Virus. By accurately identifying these diseases, the system can monitor the tomato health and alert the user for any disease detected.

The overall model performance achieved a precision of 0.915, a recall of 0.808, and an mAP@0.5 of 0.812, with mAP@0.5:0.95 reaching 0.412.

```
Model summary (fused): 72 layers, 11,129,067 parameters, 0 gradients, 28.5 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 11/11 [00:06<00:0
0,  1.75it/s]
                   all        681       2880      0.915      0.808      0.878      0.812
           Early Blight        183        234      0.881      0.825      0.879      0.835
                Healthy         84        367       0.74      0.466      0.572       0.41
            Late Blight        258        397      0.909      0.872      0.927      0.887
              Leaf Miner        157        227      0.963      0.911      0.962      0.924
               Leaf Mold        202        392      0.911      0.784      0.875      0.796
           Mosaic Virus        268        362      0.977      0.923      0.973      0.925
              Septoria         215        360      0.937      0.864      0.931      0.885
           Spider Mites        155        230      0.975      0.961      0.983      0.932
Yellow Leaf Curl Virus         153        311      0.941      0.663      0.798      0.717
```

Figure 4.2 growth model performance

4.2 Optimal Camera placement [8]

4.2.1 overview

In this study [1] Wireless Visual Sensor Network (WVSN) are used for monitoring plant growth with the added feature of a camera. The goal is to represent a mathematical formulation and an optimal solution for the camera placement to guarantee coverage of large area while maintaining high quality resolution and minimizing overlap between cameras.
To improve the performance in terms of storage and processing and reduce the response time of the image processing unit. It is also necessary to capture images with high resolution for better processing and analysis

Optimizing the number of sensor cameras will help in:

(i) Optimizing the limited storage space of the sensor camera nodes. (Means selecting an ideal quantity of cameras to achieve the best performance or efficiency for a given purpose.)

(ii) Decreasing the processing time to then be able to analyze these images quickly and isolate plants showing signs of disease.

(iii) Producing high-quality images for avoiding false detections.

(iv) Minimizing the project cost since WVSN systems can be expensive to install and maintain.

4.2.2 Optimal placement camera quality problem formulation

Preliminaries In this work, we consider the angular field of view of the camera is, $\theta \in]0, 180]$ Each camera is characterized by the following parameters:

- Focal Length (FL): is the distance between the lens and the image sensor when the subject is in focus.
- Angle of View (AOV): is the angle subtended by the camera lens, i.e., the visible extent of the scene captured by the camera lens. A wide-angle of view captures a broader area, and vice versa.
- Resolution: is the number of pixels per image. The higher the number of pixels, the higher the image quality.

The Field of View (FOV) for the covered area in the WVSN can be specified by the Angular Field of View (AFOV), in degrees, or the Linear Field of View (LFOV), in meters. The AFOV is defined by the focal length, f, and the horizontal dimension of sensor in millimeters, b, as in Equation 1. The shorter the FL, the wider the AFOV.

$$AFOV = 2\,tan^{-1}\left(\frac{b}{2f}\right) \qquad (1)$$

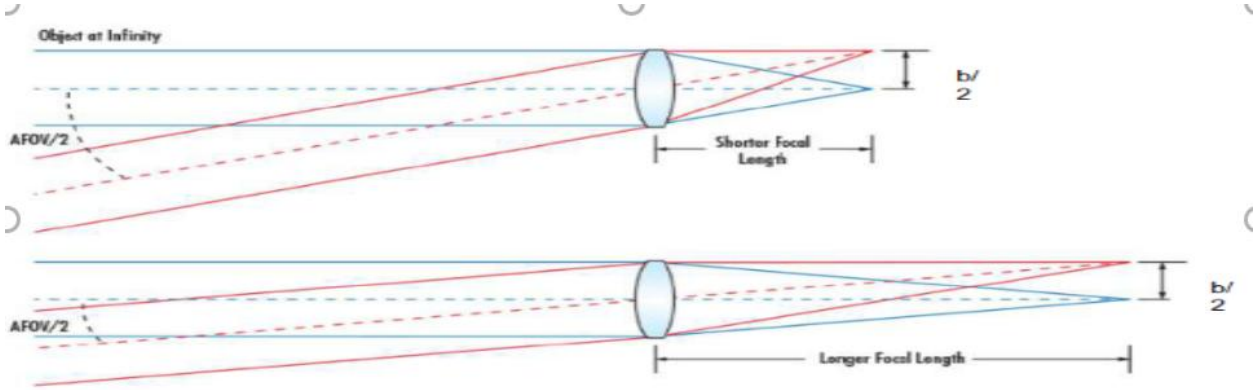


Figure 4.3 Angular Field of View (AFOV)

The larger the distance between the sensor and the monitored area, the larger the LFOV the poorer the quality of the collected images.

4.2.3 Assumption and Definition

Assume that the greenhouse is a rectangular area with length L and width W. In the beginning, we assume that there is no obstacle between the sensor camera and the plants, so we can place the cameras wherever we feel we need them.

The assumptions regarding the properties of the WVSN cameras are as follows:

- All the cameras have the same characteristics.
- All the cameras have the same resolution
- Field of view is θ €]0,180]
- Camera resolution R is defined by number of horizontal pixels times number of vertical pixels
- The monitoring area is a shape in two-dimensional Euclidean space.
- The cameras can be fixed in a predefined position and placed at the same height

4.2.4 Optimization Problem Formulation

In this work, we are interested in determining:
1) The optimal placement of cameras to have the desired quality of the image object.
2) The required number of cameras to cover the entire monitored area.
3) The positioning of the cameras so that there is no overlap between images taken by the cameras.

Figure 4.4 Horizontal Field of View (HFOV)

Symbol	Meaning
C	Set of cameras
θ	Angle of view
R	Image resolution, total number of pixels of the image
R_h	Number of pixels in the horizontal line of an image
R_v	Number of pixels in the vertical line of an image
Q	The image quality, number of pixels per unit distance
Q_{min}	The minimum accepted quality

Figure 4.5 symbols meanings

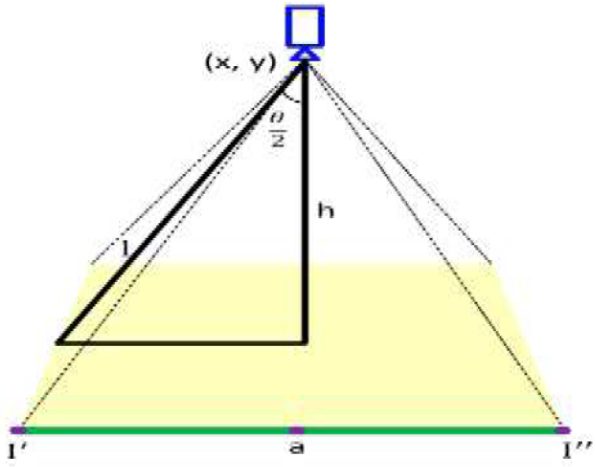


Fig 1: The Horizontal Field of View

The height, h, (or the distance from the camera to the sensed object or the ground), and the AFOV, θ, are known. But a/2 is unknown.

Hence,

$$x_{i'} = x_i - \frac{a}{2} \quad (2)$$

$$y_{i'} = y_i \quad (3)$$

$$x_{i''} = x_i + \frac{a}{2} \quad (4)$$

$$y_{i''} = y_i \quad (5)$$

Initially, we are interested in finding the optimal placement of the cameras considering the horizontal field of view. For this reason, the value of the coordinates over the Y-axis of the covered area is equal to the value of the Y-axis of the camera. And the covered area on the X-axis is:

We consider that the monitoring area is a rectangle in the two-dimensional Euclidean space defined by the points ABCD, such that $A = (x_a, y_a), B = (x_b, y_b)$, C= (x_c, y_c), and D = (x_d, y_d). We are interested in the covered area determined by the horizontal field of view of the cameras. In other words, we are interested in the part of the line defined by the points A and D. Denote the set of cameras by $C = \{1, \ldots, i, \ldots, c\}$. A camera i has an AOV denoted by θ_i, and we have $\forall (i,j) \in C^2, \theta_i = \theta_j$. Denote the coordinates of the camera i by (x_i, y_i). Our aim is to calculate the coordinates of the part covered by the camera, illustrated in (1) as a green line [I′ I′′], where point I′ is $(x_{i'}, y_{i'})$ and point I′′ is $(x_{i''}, y_{i''})$. Coordinates of both points can be calculated as follows.

$$(x_{i'}, y_{i'}) = \left(x_i - h \times \tan\left(\frac{\theta}{2}\right), y_i\right) \quad (6)$$

$$(x_{i''}, y_{i''}) = \left(x_i + h \times \tan\left(\frac{\theta}{2}\right), y_i\right) \quad (7)$$

Given h and θ, $\frac{a}{2}$ can be found as follows:

$$sin\left(\frac{\theta}{2}\right) = \frac{a/2}{l} \Rightarrow a/2 = sin\left(\frac{\theta}{2}\right) \times l$$

and

$$\cos\left(\frac{\theta}{2}\right) = \frac{h}{l}$$

Hence,

$$a/_2 = h \times \tan\left(\frac{\theta}{2}\right) \quad (8)$$

$$a = 2 \times h \times \tan\left(\frac{\theta}{2}\right)$$

4.2.5 ILP optimization problem formulation

- We present in this section the ILP formulation of our defined optimization problem above.
- We call the ILP problem as Integer Linear Programming-Optimal Placement Camera Quality (ILP-OPCQ).
- To solve the ILP-OPCQ problem we identify the workspace as a grid map; the monitored area (i.e., the monitored green line) corresponds to a vector line with:

$$L = E[r2_x] \text{ lines}$$

- The space where the cameras can be placed is viewed as a grid map with:

$$K = E[h_{max}] \text{ columns},$$

- Resolution of the camera R = $R_h * R_v$. R_h number of pixels in the horizontal line of an image and R_v number of pixels in the vertical line of an image.

4.2.6 ILP-OPCQ Objective Function Definition

We consider rephrasing the objective function as follows:

1. Minimizing the number of cameras needed for covering the entire monitored area:

$$\text{Min} \sum_{k=h_{min}}^{h_{max}} \sum_{l=r1_x}^{r2_x} P(k,l) \quad (10)$$

where $r1_x$, $r2_x$ are the X-axis coordinates of the horizontal line of the area.

$$P(k,l) = \begin{cases} 1, & if\ a\ camera\ placed\ at\ coordinates\ (k,l) \\ 0, & otherwise \end{cases}$$

2. Maximizing k in the Eq. (2), which stands for maximizing the covered area by each camera.

$$\text{Max} \sum_{k=h_{min}}^{h_{max}} \sum_{l=x_1}^{x_2} 2 \times k \times \tan\left(\frac{\theta}{2}\right) \times P(k,l)$$

3. Maximizing k in the Eq. (3), which stands for maximizing the resolution quality of images.

$$\text{Max}\left\{\frac{\sum_{k=h_{min}}^{h_{max}} \sum_{l=r1_x}^{r2_x} P(k,l) \times R_h}{\sum_{k=h_{min}}^{h_{max}} \sum_{l=x_1}^{x_2} 2 * k * \tan\left(\frac{\theta}{2}\right)}\right\}$$

4. By combining the three equations, the ILP-OPCQ objective function can be written:

$$\text{Max}\left\{\frac{\sum_{k=h_{min}}^{h_{max}} \sum_{l=x_1}^{x_2} 2 \times k \times \tan\left(\frac{\theta}{2}\right) \times P_{(k,l)} + \frac{\sum_{k=h_{min}}^{h_{max}} \sum_{l=r1_x}^{r2_x} P(k,l) \times R_h}{\sum_{k=h_{min}}^{h_{max}} \sum_{k=x_1}^{x_2} 2 \times k \times \tan\left(\frac{\theta}{2}\right)}}{\sum_{k=h_{min}}^{h_{max}} \sum_{l=x_1}^{x_2} P(k,l)}\right\}$$

4.2.7 Constrains of the ILP-OPCQ Problem

1. The cameras cannot be placed at a height less than h_{min} and greater than h_{max}.
2. We can place at most a single camera in a column.
3. The right sight covered of any camera should be lower than the X-axis coordinate of the end of the covered area. In other words, the camera's field of view should not extend beyond the specified endpoint on the X-axis.

Consequently, with the aid of the constrains, the optimization problem can be rephrased as follows:

Objective:

$$Max\{\frac{\sum_{k=h_{min}}^{h_{max}}\sum_{l=x_1}^{x_2} 2\times k\times tan\left(\frac{\theta}{2}\right)\times P(k,l) \;+\; \frac{\sum_{k=h_{min}}^{h_{max}}\sum_{l=r1_x}^{r2_x} P(k,l)\times R_h}{\sum_{k=h_{min}}^{h_{max}}\sum_{l=x_1}^{x_2} 2\times k\times tan\left(\frac{\theta}{2}\right)}}{\sum_{k=h_{min}}^{h_{max}}\sum_{l=x_1}^{x_2} P(k,l)}$$

4.3 Greenhouse System chatbot

4.3.1 The Challenge at Hand

Farmers and greenhouse operators often face challenges in managing day-to-day agricultural operations due to a lack of real-time insights, limited access to expert knowledge, and difficulty interacting with technical systems. Monitoring environmental conditions, searching for up-to-date agricultural information, and operating greenhouse devices typically require manual effort or technical expertise, which can be inefficient and error prone.

4.3.2 Proposed solution

To overcome these challenges, an AI-powered conversational chatbot can serve as an intelligent interface between the greenhouse system and the end-user (e.g., farmer or technician). By leveraging recent advances in Natural Language Processing (NLP), Retrieval-Augmented Generation (RAG), and IoT integration, the AI chatbot acts as a centralized, easy-to-use assistant capable of:

- Reducing reliance on technical expertise to operate the system.
- Offering instant, accurate answers based on live sensor data or updated online agricultural knowledge.
- Executing device-level actions in real-time via text commands.
- Improving the overall responsiveness, intelligence, and usability of the greenhouse system.

4.3.3 AI chatbot

4.3.3.1 Overview

In the smart greenhouse system, a Gemini model (a large language model developed by Google) is used as the central intelligence that interprets user questions and decides how to handle them. When a user sends a message, the Gemini model analyzes its content and determines which of the available tools is most appropriate to use. These tools include controlling IoT devices, retrieving information from a PDF-based knowledge base, accessing sensor data from a Google Spreadsheet, or searching the internet for agriculture-related content. The Gemini model functions like a smart router—it understands the intent behind the question and forwards it to the corresponding tool, then uses the tool's output to generate a relevant and context-aware response. This approach ensures that each user query is handled efficiently and accurately, depending on its nature.

4.3.3.2 Tools

To extend the capabilities of its core reasoning engine, the chatbot assistant is equipped with a set of external tools. These tools allow it to interact with the physical greenhouse environment, access structured information, and respond to user queries with meaningful and actionable insights.

4.3.3.2.1 IoT devices control

This tool enables the chatbot to issue commands to connected IoT devices within the greenhouse, such as turning on or off the heating lamp, water pump, or ventilation fan. Through this interface, users can control the environment directly by sending text commands.

4.3.3.2.2 PDF knowledge base access

The chatbot is connected to a local knowledge base derived from a structured PDF file that contains detailed information about the project, development team, and system documentation.

4.3.3.2.3 Sensor data monitoring

This tool allows the chatbot to fetch real-time environmental data (such as temperature, humidity, or soil moisture) from sensors deployed in the greenhouse. The sensor readings are stored in a cloud-based Storage, which the chatbot can query to

provide live updates. For example, a user might ask “What is the current temperature?” and receive a precise answer from the latest data entry.

4.3.3.2.4 Online agricultural search

To expand its knowledge beyond the local database, the chatbot includes a search tool for retrieving agriculture-related information from the internet. When users ask more general or uncommon questions—such as “How to treat tomato leaf curl?”—the chatbot performs a live search and returns summarized answers or relevant links, enhancing its usefulness in real-world farming scenarios.

4.3.4 Technologies used

The chatbot assistant is built using **Google's Gemini Pro model** for natural language understanding, question classification, and response generation in Arabic. It leverages the **LangChain framework** to manage conversation flow, memory, and routing logic between tools. To retrieve real-time environmental data, it connects to **Google Sheets** via the **gspread** library and analyzes it using **pandas**. For answering questions about the project and team, it extracts and searches content from a local PDF using **PyMuPDF (fitz)** and text_splitter from LangChain. IoT device control is handled by interpreting structured prompts through the Gemini model. For broader agricultural questions, the chatbot performs **live web searches** using the **Google Custom Search API**, evaluates page content using **BeautifulSoup** and Gemini, and returns summarized answers. The system is deployed as a **Flask web server** with **CORS** enabled to allow communication with the frontend.

4.4 ML model for Identifying Tomato Growth Stages

4.4.1 The Problem

Traditional agriculture relies on human observation to determine the growth stages of tomatoes, such as when to irrigate, fertilize, or adjust lighting. However, this method is often inaccurate and heavily depends on the farmer’s experience. As a result, issues like Over-irrigation, Under-fertilization and incorrect lighting can arise which negatively impact both crop yield and quality.

Additionally, resources like water and energy are often wasted due to inaccurate assessments of the plants' needs. For instance, over-watering or under-watering may occur if the growth stage is not identified properly.

4.4.2 Why do we need an ML model?

The ML model will predict the plant's growth stages based on image, allowing the smart system to automatically adjust conditions to ensure the best growing environment.

1- Growth Stage Prediction: predicts the current growth stage depending on the plant image.

2-Automatic System Adjustment: The IoT system will automatically modify lighting, irrigation, and temperature based on the model's predictions.

3- Optimized Plant Environment: These automatic adjustments will create ideal growing conditions, leading to better productivity and reduced waste.

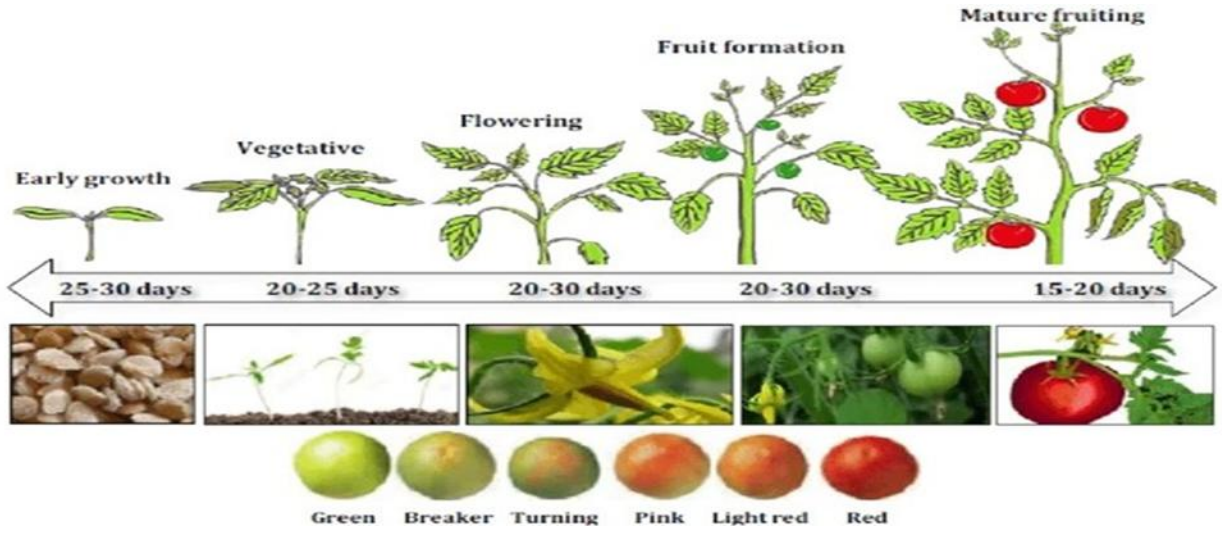


Figure 4.6 tomato growth stages

4.4.3 Tomato Growth Stages & Timelines

4.4.3.1 Seed Germination

Tomato seeds typically germinate within 5 to 10 days under optimal conditions. However, this timeframe can vary based on the temperature and humidity of the environment. The key to successful germination is maintaining a consistent temperature between 70-80°F (21-27°C) and providing moderate humidity. Seeds require a moist (not waterlogged) environment to sprout effectively.

Tips for improving germination rates

- **Use fresh seeds:** Fresh seeds have higher germination rates. Check the expiration date or conduct a germination test if you're unsure about their viability.
- **Pre-soak seeds:** Soaking seeds in warm water for 12-24 hours before planting can help speed up germination.
- **Maintain optimal soil temperature:** Use a heat mat under seed trays to keep the soil within the ideal temperature range.
- **Cover seed trays:** Covering seed trays with a plastic dome or wrap can help retain moisture and warmth, essential for germination.

4.4.3.2 Seedling

After germination, tomato plants enter the seedling stage, characterized by the development of their first true leaves beyond the initial seed leaves (cotyledons). This stage is crucial for building a strong foundation for future growth.

Required Days: 2-3 weeks post-germination.

How to care for tomato seedlings

- **Provide sufficient light:** Tomato seedlings require 14-16 hours of direct light each day. Use grows lights if natural sunlight is insufficient.
- **Water carefully:** Keep the soil consistently moist but avoid overwatering, as it can lead to fungal diseases.
- **Temperature control:** Maintain a daytime temperature of 65-70°F (18-21°C) and a slightly cooler temperature at night.
- **When to transplant seedlings outdoors:** Transplant seedlings outdoors after the last frost when they have at least two sets of true leaves and night temperatures consistently stay above 50°F (10°C).

4.4.3.3 Vegetative Growth

This stage focuses on the development of stems and leaves. The plant builds its framework and increases in size, preparing for flowering and fruiting.

Required Days: Extends until the first flowers open, typically 6-8 weeks.

The importance of nutrients and water during this stage

- **Nutrients:** A balanced fertilizer, rich in nitrogen and potassium, supports healthy leaf and stem growth.
- **Water:** Consistent, deep watering encourages strong root development. Avoid wetting the leaves to reduce disease risk.

Determinate tomatoes grow to a specific size (determined by variety) before transitioning to flower production.

4.4.3.4 Flowering and Fruit Set

Tomato plants typically begin to flower transplanting, depending on the variety and growing conditions. Flowers must be pollinated to produce fruit. While tomatoes are self-pollinating, encouraging pollinator activity or gently shaking the flowering branches can improve the fruit set.

Required Days: Lasts 2-3 weeks before fruit sets.

How to encourage healthy fruit set

- Optimal temperatures: Ensure daytime temperatures are between 70-85°F (21-29°C) for best pollination.
- Adequate watering: Consistent moisture is crucial during flowering and fruit set. Avoid fluctuations that can lead to blossom drop or poor fruit development.

4.4.3.5 Fruit Development and Maturity

Tomatoes start out green and tend to stay this color until they reach their mature size. Most varieties turn bright red when 100% ripe. Since some cultivars bear yellow, purple, or even green mature fruit, though, it's important to familiarize yourself with the type of tomatoes you're growing.

Required Days: Fruit develops over 20-30 days post-flower; ripening time varies by variety.

4.4.4 The ML Model

4.4.4.1 Overview

The key growth stages of tomatoes include the seedling, flowering, fruiting, and maturity phases. Accurate identification of these stages plays a crucial role in improving crop management and enhancing production efficiency in protected agricultural environments.

Traditional methods for monitoring tomato growth rely on manual observation and practical experience, which are inefficient, costly, and prone to human error. In recent years, artificial intelligence (AI) and deep learning technologies have been introduced into smart agriculture to improve the accuracy and efficiency of automated growth stage classification.

4.4.4.2 Dataset

Data for this project was collected from multiple sources, comprising 1,103 images containing 2,766 objects. Images representing the first two stages of growth were taken directly from the tomato farm, and the remaining stages were obtained from other sources such as Kaggle

COLOR	CLASS NAME	COUNT
	Flowering	563
	Fruit_and_Ripening	573
	Germination	543
	Seeding	520
	Vegetative	567

Figure 4.7 Data classes and number of objects per class

4.4.4.3 YOLOv9 model [9]

In this project, the YOLOv9 (You Only Look Once, version 9) model was used to predict the growth stages of tomato plants based on images captured within a protected agricultural environment. YOLOv9 is a state-of-the-art object detection model that combines high accuracy with real-time processing capabilities, making it well-suited for smart agriculture applications. The model was trained on the dataset described above. By accurately identifying these stages, the system can support automated decision-making for environmental controls such as lighting, irrigation, and temperature regulation, leading to improved crop management and increased production efficiency.

The overall model performance achieved a precision of 0.808, recall of 0.73, and mAP@0.5 of 0.806 with mAP@0.5:0.95 reaching 0.527.

Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%
all	171	263	0.808	0.73	0.806	0.527
Flowering	23	77	0.758	0.844	0.853	0.409
Fruit_and_Ripening	28	44	0.793	0.783	0.85	0.561
Germination	23	31	0.716	0.57	0.69	0.467
Seeding	55	71	0.881	0.62	0.745	0.537
Vegetative	37	40	0.893	0.833	0.892	0.661

invalid value encountered in less
invalid value encountered in less

Figure 4.8 model performance

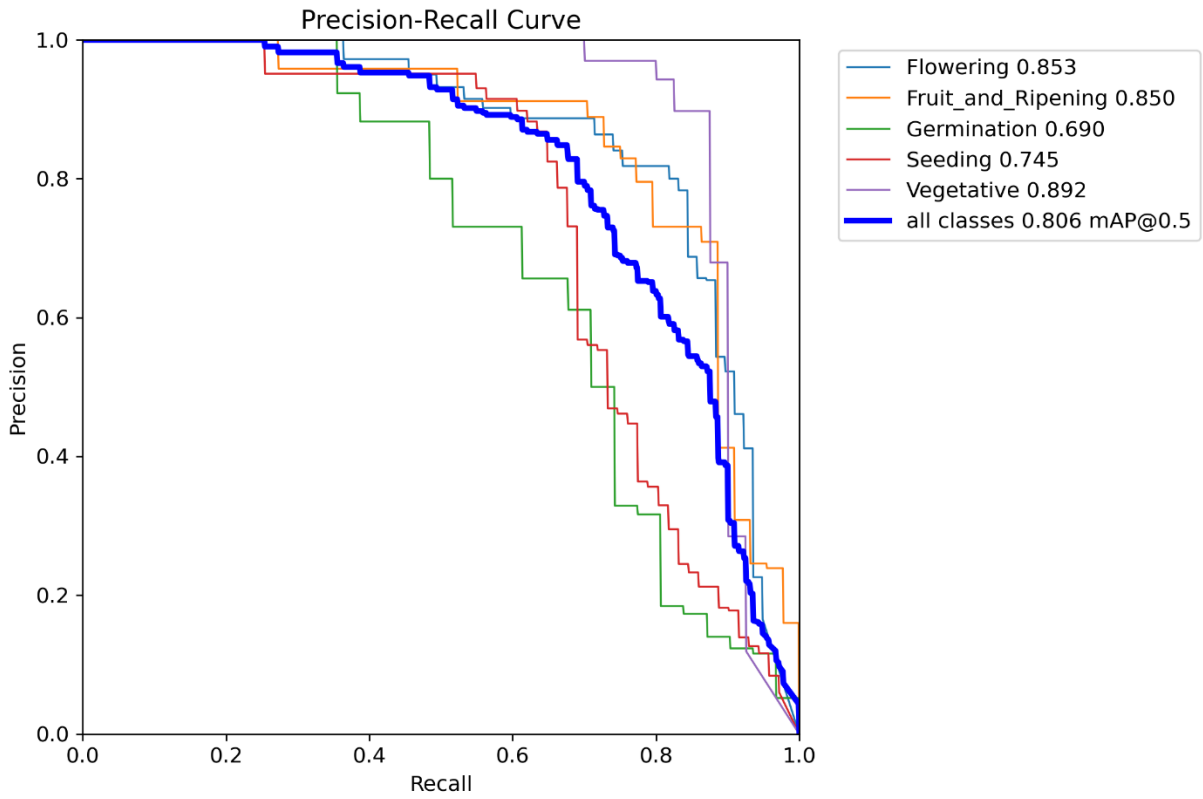


Figure 4.9 Precision-Recall Curve

4.5 AI Workflow

1. Image Capture and Upload

The ESP-CAM module captures an image of the tomato plant every hour. Each captured image is automatically uploaded to a cloud-based storage solution

2. Image Storage

The uploaded images are stored in Azure Storage. This serves as a central repository for image data. Currently, the system operates using free-tier cloud services, with plans to transition to Azure infrastructure for the final product.

3. Disease Detection and Classification

A Python script hosted in the cloud runs a machine learning (ML) model that processes the stored images. This model performs the following **tasks**:

1-Growth Stage Prediction: Determines the plant's current growth stage based on the uploaded image.

2-Disease Detection: Assesses the plant's health status by determining whether it is healthy or affected by one of the eight specific diseases the model has been trained to recognize.

4. Notification and Reporting

The machine learning model processes the analysis results by uploading the data to a cloud-based storage unit. The mobile application retrieves this data, ensuring accurate and timely updates. Users receive real-time notifications that include:

1-The current growth stage of the plant, allowing the smart system to automatically adjust environmental conditions for optimal growth.

2-The plant's health status and any detected diseases, enabling immediate action to ensure plant health.

4.6 Future Plan

4.6.1 Irrigation model [13][10]

Overview:

The integration of IoT and AI technologies has become increasingly prevalent in developing optimized irrigation models. These advanced models leverage a variety of environmental parameters, including:

- Humidity
- Temperature
- soil moisture

rather than relying on a single variable. By incorporating multiple factors, they enable more informed and precise decision-making, ultimately enhancing water use efficiency in agricultural practices.

Challenge:

However, a significant challenge in implementing such models is the scarcity of available data, which hampers the training and validation of these systems.

Expected:

The primary output of these models is the determination of the optimal quantity of water required for irrigation, ensuring sustainable water resource management while maintaining crop health and productivity.

Chapter 5: Data Analysis

5.1 Monitoring

5.1.1 Introduction

The analysis for monitoring a smart greenhouse system offers significant benefits by providing real-time insights and data visualizations, enabling users to monitor key metrics like temperature, humidity, and soil moisture. With customizable filters and interactive visualizations, users can easily analyze trends, identify patterns, and make data-driven decisions to optimize greenhouse conditions. The dashboard enhances resource efficiency and helps users take proactive actions in response to alerts for abnormal conditions. While its user-friendly interface makes it accessible to both technical and non-technical users. Ultimately, the Power BI dashboard improves operational efficiency, contributing to better decision-making and higher productivity in greenhouse operations.

5.1.2 Analysis factors:

The analysis focuses on several key factors, including humidity, temperature, electricity consumption (which will be discussed in detail in section 1.4, and soil moisture for each type of soil. Line charts were selected as the primary visualization tool for this data because they effectively capture changes over time, highlight emerging patterns and trends, and make it easier to identify outliers and anomalies.

5.1.3 Real-time analysis

The Power BI dashboard page dedicated to real-time analysis provides a comprehensive view of the temperature, humidity, soil moisture, and electricity usage over the past 24 hours. This page features dynamic line charts for each metric, allowing users to visually track fluctuations throughout the day and identify immediate trends or anomalies. By displaying these factors in real-time, the dashboard enables users to assess the current environmental conditions within the greenhouse and quickly respond to any irregularities,

such as temperature spikes or sudden drops in humidity. The inclusion of electricity usage helps monitor resource consumption, offering insights

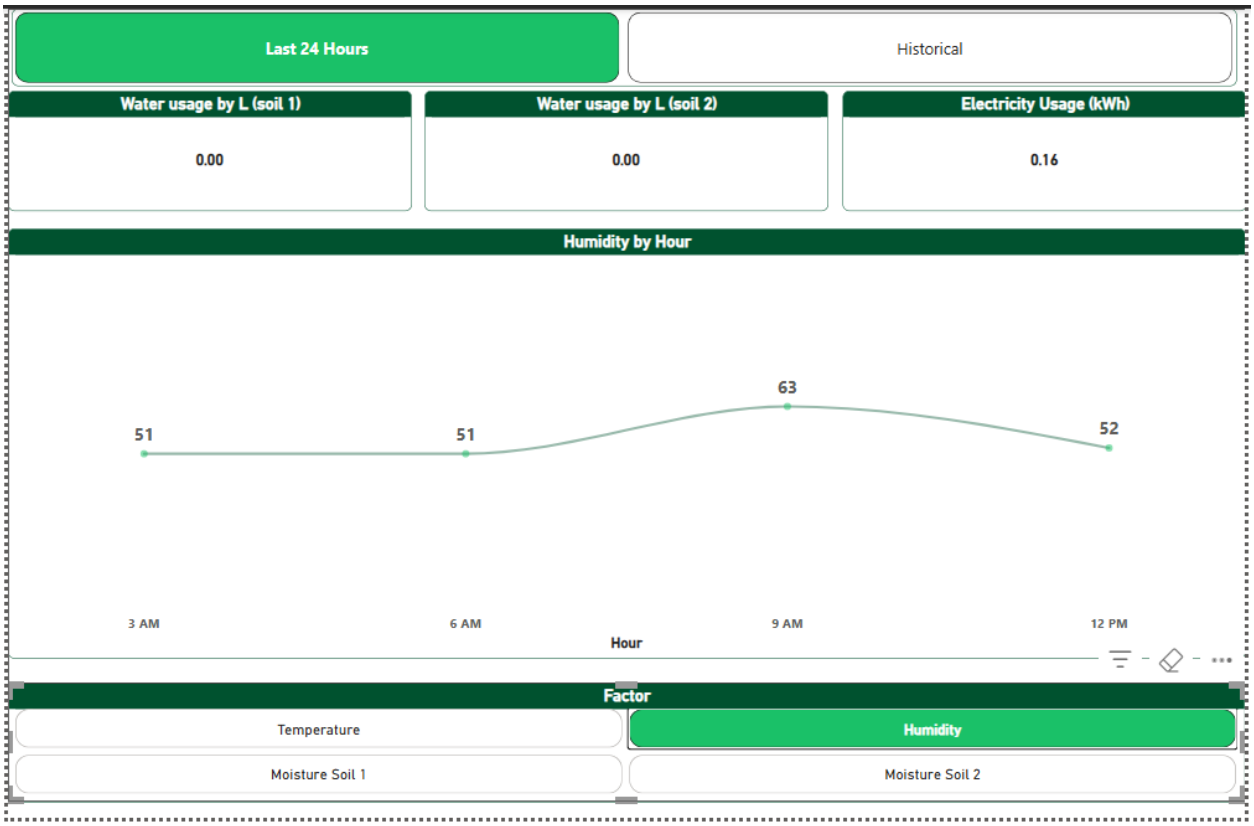


figure 5.1 real time monitoring

into efficiency and potential savings. This real-time analysis empowers greenhouse managers to make informed decisions such as managing energy usage, all of which contribute to more efficient and sustainable greenhouse operations. (figure1.1)

5.1.4 Historical analysis

The historical analysis page in the Power BI dashboard allows users to explore the temperature, humidity, soil moisture, and electricity usage data over extended time periods, offering deeper insights into trends and patterns. By utilizing a slicer, users can easily filter the data by specific timeframes, such as daily, or monthly intervals, enabling them to compare conditions across different seasons, growth cycles, or operational periods. This feature helps identify long-term trends, such as seasonal variations in temperature or humidity levels, and assess how environmental factors correlate with electricity usage over

time. With the ability to view historical data alongside real-time metrics, greenhouse managers can make more informed decisions for future planning, optimize resource management. (figure 1.2)

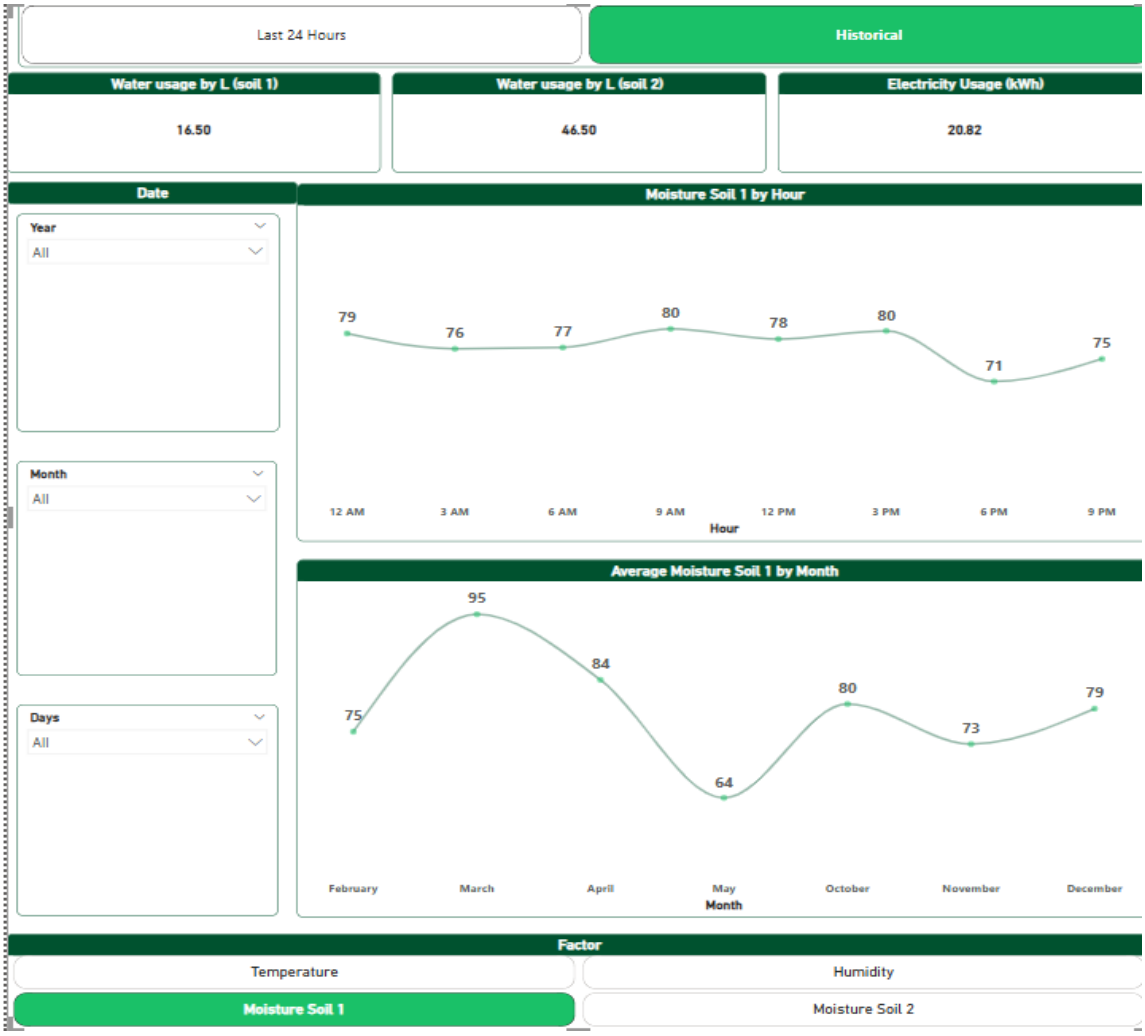


Figure 5.2 Historical data analysis

5.2 Predicting Electricity

5.2.1 Introduction

Monitoring the system environment, including factors such as temperature, humidity, and soil moisture, is essential for optimizing system performance and ensuring user satisfaction. However, it is equally important to track the electricity usage of the

system. By monitoring energy consumption , users can better manage resources, improve efficiency, and make informed decisions about system operation. Furthermore, this data can be leveraged to predict future electricity bills, also water usage in Liter enabling cost-effective management of the system's energy requirements.

5.2.2 System Overview

The system sending the data every hour with the number of minutes that the devices work.

Devices that use electricity in the system includes:

- Air Conditioning (AC): Consumes 40W.
- Heat lamp: Consumes 75W.
- Light bulb: Consumes 12W.

5.2.3 Power Consumption Calculation

$$Air\ Conditioning\ usage = \frac{total\ operating\ time\ in\ minutes}{60} \times 40W$$

$$Heat\ Lamp\ usage = \frac{total\ operating\ time\ in\ minutes}{60} \times 75W$$

$$light\ bulb\ usage = \frac{total\ operating\ time\ in\ minutes}{60} \times 12W$$

5.2.4 Total Power Consumption

The total power consumption can be calculated by summing the power used by each and divided by 1000 to get the usage in terms of kWh.

$$Total\ Power\ Consumption\ in\ kWh = \frac{Air\ Conditioning\ usage + Heat\ Lamp\ usage + Light\ Bulb\ usage}{1000}$$

5.2.5 predict Power Consumption

Accurately forecasting future electricity demand is essential for maintaining stable and efficient operation of our smart agriculture system, especially when resources like energy must be optimized under changing environmental conditions.

5.2.5.1 Challenges with Historical Data

In our case, the collected historical electricity usage data is not sufficiently stable or consistent to support traditional time-series forecasting models. The system is still in a development and testing phase, meaning usage patterns vary significantly and do not yet represent a stable, real-world operational profile. This makes time-series models unreliable, as they assume regularity and consistency in the data over time.

5.2.5.2 Alternative Approach Using Weather Forecasting APIs

To overcome the limitations of historical data, we adopted an alternative, context-aware forecasting method. Instead of relying purely on past usage trends, we use real-time and forecasted environmental data—especially weather data—obtained from a public weather API.

Key weather parameters used include:

Temperature

Sunrise and sunsit

This factor influence the operation of various components in our system, such as:

- Cooling and heating systems in greenhouses
- Lighting systems

By analyzing upcoming weather conditions, we can predict how much energy the system will likely consume under those environmental scenarios. For example, hotter days may require more energy for cooling, while cloudy days may increase dependency on artificial lighting.

5.2.5.3 Accuracy and Practical Use

While this method is not perfectly accurate, it provides a reasonable and practical way to estimate energy needs under expected conditions. This approach:

- Helps anticipate system behavior in upcoming days.
- Assists in resource allocation and load balancing.
- Supports fail-safe design by identifying weather scenarios where the system might be under strain.

Although not as precise as models trained on large and clean historical datasets, this technique offers the best available solution for predictive energy management in the current project phase.

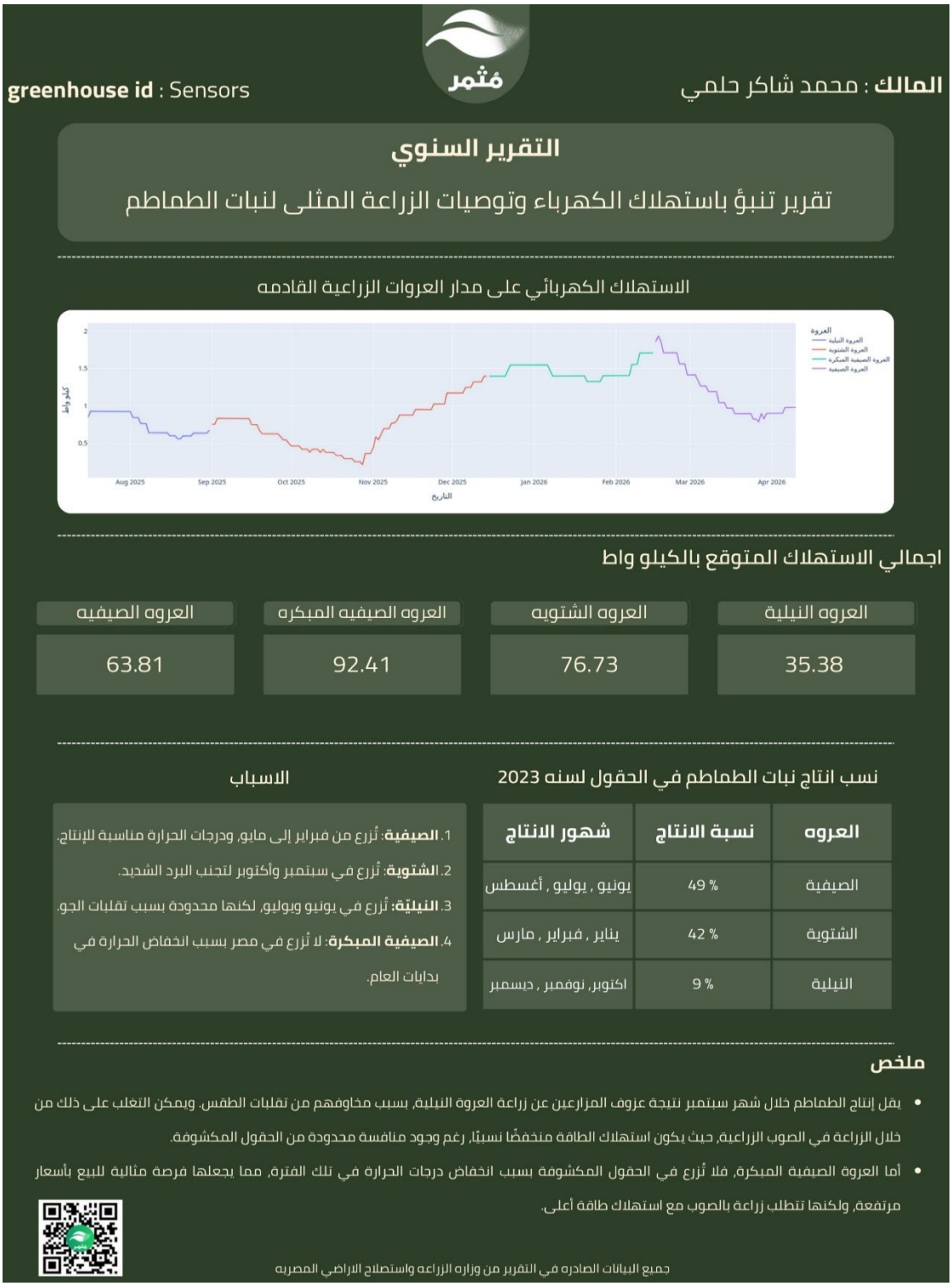


Figure 5.3 power report

5.3 Future plane

5.3.1 Predicting Water Usage

We can follow the same steps we have done in electricity to predict the water consumption in Liters

- Water pump flow rate = 90L / H
- Calculate the total water consumption daily
- Train a time-series model on the historical water consumption data
- Predict the water consumption.

5.3.2 Time-Series Data Analysis

To predict future electricity usage, you wil use time-series analysis. Here's a step-by-step breakdown of how you might approach this:

1. **Data Preprocessing:**
    - Clean and format historical data to ensure that it's ready for analysis.
    - Handle missing or incomplete data (e.g., by interpolation or removing gaps).
2. **Feature Selection:**
   Select electricity usage aggregated by month and the growth stage
3. **Modeling Electricity Usage Patterns:**
    - Seasonal Autoregressive Integrated Moving Average with exogenous variables (SARIMAX): A popular method for time-series forecasting, useful if you have a continuous time-series data of electricity consumption with seasonal trends.
4. **Training the Model:**
    - Train forecasting model on the historical electricity usage data from the sensors, with features such as time
5. **Prediction of Future Electricity Usage:**
   Once trained, the model can predict future electricity usage. For example, based on past data, it can predict the expected electricity consumption for the next day, week, or month, given the patterns of usage.

Chapter 6: Software development

6.1 Mobile Features

1. Real-Time Monitoring

- Track vital greenhouse conditions such as:
    - Temperature
    - Humidity
    - Soil moisture levels
    - Light intensity
- Get live updates from connected IoT sensors directly to your mobile device.

2. Intelligent Control

- Remotely manage key greenhouse systems, including:
    - Irrigation: Automate watering schedules or trigger manual irrigation.
    - Cooling & heating: Control fans and heating lamps to maintain optimal temperature.
    - Lighting: Adjust artificial lights to suit your crop needs.
- Set predefined rules to automate responses to environmental changes.

3. Data Analytics, Insights, and AI help

- Access historical data to monitor data and analyze greenhouse performance over time.
- Use graphical representations to make informed decisions about crop health and productivity.
- Get more advanced help with Muthmir AI

4. Community and Knowledge Sharing

- Join a community of smart farmers to:
    - Share experiences, tips, and best practices.
    - Discuss challenges and solutions in smart agriculture through posts or chat.

- Access expert advice and curated content on sustainable farming techniques.

5. Multi-Greenhouse Support

- Manage multiple greenhouses from a single app, each with individual configurations.

6. User-Friendly Design

- Simple and intuitive interface for users of all technical levels.
- Easy setup and login process to start monitoring instantly.

7. Security and Privacy

- Ensure secure communication with IoT devices using encryption.
- Protect user data and greenhouse information with advanced privacy features.

8. Cross-Platform Compatibility

- Available for both Android and iOS, providing a consistent experience across devices.

6.2 technology used

6.2.1 Comparison of Mobile Development Frameworks

Framework	Language	Cross-Platform Support	Performance	Community Support	Hot Reload	Native Features Access	Ease of Learning
Flutter	Dart	Android, iOS, Web, Desktop	Near-native performance	Large and active (by Google)	Yes	Strong with customizable widgets	Moderate
React Native	JavaScript	Android, iOS	Good	Very large (by Meta)	Yes	Good with third-party libraries	Easy for JS developers
Ionic	JavaScript, TypeScript	Android, iOS, Web	Moderate (WebView-based)	Large (open-source)	Yes	Relies on plugins	Easy for web developers
Xamarin	C#	Android, iOS,	Near-native	Moderate (by	Yes	Strong, especially for	Moderate

		Windows	performance	Microsoft)		Microsoft products	
Swift/Objective-C (Native iOS)	Swift/Objective-C	iOS only	Excellent (Fully native)	Smaller (iOS-specific)	No	Full access (iOS only)	Moderate-hard
Kotlin/Java (Native Android)	Kotlin/Java	Android only	Excellent (Fully native)	Smaller (Android-specific)	No	Full access (Android only)	Moderate-hard

6.2.2 Why We Used Flutter in the Muthmir Project

For the Muthmir project, Flutter was chosen because it provides a single codebase that works seamlessly across multiple platforms, including Android, iOS, and even the web. This reduced the development time and ensured consistent performance across devices. Flutter's customizable widgets allowed us to create a user-friendly and visually appealing interface tailored to the needs of greenhouse management.

Additionally, Flutter's hot reload feature significantly boosted our development speed by enabling real-time UI updates during coding. With Google's backing and a strong developer community, Flutter ensured stability, continuous support, and easy integration with Firebase, which powers our backend services like authentication and data management. In summary, Flutter offered the perfect blend of performance, scalability, and ease of development for Muthmir.

6.2.3 Version Control

we used GitHub due to its efficiency in managing code collaboration among team members. GitHub provides features like pull requests, issue tracking, and branch management, which allowed us to work seamlessly on different parts of the project while maintaining code integrity. It also serves as a central repository for storing our project, ensuring that all team members have access to the latest version of the code.

6.2.4 Backend

The backend infrastructure leverages a combination of Firebase for authentication and a custom server setup for core functionality, utilizing HTTP/S and MQTT protocols for communication.

Authentication with Firebase:

Firebase Authentication is employed to handle user identity management securely and efficiently. It provides a robust, scalable solution for user authentication, supporting

multiple authentication methods such as email/password, phone number, and social media logins (Google). Firebase Authentication ensures secure user sessions with JSON Web Tokens (JWTs), which are validated on the client side and passed to the backend server for authorization. This setup simplifies user management, supports seamless integration with Firebase's security rules, and ensures scalability for handling large user bases.

Server Communication via HTTP/S

The mobile app communicates with our custom backend server primarily through HTTP/S for RESTful API interactions. The server, hosted on a secure cloud infrastructure, handles requests for data retrieval, updates, and other CRUD operations. API endpoints are designed to be stateless, ensuring scalability and reliability. HTTPS ensures all data transmitted between the mobile app and the server is encrypted, protecting sensitive user information and maintaining data integrity. The REST APIs are optimized for performance, using JSON as the data interchange format for lightweight and efficient communication.

Real-Time Communication with MQTT

For real-time, event-driven communication, the backend utilizes the MQTT (Message Queuing Telemetry Transport) protocol. MQTT is a lightweight, publish-subscribe-based messaging protocol ideal for mobile applications due to its low bandwidth and power consumption. The app connects to an MQTT broker hosted on our server, allowing it to subscribe to specific topics for receiving real-time updates, such as notifications, live data feeds, or system status changes. The broker ensures reliable message delivery with Quality of Service (QoS) levels, enabling efficient and scalable real-time interactions. TLS encryption is applied to MQTT connections to secure data in transit.

6.3 Software Development Process

6.3.1 Mobile Application

The mobile app is an integral part of the Smart Farming System, providing users with a convenient way to interact with their farming operations and connect with other community members. It allows users to monitor real-time sensor data, which is collected from IoT devices deployed in the field. The app enables remote control of farming devices, such as irrigation systems and temperature regulators, providing flexibility and efficiency in managing agricultural tasks.

6.3.2 Software process model choosing

Methodology: Agile with Scrum

The development team has adopted the Scrum framework to guide the software development process. Although Scrum typically involves a dedicated Product Owner, the team collaboratively assumed this role. Requirements were directly gathered through interviews, and discussions with prospective users and stakeholders, ensuring that the app aligns with the target audience's needs and project goals.

Key Aspects of the Scrum Framework:

- **Sprints**: The development is divided into **2-week sprints**, ensuring regular delivery of functional increments.
- **Scrum Roles**:
    - **Product Owner**: Team collaboratively assumed this role.
    - **Scrum Master**: Facilitates the Scrum process, ensuring the team adheres to best practices.
    - **Development Team**: Focused on delivering the tasks planned for each sprint.
- **Scrum Events**:
    - **Sprint Planning**: Held at the beginning of each sprint to define tasks.
    - **Daily Scrum:** A short daily meeting to track progress and address blockers.
    - **Sprint Review**: Conducted at the end of the sprint to showcase the deliverables.

Stages of Development Using Scrum:

1. **Backlog Creation**:
    - Identify and prioritize all app features and tasks in the product backlog.
2. **Sprint Execution**:
    - Deliver increments of functionality through 2-week sprints.
3. **Review and Feedback**:
    - Demonstrate sprint deliverables to stakeholders and incorporate feedback.

Requirement Gathering Approach:

- The team conducted interactive sessions with farmers, agricultural experts, and other stakeholders to understand the challenges they face and the features they would value most.
- feedback mechanisms were employed to prioritize app functionalities based on their practicality and impact on farming operations.
- By handling the requirements gathering process ourselves, we ensured a user approach and direct alignment with stakeholder expectations.

Functional Requirements:

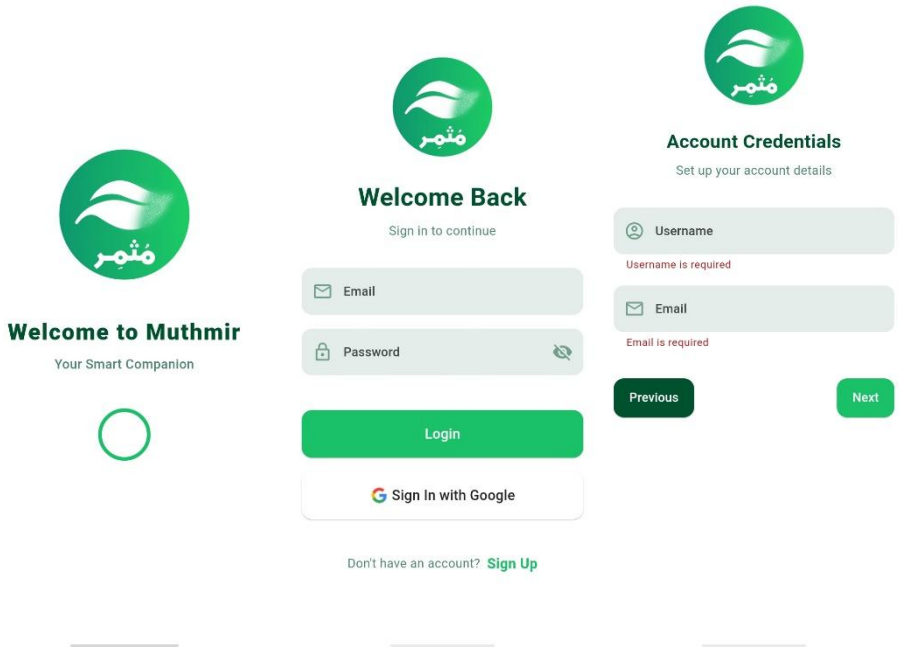
- Users can sign up, sign in, edit profiles, and log out (from settings page).
    - **Edit Profile**: Users can change their username and profile picture.
- Greenhouses are assigned a random ID by the admins
- User can choose to stay signed in using a checkbox in sign in page
- Real-time sensor data is displayed on dashboards.
- User can click a sensor to view all data for the selected date
- User can choose which date to view its data
- User can reorder sensors and group them in collections
- Owned greenhouses are shown in the sidebar
- Add/remove a new greenhouse from the sidebar
- User can choose a grid view or row view for sensors data
- Latest disease AI output is displayed in the health page (image and class)
- Upload a picture for AI to check
- Users can control farming devices remotely.
- AI chatbot
- Allow chatbot control for devices
- Community section supports posting, commenting, reacting, deleting posts, and user chat.
    - **Post**: Users can include attachments (e.g., images, documents) in their posts.
    - **Chat**: Accessible from feed page. A private chat between to users
    - **Search:** Search other users
    - **Follow/Unfollow:** Follow/unfollow users from their profile page
- Notifications for device activity and community interactions.

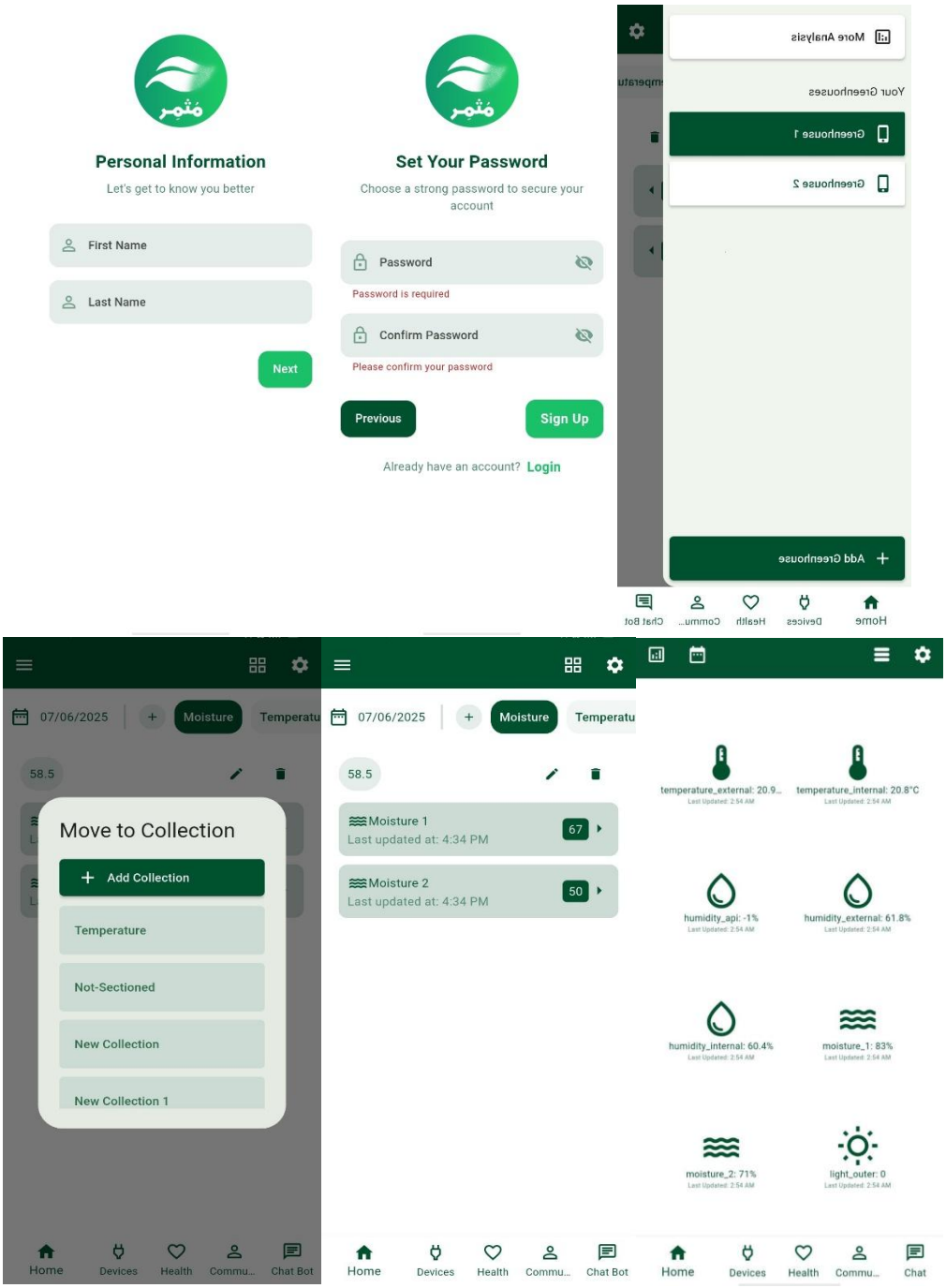
Non-Functional Requirements:

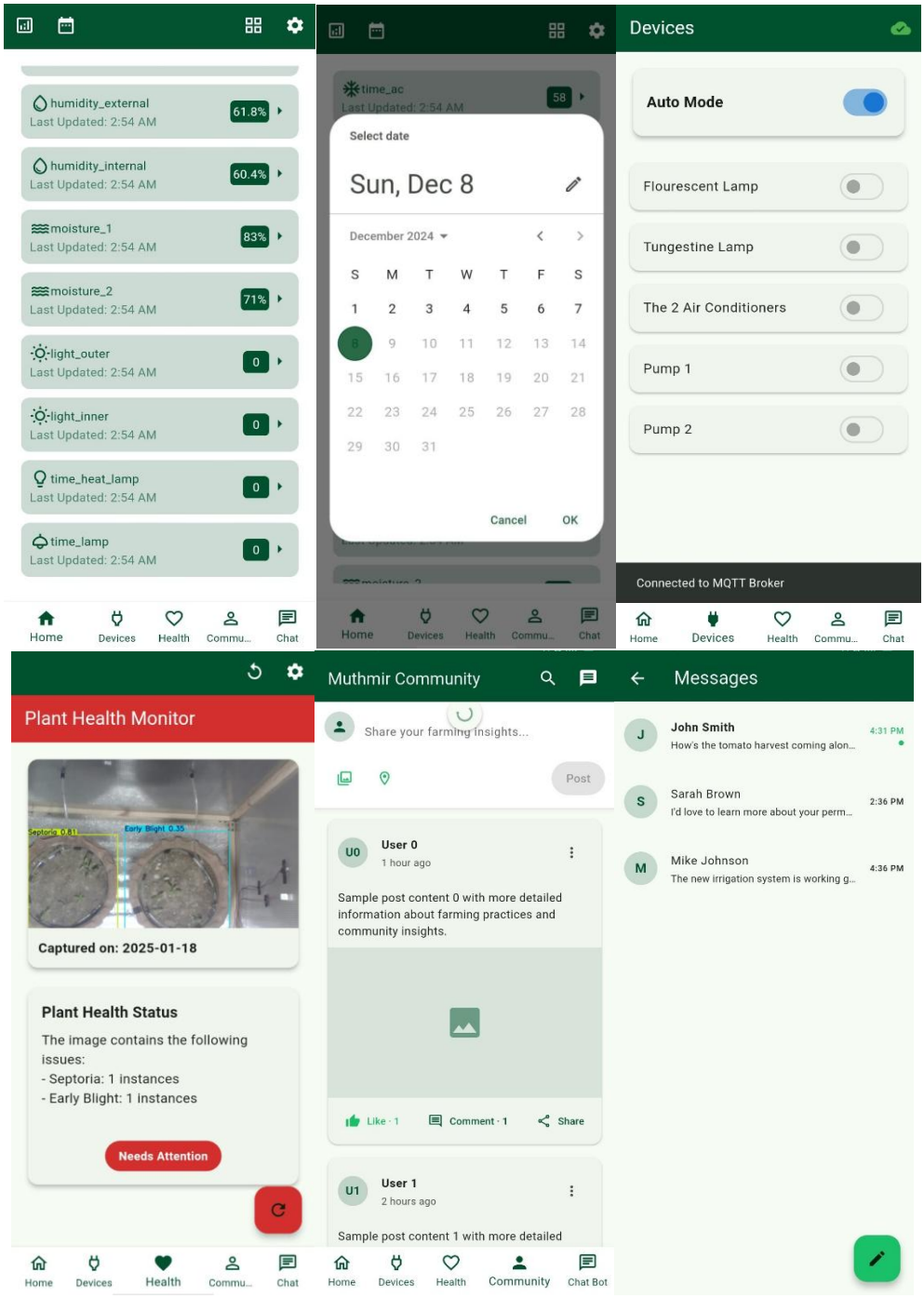
- The app must be responsive and work seamlessly on Android, iOS devices, and web.
- Data updates (sensor data) should have a latency of no more than 2 seconds.

- The app must support the Arabic language, with the ability to switch between Arabic and other supported languages.
- The system must handle up to 10,000 concurrent users.
- Ensure data security with encryption and secure authentication protocols.
- A help page must be included, detailing all app features and providing user guidance for functionality.

6.3.3 Design and Features




Personal Information
Let's get to know you better
First Name
Last Name
Next
Set Your Password
Choose a strong password to secure your account
Password
Password is required
Confirm Password
Please confirm your password
Previous
Sign Up
Already have an account? Login
Move to Collection
Add Collection
Temperature
Not-Sectioned
New Collection
New Collection 1
07/06/2025
Moisture
Temperatu
58.5
Moisture 1
Last updated at: 4:34 PM
67
Moisture 2
Last updated at: 4:34 PM
50
temperature_external: 20.9...
temperature_internal: 20.8°C
humidity_api: -1%
humidity_external: 61.8%
humidity_internal: 60.4%
moisture_1: 83%
moisture_2: 71%
light_outer: 0
Home
Devices
Health
Commu...
Chat Bot
Chat


humidity_external
Last Updated: 2:54 AM
61.8%
humidity_internal
Last Updated: 2:54 AM
60.4%
moisture_1
Last Updated: 2:54 AM
83%
moisture_2
Last Updated: 2:54 AM
71%
light_outer
Last Updated: 2:54 AM
0
light_inner
Last Updated: 2:54 AM
0
time_heat_lamp
Last Updated: 2:54 AM
0
time_lamp
Last Updated: 2:54 AM
0
Home
Devices
Health
Commu...
Chat
time_ac
Last Updated: 2:54 AM
58
Select date
Sun, Dec 8
December 2024
S M T W T F S
1 2 3 4 5 6 7
8 9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
Cancel
OK
Devices
Auto Mode
Flourescent Lamp
Tungestine Lamp
The 2 Air Conditioners
Pump 1
Pump 2
Connected to MQTT Broker
Plant Health Monitor
Captured on: 2025-01-18
Plant Health Status
The image contains the following issues:
- Septoria: 1 instances
- Early Blight: 1 instances
Needs Attention
Muthmir Community
Share your farming insights...
Post
U0
User 0
1 hour ago
Sample post content 0 with more detailed information about farming practices and community insights.
Like · 1
Comment · 1
Share
U1
User 1
2 hours ago
Sample post content 1 with more detailed
Community
Chat Bot
Messages
J
John Smith
How's the tomato harvest coming alon...
4:31 PM
S
Sarah Brown
I'd love to learn more about your perm...
2:36 PM
M
Mike Johnson
The new irrigation system is working g...
4:36 PM

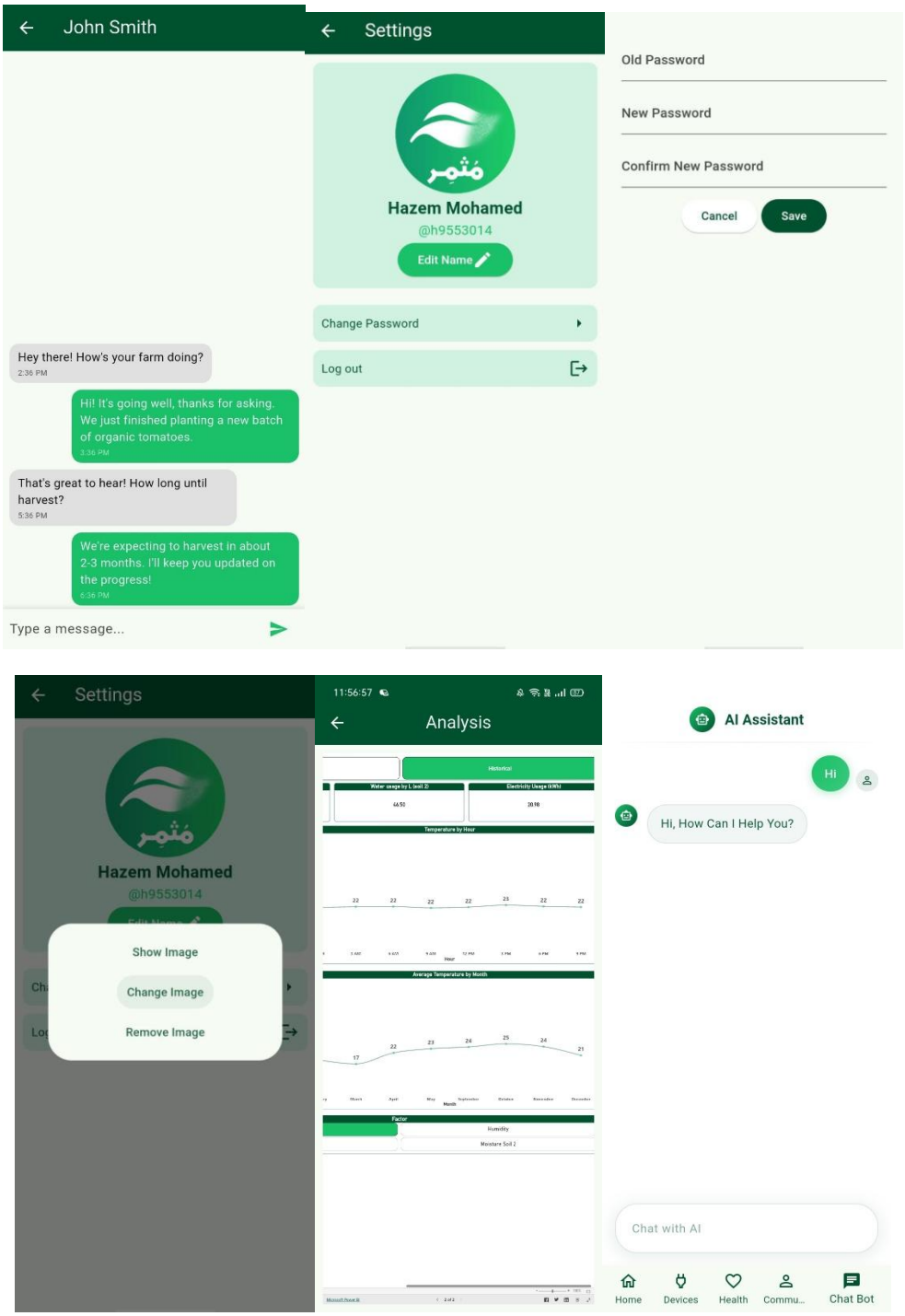


Figure 6.1: Design

6.3.4 Development Timeline:

Requirement Analysis (2 sprints):
Document functional and non-functional requirements and finalize the tech stack and architecture.

UI/UX Design (2 sprint)

Sprint 1:

- Create wireframes for app screens: dashboard, community section, profile, settings, and post creation.

Sprint 2:

- Create mockups for the design of the community section, profile editing, and device control interfaces.
- Gather initial user feedback on wireframes and mockups.

Development (5 sprints)

Sprint 1:

- Backend: Set up the database structure and configure authentication protocols.
- Backend: Build APIs for user management (sign up, sign in, edit profile, and logout).
- Frontend: Develop the login/sign-up screens and user profile page.

Sprint 2:

- Backend: Implement APIs for sensor data collection and control of farming devices.
- Frontend: Integrate sensor data display into the dashboard.
- Frontend: Develop the dashboard for displaying real-time sensor data.

Sprint 3:

- Frontend: Implement the community section layout.
- Backend: Implement APIs for community interactions (posting, commenting, reacting, deleting).
- Frontend: Implement post creation, including attachments (e.g., images, documents).
- Frontend: Design and implement the notifications feature.

Sprint 4:

- Backend: Add attachment support to posts.

- Backend: Refine APIs for sensor data and device control, ensuring real-time updates.
- Frontend: device control features in the app.
- Frontend: Implement the post/comment/like/react/delete features.

Sprint 5:

- Backend: Implement final testing and optimization of APIs.
- Frontend: Finalize UI for profile editing, community features, and device control screens.
- Implement full integration of all features, ensuring the app works seamlessly across all screens.
- Integrate Arabic language support into the user interface.

Integration and Testing (3 sprints)

Sprint 1:

- Conduct unit testing for individual features (login, sign-up, dashboard, profile edit, post features, device control).

Sprint 2:

- Conduct integration testing between frontend and backend systems.
- Perform usability testing, ensuring the user interface is intuitive and works as expected.

Sprint 3:

- Test the app's responsiveness and real-time data display under various conditions.
- Finalize any bug fixes, optimize performance, and prepare for deployment.

6.3.5 Risk management

Risk	Type	Probability	Effect	Planning
Team members in different courses with different exams' schedule might hinder the project's flow	People	High	Serious will Hinder the project's flow	Consider each member's availability and assign tasks due to their availability and tasks' priority
Not all the team members familiar with used frameworks like Flutter	People	Moderate	Tolerable	Assign tasks due to each member's knowledge and provide suitable tutorials for learning
A new strong update which causes a revolution in the used technology in the project	Technology	Low	This will cause repetition of the work to be compatible with the new update	The team has to be aware of the updates and follow the news of technology to be ready for the updates

Table 6.2 Risk Management

6.4 System requirements

6.4.1 Minimum Requirements:

- **Operating System**: Android 6.0 (Marshmallow),iphone6 (ios9).
- **Processor**: Quad-core processor or higher.
- **RAM**: At least 1 GB of RAM.
- **Storage**: A minimum of 100 MB of free storage space.
- **Display**: 720p resolution or higher.
- **Network**: Active internet connection (Wi-Fi or mobile data) for features like Firebase authentication, Firestore database access, and other cloud-based functionalities.

6.4.2 Recommended Requirements

- **Operating System**: Android 9.0 (Pie), iPhone x (ios12).
- **Processor**: Octa-core processor with 64-bit architecture.
- **RAM**: 3 GB or more for smooth performance.
- **Storage**: 200 MB or more free space for additional app data and user-generated content.
- **Display**: Full HD (1080p) resolution for optimal UI experience.
- **Network**: Stable high-speed internet connection for real-time updates and cloud operations.

6.4.3 Coding standard

6.4.3.1 Introduction

This section states the coding standards and best practices that must be followed by all team members to ensure our codebase is consistent and to ensure easy maintenance and scalability.

General Guidelines

- **Consistency:** Write code in a consistent manner, following the conventions specified in this document.
- **Readability:** Write clean and easy-to-read code that anyone can understand.
- **Commenting:** Write comments to explain complex logic, but avoid obvious comments. Comments should clarify why something is done, not what is done.

(**bad comment example:** function getSensorData() // this function returns sensor data

6.4.3.2 Naming Conventions

Variables and Constants

- CamelCase for variables (e.g., profilePicture, soilMoisture).
- UPPER_SNAKE_CASE for constants (e.g., VARIABLE_NAME).
- Avoid one-letter variable names, unless used in loops (e.g., i, j).
- Booleans must answer a question (e.g., isDarkMode, isPasswordHidden)
- Private variables must start with an underscore "_"
- Variable names must be easy to differentiate (sensorData and sensorsData are just one letter different)

Functions and Methods

- camelCase for function names (e.g., getSensorData(), updatePassword()).
- Function names should be descriptive of what they do to avoid using comments.
- Function name should contain a verb (update, get, delete)

Classes

- PascalCase for class names (e.g., UserProfile, FirebaseAuthService).

File

- Use lowercase letters with underscore to separate words (e.g., use_profile.dart, settings_page.dart).

6.4.3.3 Code Formatting

- Indentation: Use one tab for indentation for each nested level
- Line Length: Limit lines to 60-70 characters to improve readability.
- Braces: Always use braces {} for blocks, including if conditions with one statement
- File vertical length should be 200-250 line

6.4.3.4 Error Handling

- Use try-catch blocks to handle errors.
- Avoid empty error handling. Always log or display meaningful messages.

6.4.3.5 Version Control Practices

Git Commit Messages

- Write clear and descriptive commit messages
- State all major changes in the commit message
- **Branching Strategy**
- Use descriptive names for branches (e.g., health-page, login-with-google-fix).
- Always create a pull request for code review before merging into the main branch.

6.4.3.6 Code Review Process

All code must be reviewed by at least one team member before being merged into the main branch.

6.4.4 Use Case

Figure 6.2 Use case diagram

6.4.5 Class Diagram

Figure 6.3: Class Diagram (scan the QR to download the file)

6.4.6 Data Base Schema

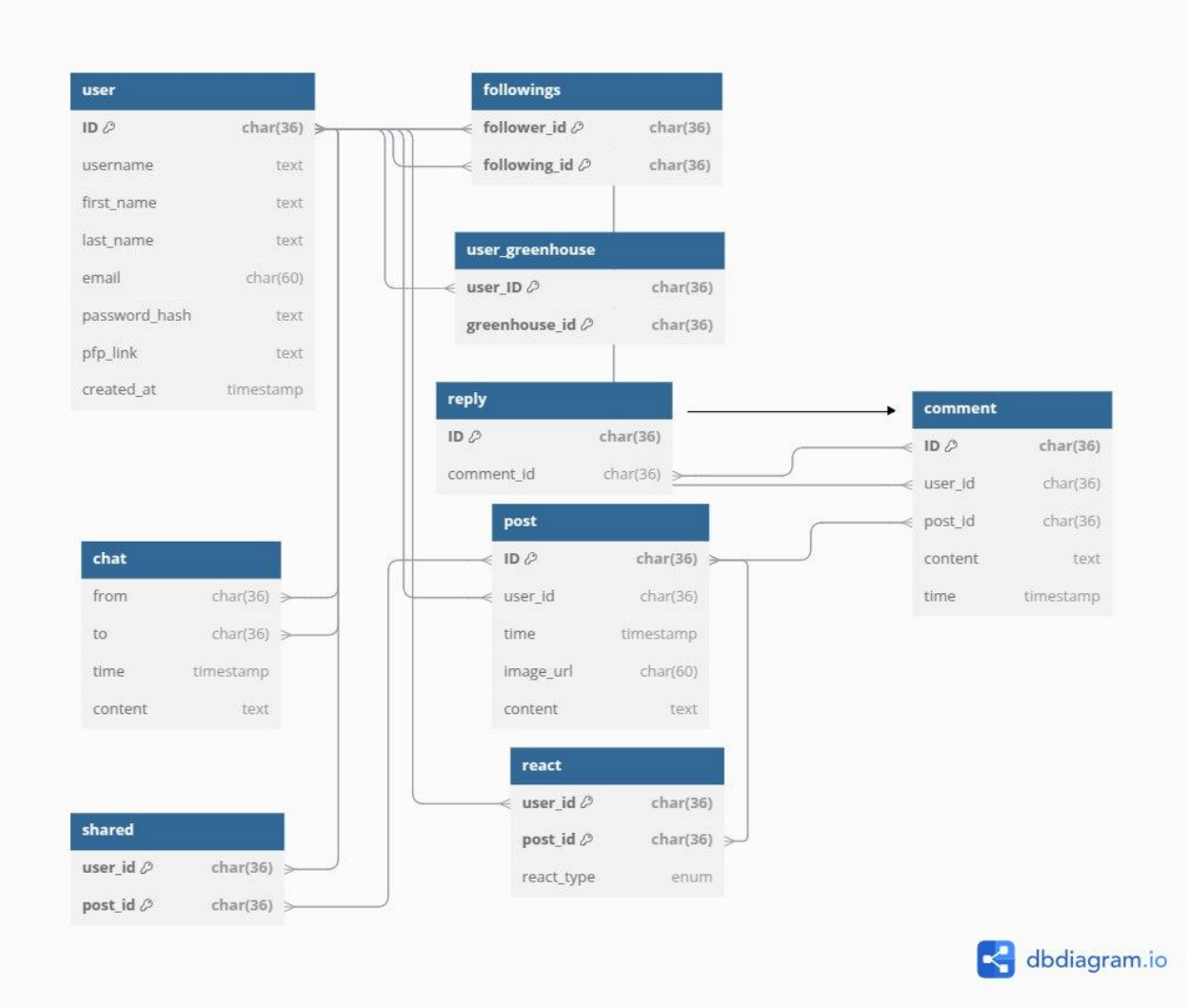


Figure 6.4 DB design

Chapter 7: Team Management

7.1 First Semester Work

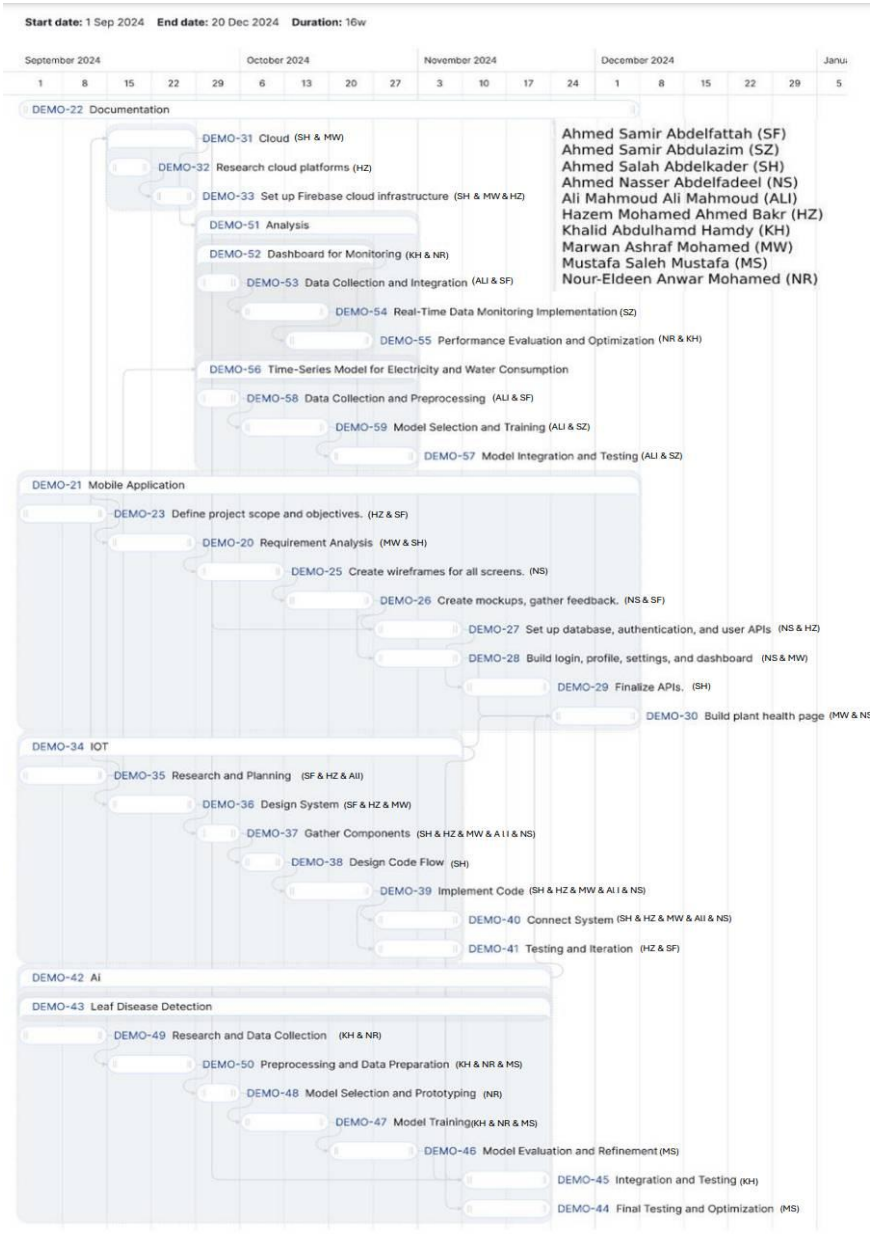


Figure 7.1: Gantt chart 1

7.2 Second Semester Work

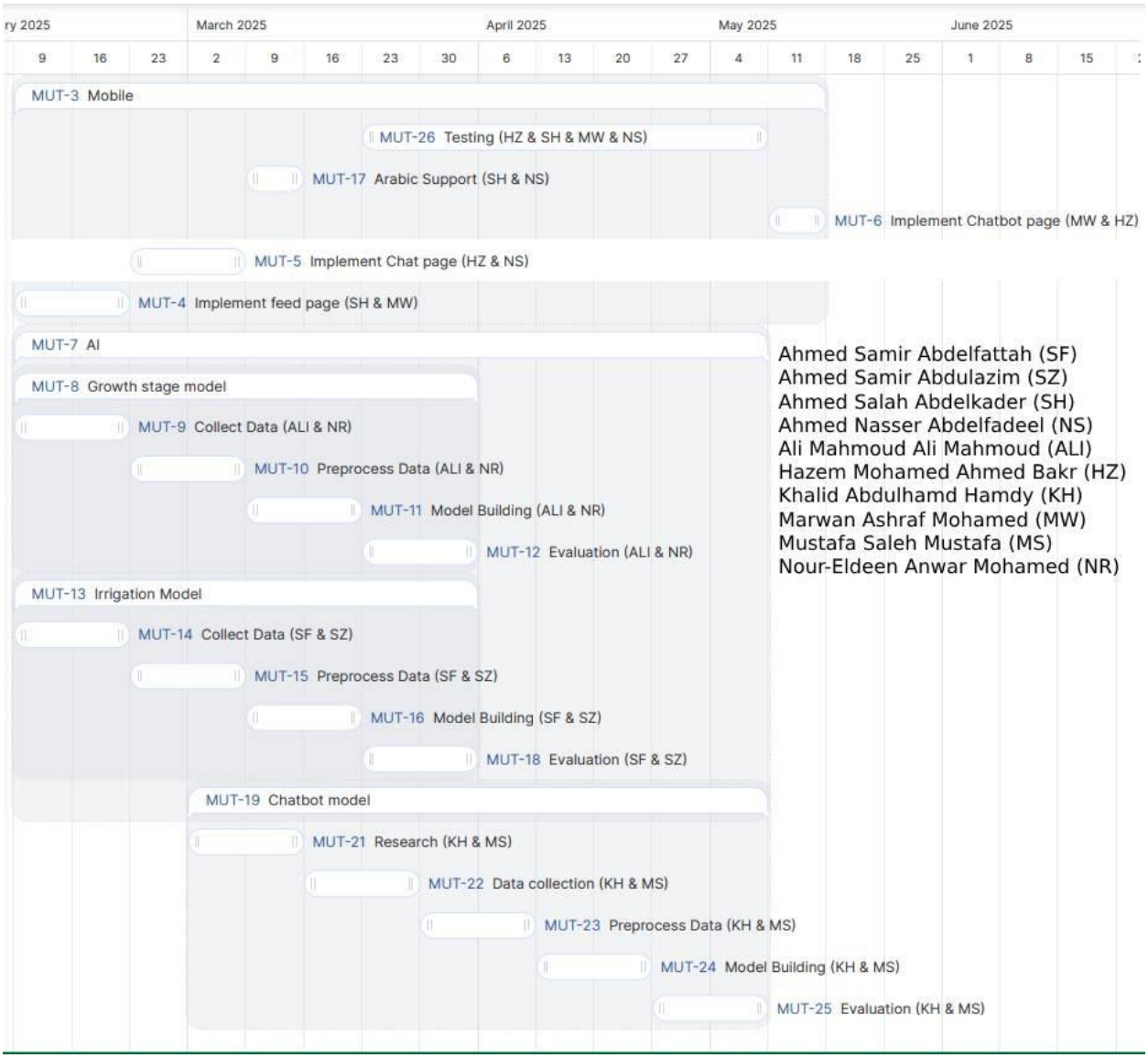


Figure 7.2: Gantt chart 2

7.3 Cloud Technologies

NAME	BRIEF DESCRIPTION
AZURE	Microsoft's cloud computing platform offering services like VMs, databases, and AI.
FLASK	A lightweight Python web framework ideal for building small to medium web applications.
NGINX	A high-performance web server, reverse proxy, and load balancer.
GUNICORN	A Python WSGI HTTP server for running Flask/Django apps in production.
MOSQUITTO	An open-source MQTT broker used for lightweight messaging, often in IoT systems.
MONGODB	A NoSQL document-based database, useful for flexible and scalable data storage.
MYSQL	A popular open-source relational database management system (RDBMS).

REFERENCES

[1] **IoT Based Smart Home: A Machine Learning Approach ,"Publisher:** IEEE, 2021 24th International Conference on Computer and Information Technology (ICCIT)**", Published:**18-20 December 2021.

[2] **A Smart Microcontroller Architecture for the Internet of Things, "by Zhenyu Wu ,Kai Qiu and Jianguo Zhang** School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing 210003, China,Ordnance N.C.O. Academy, The Army Engineering University of PLA, Wuhan 430075, China**",Published: 25 March 2020.**

[3] **Comparative Analysis and Practical Implementation of the ESP32 Microcontroller Module for the Internet of Things, "**A**lexander Maier, Andrew Sharp, Yuriy Vagapov** School of Applied Science, Computing and Engineering, Glyndwr University, Plas Coch, Mold Road, Wrexham, LL11 2AW, UK**",Published: September 2017.**

[4] ESP32 Based Data Logger**, "Ibrahim Al Abbas**, International Journal of Computer1 Science and Mobile Computing**",Published:** May 2019.

[5] **Temperature and Humidity Monitoring Using DHT22 Sensor and Cayenne API,**

"Whisnumurti Adhiwibowo , April Firman Daru , Alauddin Maulana Hirzan Universitas Semarang Jln. Soekarno Hatta, Tlogosari, Semarang**",Published:** 31 January 2020.

[6] **IoT BASED SOIL MOISTURE MANAGEMENT USING CAPACITIVE SENSOR AND USER-FRIENDLY SMARTPHONE APPLICATION, "R and D National Institute for Agricultural and Food Industry Machinery - INMA Bucharest,** Publication name: INMATEH-Agricultural Engineering**",Published: Apr 30, 2022.**

[7] **Optimasi Pesawat Atwood Menggunakan Sensor LDR (Light Dependent Resistor), "**Journal of Applied Sciences, Electrical Engineering and Computer Technology, Publisher: Politeknik Negeri Batam**",Published: 31-08-2020.**

[8] Optimal Placement of Camera Wireless Sensors in Greenhouses, **"Asmaa Ali and Hossam S. Hassanein** School of Computing Queen's University Kingston, Ontario, Canada**", Published:2021 at 23:24:30 UTC from IEEE Xplore.**

[9] WHAT IS YOLOV9: AN IN-DEPTH EXPLORATION OF THE INTERNAL FEATURES OF THE NEXT-GENERATION OBJECT DETECTOR, "Department of Sciences and Humanities, National

University of Computer and Emerging Sciences, Lahore 54770, Pakistan;*Correspondence: m.yaseen@nu.edu.pk;**",Published:** September 13, 2024.

[10] **An IoT based Watering System for Tomato Crop on Smart Agriculture , "George Princess & Poovammal Eswaran** SRM Institute of Science and Technology**", Published: may 2023.**

[11] Effects of supplemental lighting using HPS and LED lamps with different light spectra on growth and yield of the cucumber (Cucumis sativus L.) during winter cultivation in greenhouse,"Publisher: Walter de Gruyter GmbH,Publication name: Folia Horticulturae **", Published: 2021 Treder et al., published by Sciendo.**

[12] ADeep-Learning-Based Model for the Detection of Diseased Tomato Leaves,"[AkramAbdullah 1,†, Gehad Abdullah Amran 2,*,† , S. M. Ahanaf Tahmid 1, Amerah Alabrah 3 Ali A. AL-Bakhrani 4 andAbdulazizAli5]**", Published:(May 2024) { MDPI, Basel, Switzerland}.**

[13] GREENHOUSE MANAGEMENT AND BEST PRACTICE IN EGYPT,"**{Water and Water Structures Engineering Department, Faculty of Engineering, Zagazig University},[M. A. A. Abdrabbo1, Abdelazim Negm2, Hassan E. Fath3 and Akbar Javadi4]",published:2019.**