Part IV Research Project

Project Report

Developing AI-powered Image Processing Web Application for Plant Monitoring in Smart Farm

Kim Zuo

Co-worker: Do Jun Kwon

Supervisor: Dr Ho Seok Ahn

11 October 2024

ENGINEERING
DEPARTMENT OF ELECTRICAL,
COMPUTER, AND SOFTWARE ENGINEERING

ME000-2024

PROJECT TITLE GOES HERE

Kim Zuo

ABSTRACT

The growing global food demand, coupled with climate change and resource scarcity, necessitates innovative solutions for efficient farmland management. This paper proposes a smart agriculture system utilising an AI-powered image processing web application to enhance crop management and reduce labour. Our system integrates computer vision AI models to monitor plant growth, including size, ripeness, and disease detection, alongside a web interface displaying plant status. This allows for informed decision-making by collecting, processing, and displaying real-time data on plant growth and environmental conditions. A user survey rated the system's usability and performance highly, with an average satisfaction of 4.7 out of 5, noting the interface as easy to navigate, beneficial, and visually appealing. By combining sensing technology and artificial intelligence, we show the potential of comprehensive crop monitoring solutions that improve agricultural productivity and sustainability. This approach leads to more efficient and data-driven agricultural practices, which can demonstrate excellent efficiency in the agricultural sector.

DECLARATION

Student

I hereby declare that:

1. This report is the result of the final year project work carried out by my project partner (see cover page) and I under the guidance of our supervisor (see cover page) in the 2024 academic year at the Department of Electrical, Computer and Software Engineering, Faculty of Engineering, University of Auckland.
2. This report is not the outcome of work done previously.
3. This report is not the outcome of work done in collaboration, except that with a potential project sponsor (if any) as stated in the text.
4. This report is not the same as any report, thesis, conference article or journal paper, or any other publication or unpublished work in any format.

In the case of a continuing project, please state clearly what has been developed during the project and what was available from previous year(s):

Signature:

Date:

Table of Contents

List of Figures

Acknowledgements

I would like to express my deepest gratitude to those who have contributed to the successful completion of this project.

First and foremost, I would like to thank my supervisor Ho Seok Ahn, JongYoon Lim, Bruce A. MacDonald for their invaluable guidance, insightful suggestions, and constant support throughout the project.

I am also grateful to my project partner, Do Jun Kwon, his collaboration and hard work were crucial in overcoming the challenges we faced.

I would also like to extend my thanks to the past students who worked on this long-term project. Through studying their code, I have gained valuable knowledge about AI and web application development, which has greatly enhanced my learning experience.

This research was funded by the New Zealand Ministry for Business, Innovation and Employment (MBIE) on contract UOAX2116, Artificial Intelligence-based Smart Farming System, and funded by Ministry of Trade, Industry and Energy (MOTIE), Republic of Korea, Development of an Agricultural Robot Platform Capable of Continuously Harvesting more than 3 Fruits per minute and Controlling Multiple Transport Robots in an Outdoor Orchard Environment. Ho Seok Ahn* is the corresponding author.

1. Introduction

Climate change, population growth, and resource scarcity pose significant challenges to the agricultural sector, demanding increased productivity and sustainability [1]. Artificial Intelligence (AI) offers promising solutions to these challenges by enhancing crop management and optimizing agricultural practices [2].

Our research introduces an innovative smart agriculture system that leverages AI to improve crop management and streamline plant monitoring. By collecting, processing, and displaying real-time data on plant growth and environmental conditions, the system aims to support farmers and agricultural experts in making more informed decisions [3]. Key contributions of our research include:

1. An AI-based monitoring system integrating environmental variable detection, multi-model AI processing for tomato size and ripeness measurement, and real-time data collection and visualization.
2. A web application developed using Flask with a React-based interface, enabling efficient data transfer and user-friendly access to environmental variables and analyzed information.
3. Real-world validation through tests conducted in an actual tomato greenhouse, providing insights into system effectiveness and areas for future improvement.
4. Open-source implementation, contributing to the community and facilitating further development in smart agriculture.[1]

To address implementation challenges and verify the system's effectiveness, we collected real-world data in a tomato greenhouse shown in Figure 1. This practical application allowed us to explore the technology's feasibility and identify areas for improvement. Our research contributes to the growing body of knowledge on smart agriculture applications and provides insights into the practical challenges and opportunities of implementing AI-driven crop monitoring solutions [4].

Figure 1 Actual tomato indoor farm environment.

[1] https://github.com/dkwo575/P4P-61-Webpage

User experience is crucial for effective adoption of such systems. Our user survey revealed an average rating of 4.7 out of 5 for the web interface, indicating strong usability across dimensions such as navigation ease, data visualization clarity, and intuitive design. These results highlight the potential of our AI-based approach to enhance agricultural management efficiency while minimizing manual labor.

By combining sensing technologies with artificial intelligence, our system aims to create a comprehensive crop monitoring solution that can potentially lead to more efficient, data-driven agricultural practices. This research aligns with the broader goal of improving agricultural productivity and sustainability through technology, addressing the pressing need for innovative solutions in modern agriculture [5].

2. Related Works

2.1 AI-driven Web Platforms for Smart Farming

AI-driven web platforms are crucial for smart farming, facilitating the integration of data collection, storage, processing, and visualization [6]. These platforms typically comprise a Backend database, AI models, APIs, and user-friendly interfaces, allowing farmers to access and interpret complex agricultural data easily.

Chukkapalli et al. [7] present a detailed analysis of the smart farming ecosystem and elucidate the key benefits of employing ontologies and AI systems within it. They developed two ontologies to support AI applications in a cooperative environment: the member farm ontology and the cooperative agriculture ontology. These ontologies utilize elements like Farm-Based Units (FBU), On-Board Units (OBU), Worker-Based Units (WBU), and Home-Based Units (HBU) to describe the units existing on the farm, as well as Cooperative-Based Units (CBU) to denote units within the cooperative. Their research not only highlights the significance of these technologies in enhancing agricultural management but also presents a novel approach for connecting various products from partners, emphasizing the need for different security levels to strengthen system security.

The iFarm system, developed by Murakami, exemplifies a web-based cultivation and cost management system that enhances agricultural management efficiency through smartphones, cloud servers, and web browsers [8]. This system enables farmers and managers to monitor farm information via a website or smartphone application and upload work plans to the server. All data is stored in the cloud, accessible to other managers, resulting in improved work efficiency for farm staff [9].

Bhat et al. highlighted the significance of big data in smart agriculture, emphasizing its potential when combined with massive data collection, storage, AI, and predictive analysis [10]. They argued that the integration of big data and AI in agriculture could address current challenges and substantially increase agricultural production quality and quantity [11]. Moreover, AI and big data can predict soil quality, diseases, pests, and crop harvesting times [10].

2.2 AI-based Plant Growth Monitoring

Artificial Intelligence, particularly machine learning and computer vision, has become a powerful tool for analyzing image data generated by cameras in smart agricultural environments [12]. AI technologies have been applied to various aspects of crop management, plant disease detection, yield estimation, and decision-making processes [13].

Demilie [14] provides a detailed overview of recent advancements from 2010-2024 in plant disease detection and classification using machine learning (ML) and deep learning (DL) techniques. The research focuses on the accuracy, speed, and potential application of different techniques in practical agricultural production. Demilie emphasizes that Convolutional Neural Networks (CNNs) are the optimal choice for classifying plant diseases due to their flexibility and feature extraction properties. Pangilinan, Legaspi, and Linsangan [15] focus

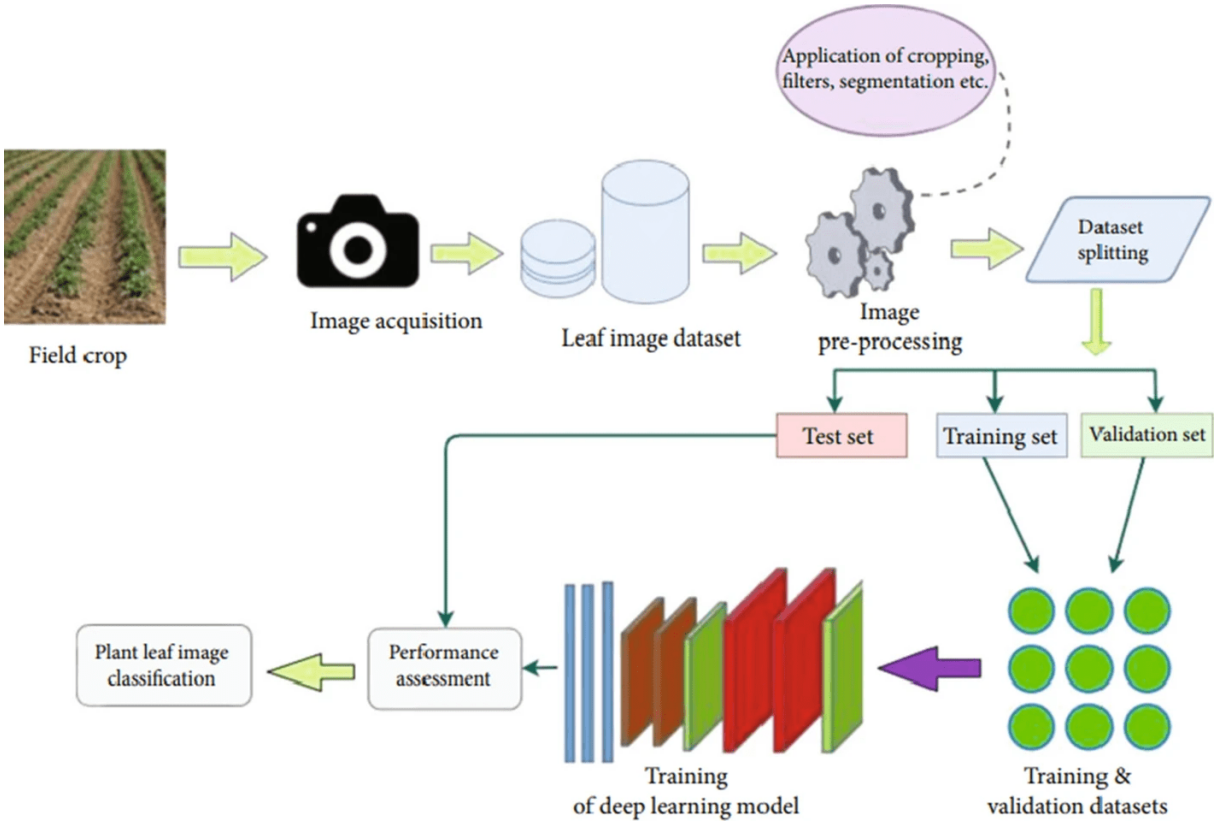


Figure 2 Computer vision-based techniques for plant disease detection and classification by Demilie, W. B.

on the recognition of individual tomatoes, assessing the accuracy of different CNN models to determine the most suitable model for predicting tomato ripeness. They compared the performance of InceptionV3, ResNet50, and VGG19 models in classifying tomato ripeness.

Nourbakhsh et al. [16] simulated scenarios with multiple tomatoes present in an image, identifying the positions and ripeness of each tomato. They used Mask R-CNN and DBSCAN clustering for tomato ripeness evaluation and localization. This approach is particularly useful for scenarios where multiple tomatoes need to be assessed simultaneously.

JinhaJung developed an AI-based system for early plant disease detection using image analysis. Deep learning models trained on a large dataset of leaf images achieved 95

Haiyan and Yong designed an intelligent diagnosis system to help non-skilled farmers detect plant diseases or nutritional deficiencies using Artificial Neural Networks (ANNs). Their ANN-based system analyzed crop symptoms in five categories (overall, root, leaf, fruit, and crop factors) and made diagnoses with an error rate of less than 8

Ramos et al. developed a method to determine and estimate coffee yield by counting coffee cherries using computer vision and AI. Their system categorized coffee cherries as ripe, unharvestable, or intermediate, estimating weight and maturity rate. This approach aimed to provide production data for efficient manpower allocation, machinery preparation, and maximizing economic benefits for coffee growers [17].

Konstantinos emphasized the application of machine learning in agricultural production, noting its use in yield prediction, weed detection, disease detection, and crop quality assessment [18]. These tasks typically involve applying machine learning to sensor data and computer vision, evolving farm management systems into practical AI systems that can maximize productivity and economic benefits while managing crops more efficiently [13].

2.3 Environment Monitoring System

Environmental sensor systems, often utilizing Internet of Things (IoT) technology, are essential for smart farming [19]. IoT sensor systems enable real-time monitoring of environmental variables and data collection [20], consisting of sensor networks that measure various environmental conditions, soil properties, and plant health indicators [21].

Rahman, Kohinoor, and Sami [22] developed a comprehensive smart farming automation system focusing on enhancing productivity in poultry farming. Their system integrates various sensors such as temperature and humidity sensors, rain sensors, and water level sensors. The system includes features like automatic food and water supply mechanisms and automated curtain controls during rainfall. An alert system notifies personnel of extreme environmental issues or shortages in food and water storage, while a data-driven predictive model suggests future trends. Their approach demonstrates how to reduce manual intervention and excel in big data management and monitoring design.

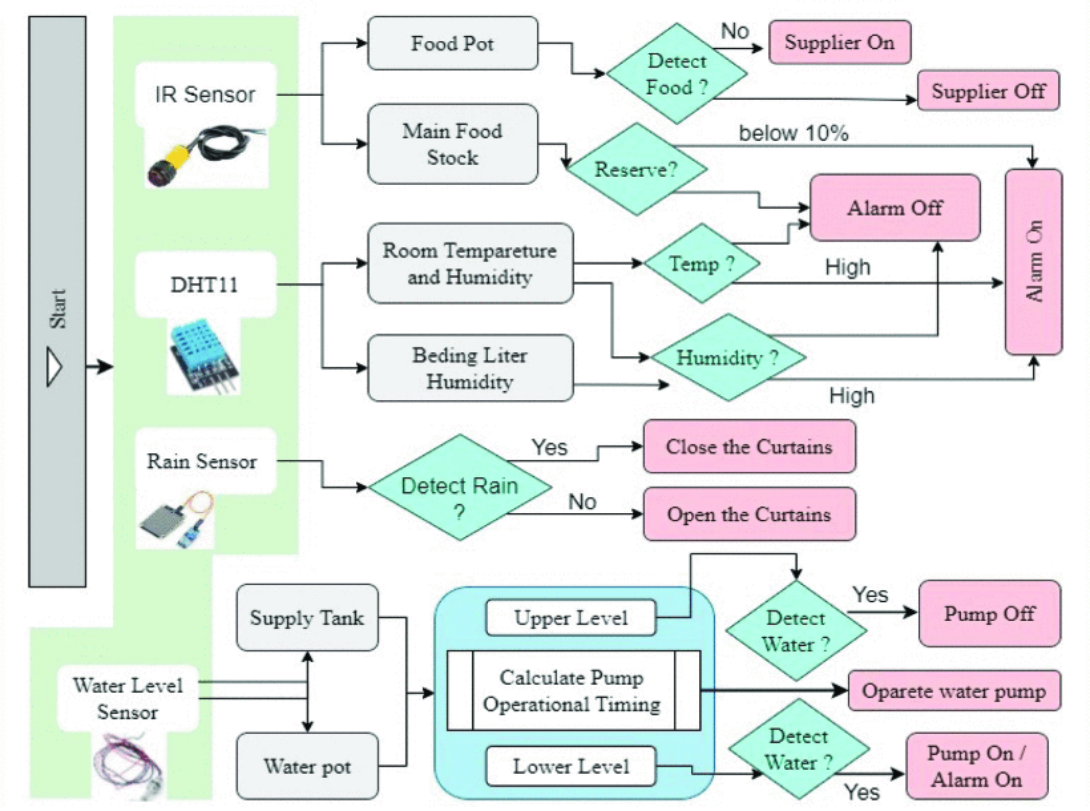
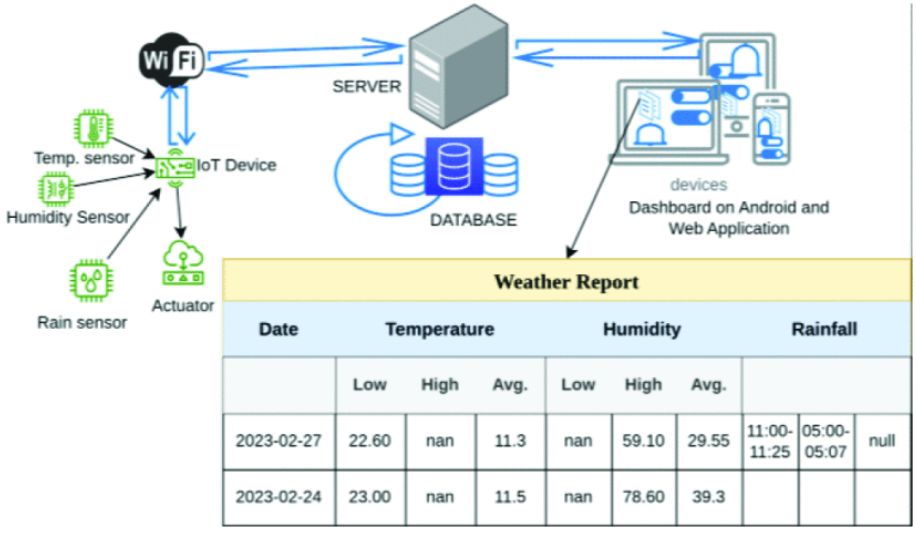


Figure 3 System Overview of IoT-based Automated Farm By Rahman, Kohinoor, and Sami.

Their study highlighted that IoT in agriculture can increase efficiency, reduce costs, and improve the economic level of agriculture. IoT technology can make farms smarter, more efficient, and more productive by reducing manual work and increasing automation [23]. Environmental monitoring, using various sensors to track factors such as temperature, humidity, and soil moisture, is crucial in smart agriculture [24].



Weather Report

Date	Temperature			Humidity			Rainfall		
	Low	High	Avg.	Low	High	Avg.			
2023-02-27	22.60	nan	11.3	nan	59.10	29.55	11:00-11:25	05:00-05:07	null
2023-02-24	23.00	nan	11.5	nan	78.60	39.3			

Figure 4 Web Server of IoT-based Automated Farm By Rahman, Kohinoor, and Sami.

Dolci emphasized that IoT technology can increase agricultural sophistication without significant capital investment. Low-cost sensors for humidity, temperature, light, and soil moisture can help farmers easily monitor environmental variables and achieve high results with low capital [25]. These sensors provide farmers with important data to optimize irrigation, pest control, and overall crop management [20].

Wolfert introduced tomato indoor farm technology utilizing IoT. By developing sensor and actuator-based systems, users can access information related to crop growth, climate, and irrigation settings through a web-based decision-making system [26]. This approach allows for reduced resource use and increased productivity based on crop monitoring using IoT system technology [27].

3. System Architecture

3.1 Initial Ideation and Development

The initial stage of the project involved analyzing content from past year projects, including a web application designed to display SMART Farm environment data and 3D models of tomato plants built using Vue.js and React. Additionally, AI models capable of analyzing tomato size and ripeness based on images and corresponding depth maps were examined.

Upon individual analysis of each component, it was noted that while the web application was well-designed, a significant issue was identified. The fake data provided by the Frontend server to be displayed on the web application was sourced from React .ts files imported by each page. This indicated that the original design did not consider data transport between Frontend and Backend, necessitating the development of a method to facilitate this communication. On a positive note, the AI models were well-designed, though during implementation, modifications were made to the output of the tomato ripeness analysis model to color-code detected tomatoes based on their ripeness level.

During this initial phase, the Keyestudio ESP32 Smart Farm Kit[2] was assembled. This device integrates various sensors to achieve automation, wireless control, and intelligent management on a simulated farm constructed from Basswood Board. While one team

[2]https://docs.keyestudio.com/projects/KS0567/en/latest/wiki/index.html

member focused on enabling IoT sensors to record data to a local MySQL database, another investigated methods to connect the React Frontend with a Flask Python server.

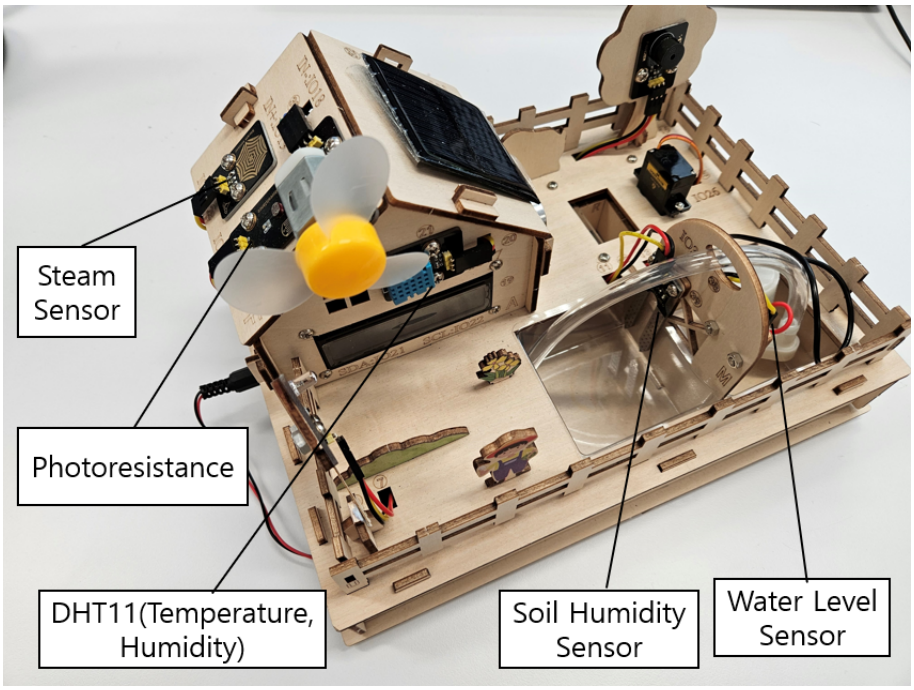


Figure 5 Keyestudio ESP32 Smart Farm Kit

Prior to commencing coding, several weeks were dedicated to discussing the overall system architecture. The final version of the System Architecture, as illustrated in Figure6 consists of three key components:

1. Input Devices (left):
    - IoT Sensor for environmental data collection
    - Laptop with RealSense camera for stationary image capture
    - Mobile Robot with 3 Intel RealSense cameras for dynamic image collection
2. AI Server (center):
    - Backend Server, which includes:
        - MySQL database for IoT Sensor data storage
        - Original Image Dataset for storing Tomato and Leaf images
        - AI Image Output for processed images
    - AI Model, performing:
        - Target Detection, Size and Volume Measurement
        - Leaf Detection and Disease Identification
        - Tomato Ripeness Analysis
3. User Interface (right):
    - SMART Farm Monitor Webpage, accessible by multiple users simultaneously

Areas outlined in red indicate my partner's focus, while those in pink represent my areas of concentration. Orange outlines denote sections we both developed, comparing our respective approaches for optimal results.

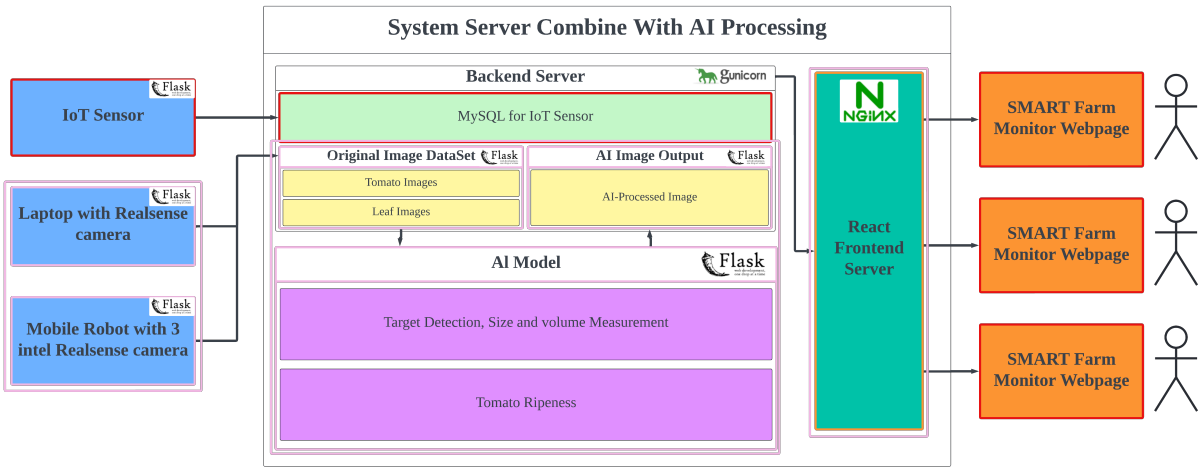


Figure 6 Overall System Architecture

3.2 Frontend Server Development

At the project's inception, consideration was given to utilizing Axios[3] in conjunction with a Flask Backend to modify the .ts file containing the data structure for importing into each page and other files. Figure7 illustrates the data.ts file import in the original design. As discussed in the Initial Ideation and Development section, this Frontend system did not account for future development, as the data displayed on the web application was directly imported from a TypeScript file rather than utilizing an API for data transfer management. Although the method initially developed was not ultimately used, it is worth discussing the

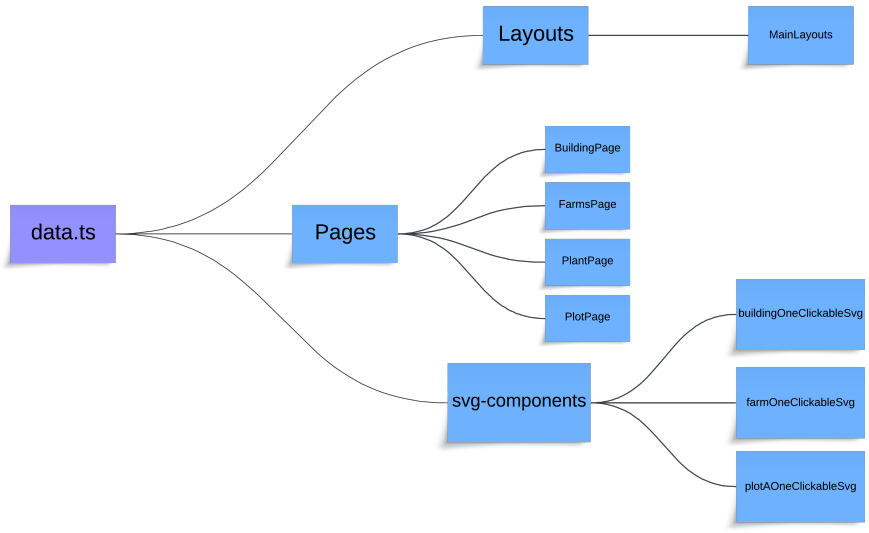


Figure 7 Original Design Fake Data Import

logic behind it and the reasons for its exclusion from the final implementation. The design aimed to maintain the data.ts structure from the original Frontend design by creating a JSON file editor using Python. Through Axios, HTTP packages would be sent between the Frontend web application server and the Backend Flask server, synchronizing the Backend JSON file information with the data.ts. This approach ensured a copy of the Frontend information was maintained in the Backend for emergency situations requiring Frontend data erasure and reset, or for creating logs based on the data. Figure8 illustrates the design

[3]https://www.geeksforgeeks.org/axios-in-react-a-guide-for-beginners/

for upgrading the Frontend server.

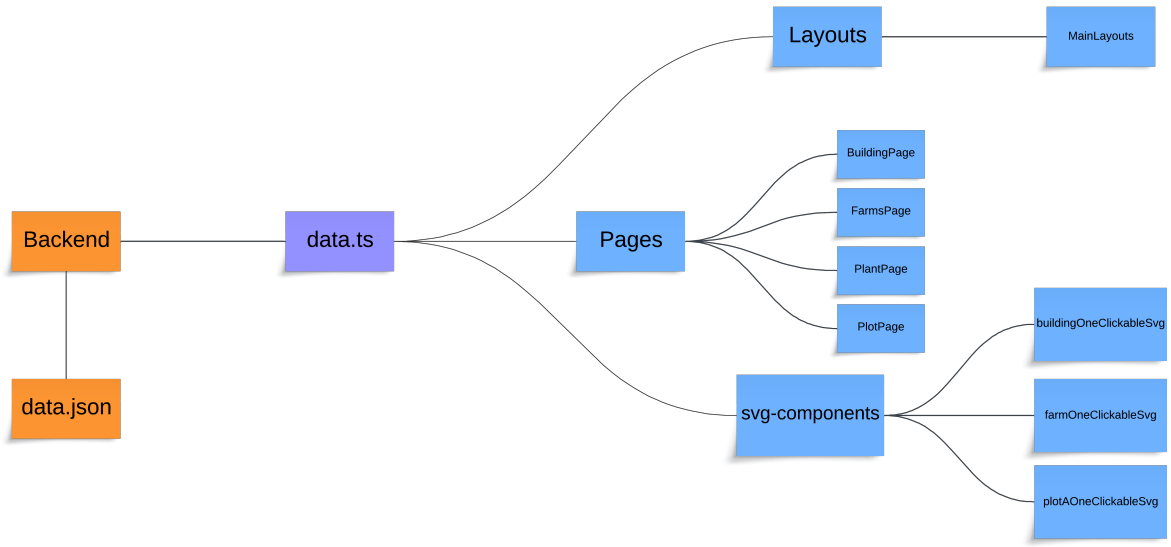


Figure 8 Upgrade Design Fake Data Import

The Figure9 provides a detailed representation of the Frontend data structure from the original design. While this structure could serve as a basis for creating a JSON editor, upon closer examination, it became apparent that this approach was overly complex for the project's needs. Consequently, this was not selected as the final solution.

					Variable Name	Variable Type	
data					**data**	**List [dict {}, {}]**	**use to store all the information for this database**
	id				id	int	use to store farm id
	farmName				farmName	str	use to store farm name
	buildings				**buildings**	**List [dict {}, {}]**	**store all infomation relate to a building in farm**
		id			id	int	use to store buidling id
		buildingName			buildingName	str	use to store buildings name
		events			**events**	**List [dict {}, {}]**	**use to store event happend in building**
			date		date	str	date of this event happend
			text		text	str	what event happend
		environment			**environment**	**List [dict {}, {}]**	**use to store the envrionement variable in different date**
			date		date	str	date of these environment data record
			temperature		temperature	int	temperature for that date
			humidity		Humidity	int	fluorescents for that date
			light		Light	int	Light for that date
			WaterLevel		Water Level	int	Water level for that date
			SoilHunidity		Soil Humidity	int	Soil humidity for that date
			Steam		Steam	int	steam data for that date
		data			**data**	**List [dict {}, {}]**	**use to store the data for all plants in the buidling**
			date		date	str	date of these data record
			area		area	int	area of plants take in total
			fruitlets		fruitlets	int	fruitlets all plants in total
			height		height	int	height for the highest plant
			leaves		leaves	int	leaves in total
			volume		volume	int	volument in total
			width		width	int	width in total
		plots			**plots**	**List [dict {}, {}]**	**use to store the data for each plant specific**
			id		id	int	plant id
			plotName		plotName	str	plant name
			data		**data**	**List [dict {}, {}]**	**use to store the data for each plant in different date**
				date	date	str	date of these data record
				area	area	int	area of plants take
				fruitlets	fruitlets	int	fruitlets this plant grow
				height	height	int	height of this plant
				leaves	leaves	int	leaves this plant have
				volume	volume	int	volument take for this plant
				width	width	int	wid take for this plant

Figure 9 Frontend Data structure

An alternative approach was developed by the team member responsible for IoT sensors. This method involved recording data captured by IoT sensors into a MySQL database. By creating local constants in the React web application, data could be efficiently transferred from the MySQL database to the Frontend web application, streamlining the process and improving overall system architecture.

3.3 Monitoring System and Backend Server Development

The monitoring system utilizes RealSense D435 cameras to capture 1280 x 720p color images along with 640 x 480p depth images per shot. This necessitates a system capable of transferring both image data and depth map files, as well as recognizing the types of files being transferred. After data transfer to the server, the system must efficiently route this information to the AI models for analysis. Following team discussions, it was determined that the final output would involve transferring AI-processed images to the web application for comparison with raw image data.

The decision was made to develop the Backend server using the Flask[4] package from Python. To explore the implementation of AI model integration and data transfer between server and client, initial development focused on creating two scripts: a server script (app.py) and a client script (client.py). This approach facilitated learning about Flask's data transfer capabilities. The initial concept involved transferring raw image data from the client to the server, where image processing would occur, with the processed images subsequently displayed on the web application, this simple workflow is illustrates by Figure13 .

Figure 10 Backend Initial Idea

However, this transmission method was found to limit the system's long-term scalability. A single-structure approach would not adequately meet potential future needs for additional functionalities. To address this limitation, a modular approach was adopted. Using a script similar to client.py, the AI-Powered Smart Farm Backend System, originally intended as a monolithic implementation, was split into independent modules. This modular design allows different functions to be executed by connecting various modules to the server, enhancing the system's flexibility and adaptability to future requirements.

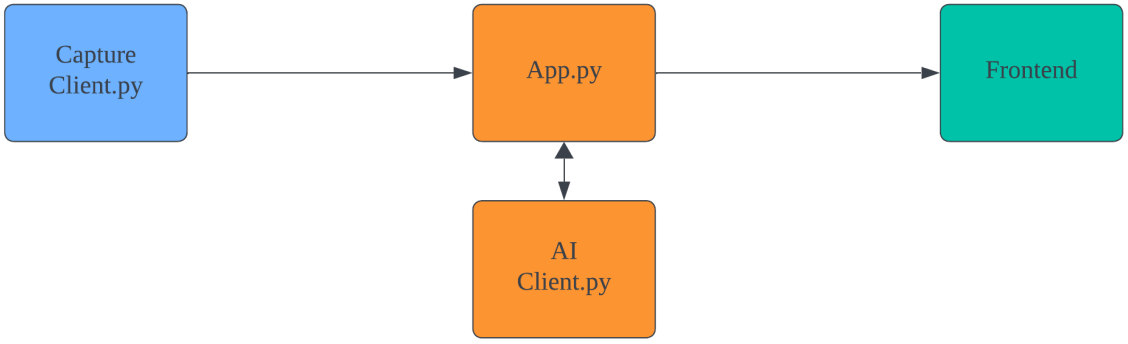


Figure 11 Backend Implement AI processing

[4]https://flask.palletsprojects.com/en/3.0.x/

To facilitate proper file management and storage, the server was designed to distinguish between different client sources. This was accomplished by assigning unique IDs to each client script, allowing for identification of the client type. In the implemented system, files transferred from clients with the ID `'realsense_client'` are stored in the 'Original' folder, while those from `'ai_client'` are directed to the 'Result' folder. The client ID information is transmitted to the server along with the file name and file type (e.g., PNG or NPY).

The server's functionality extends beyond unilateral reception of information from clients. Given its central role in connecting various clients, the server is designed to notify all connected clients when a new file is uploaded. This functionality is implemented using WebSocket technology. Upon receiving a new file, the server generates a URL for file access and download based on the file information. This URL, along with the client ID, is then broadcast to all connected clients.

To prevent redundant downloads and uploads, clients are programmed to ignore notifications about files uploaded by clients with matching IDs. This design choice is based on the assumption that clients with the same ID serve similar functions. For example, in the System Architecture6 both the laptop with RealSense camera and the mobile robot with three Intel RealSense cameras upload pictures to the server using the `'realsense_client'` ID. This mechanism prevents clients from automatically downloading and re-uploading their own files, and avoids the unnecessary download of files from functionally similar clients. Figure13 illustrated the backend structure adding specific ID to each client. As previously

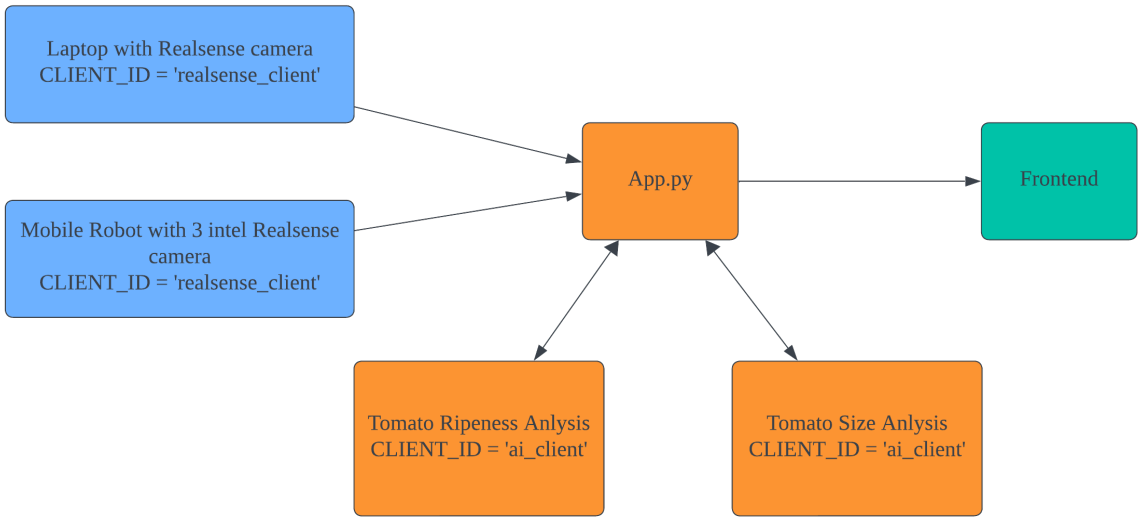


Figure 12 Backend Implement CLIENT_ID

mentioned, to prevent clients from automatically downloading and re-uploading their own files, and to avoid unnecessary downloads from functionally similar clients, other clients do not automatically download files when the server sends a notice through WebSocket if the uploading client's ID matches their own.

However, from a long-term perspective, this design lacks sustainable development potential. Relying solely on ID to distinguish client attributes would prevent communication through the server between clients with identical IDs. To address this limitation, an update was implemented on August 15th, introducing an additional attribute for client identification named CLIENT_NAME.

This new structure utilizes CLIENT_ID to categorize the client type, while CLIENT_NAME provides more detailed information about the specific client. This dual-attribute approach allows for a more nuanced and flexible client management system. The CLIENT_ID serves to broadly classify clients, while the CLIENT_NAME enables finer-grained identification and differentiation.

This enhanced identification system improves the server's ability to manage all clients and their data transmissions more effectively. It provides a more robust foundation for future scalability and inter-client communication, even among clients of the same type or category.

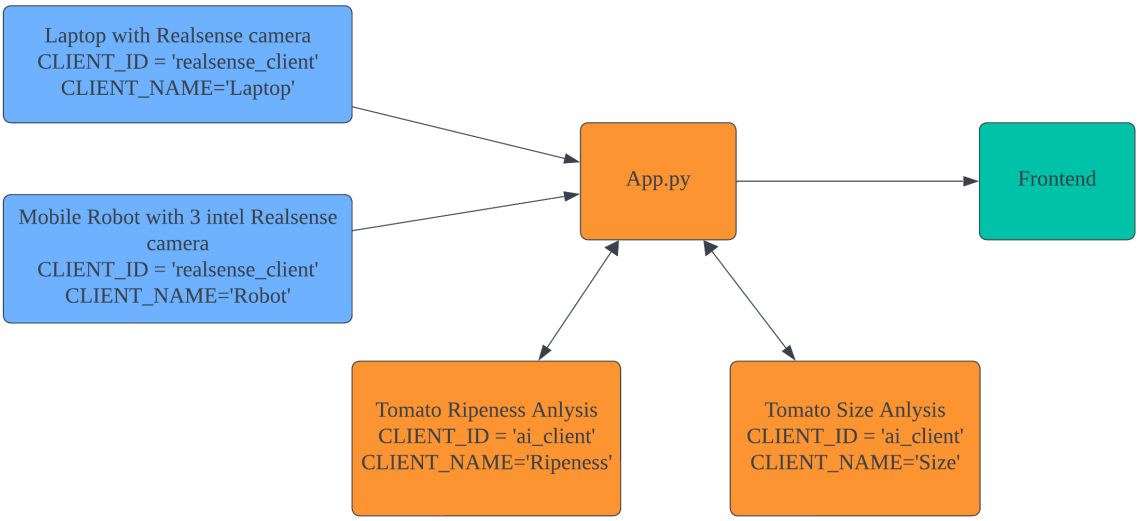


Figure 13 Backend Implement CLIENT_ID

3.4 Monitoring System Workflow

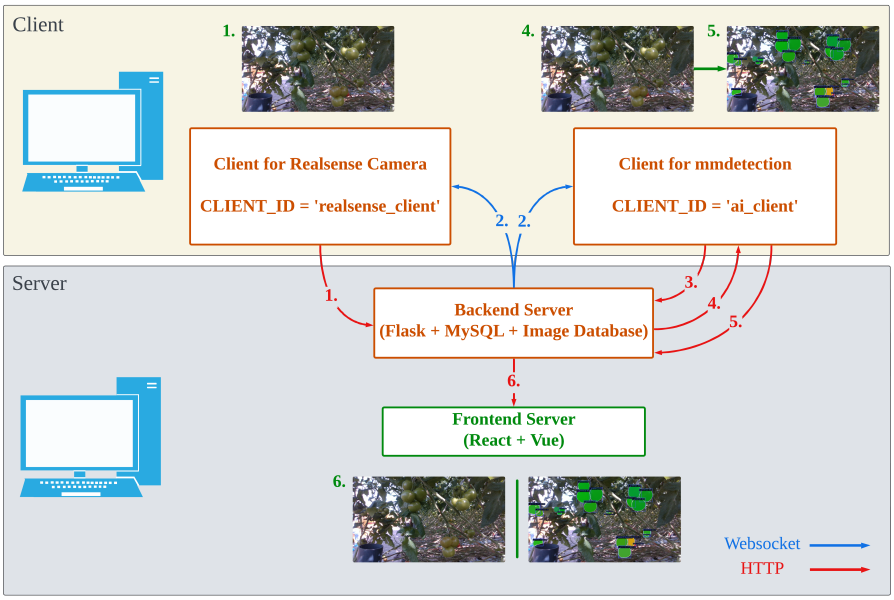


Figure 14 Plant growing monitoring Workflow.

Figure 14 illustrates the workflow of our plant growing monitoring method delineating into two primary sections:

1. Combined Backend (highlighted in amber): These components include image collection, image processing and data management functions.
2. Frontend System (highlighted in green): This section represents the user interface and client-side operations.

These components interact through two distinct communication channels:

1. HTTP Protocol (represented by red arrows): Use API calls, enabling request-response interactions for data retrieval and submission.
2. WebSocket Message Transfer (represented by blue arrows): Facilitates real-time, bidirectional communication, allowing for instant updates and live data streaming between the server and client.

Content Bellow is our System Workflow in detail:

1. **Image Capture and Upload:**
   - The Realsense Camera Client captures raw images.
   - The client uploads raw PNG images and NPY data files to the Backend Server (using HTTP protocol).
2. **Server Notification:**
   - Upon receiving the uploaded data, the Backend Server broadcasts a notification to all connected clients via WebSocket.
   - The notification includes information about the upload event from realsense_client.
3. **AI Client Data Request:**
   - The AI Processing Client (mmdetection) receives the WebSocket notification and sends a download request to the Backend Server.
4. **Data Transfer to AI Client:**
   - The Backend Server sends the PNG image and NPY data to the AI Client (using HTTP protocol).
5. **AI Processing and Result Upload:**
   - The AI Client processes the received PNG and NPY data using the mmdetection model.
   - After processing, it generates a new PNG image with detection results.
   - The AI Client uploads the processed image back to the Backend Server (using HTTP protocol).
6. **Web Application Display:**
   - The Backend Server sends the processed image to the Frontend Server.
   - The Frontend Server displays both the original and processed images on the Web application page for user viewing.

Detailed Component Description:

The client script, responsible for controlling the RealSense camera, manages both data capture and transmission. Upon image acquisition, the data is automatically transmitted to the server via HTTP protocol. The server, upon receipt of a new file, logs pertinent metadata including file format, uploading client ID (which denotes file type), and the file's download URL. This information is then disseminated to all connected clients through WebSocket broadcast.

As previously outlined, clients receiving notifications about files with IDs differing from their own initiate automatic downloads. Clients equipped with image processing capabilities, such as those illustrated on the right side of Figure 14, employ AI models to analyze the downloaded files and subsequently return processed images to the server.

Each client Python script is uniquely identified by two key attributes: Client_id and Client_name. These identifiers are transmitted to the server via HTTP protocol during file uploads, serving as reference points for file storage and management. The Client_name attribute is omitted from Figure 14 for clarity, as the distinct Client_ids suffice for macroscopic system differentiation. This dual-identifier design enables the server to communicate with specific clients via WebSocket, facilitating targeted message delivery and triggering customized functions.

The workflow depicted in Figure 14 demonstrates the system's capacity for efficient and targeted file distribution across multiple clients. This architecture significantly enhances the functionality and scalability of the smart agricultural monitoring system. It establishes a robust foundation for future expansions, such as the integration of additional AI models or adaptation to diverse agricultural monitoring projects. Furthermore, the highly customize nature of the Client and Server scripts allows for the transmission of various content types, enabling Clients and Servers to perform a wide array of purposeful functions based on their ID and Name attributes.

This sophisticated client-server architecture, coupled with the modular design approach discussed earlier, positions the system to effectively meet both current requirements and potential future needs in smart agricultural monitoring.

4. Result

4.1 System Workflow Verify

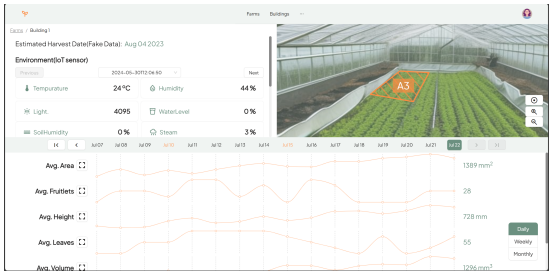
To simulate file transmission in a real-world environment and verify ideas in the Backend Development stage, we implemented the distributed setup as shown in the Figure 14 by separating different scripts to run on distinct devices, effectively emulating a realistic client-server architecture.

One device was designated to simulate the server, while another device emulated various clients interacting with the server through upload and download operations, as well as other corresponding actions. This configuration allowed us to test the system's functionality in a distributed environment, closely mimicking real-world conditions.

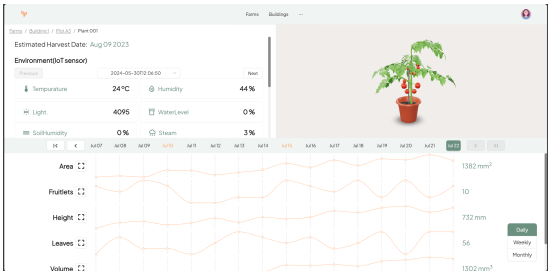
After ensuring that the entire system functioned exactly as intended in this distributed setup, we proceeded to integrate the whole system on a Linux machine using Gunicorn and Nginx. This additional step was taken to further test our server's performance and reliability in a Linux environment, which is commonly used for production deployments.

By implementing this two-phase testing approach we were able to thoroughly validate our system's capabilities under various operational scenarios. This comprehensive testing strategy helps ensure that our smart agricultural monitoring system can perform robustly in real-world applications, whether in a distributed network or on a centralised Linux server.

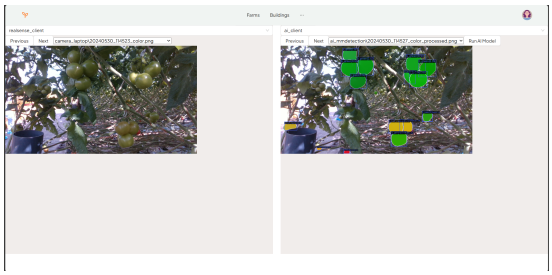
4.2 Integrate Backend With Frontend Web application



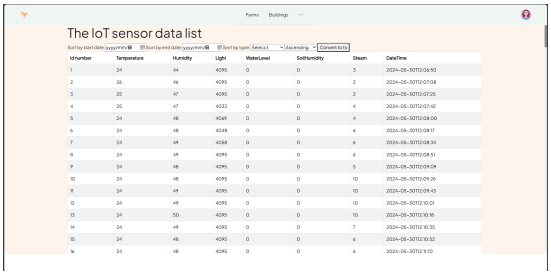
(a) Region Select page. Users can select buildings and areas to inspect.



(b) Plant Page. Users can view the environmental data of the plant.



(c) Image Page. Users can view the plant ripeness results.



(d) MySQL Database displayed in the Web Interface.

Figure 15 Overview of the Frontend Pages

Figure 15 provides a detailed explanation of the key web pages in our application, with the Image Page being particularly relevant to my focus area.

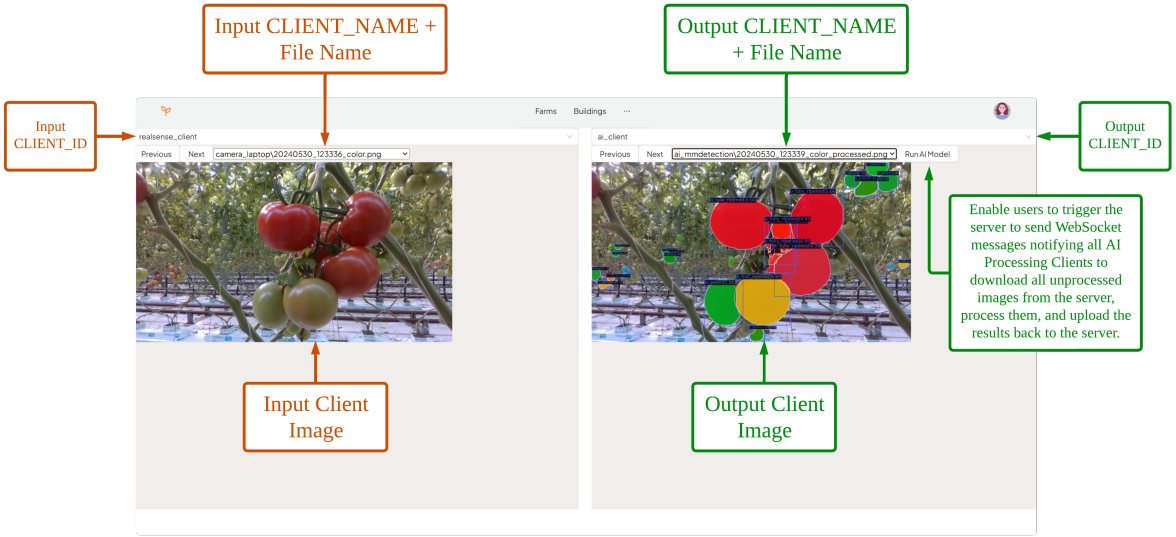
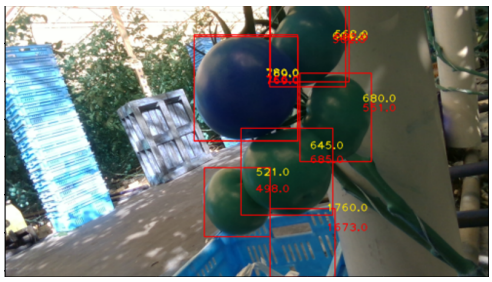


Figure 16 Frontend Image Page with Explained.

Figure 16 illustrates and analyzes the design of this page in detail. In this image, the left section is defined as the input, displaying images stored in the 'Original' folder of the

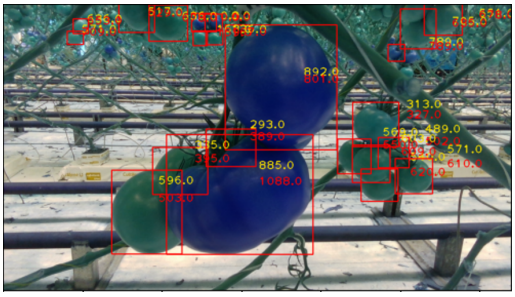
server database, which contains all images uploaded from sources such as the RealSense Camera. The right section is defined as the output, showing images stored in the 'Result' folder of the server database, where all images uploaded by AI Processing Clients are kept.

(a) Original Image for Size Measurement



(b) Size Measurement Result

(c) Another Example of Original Image for Size Measurement

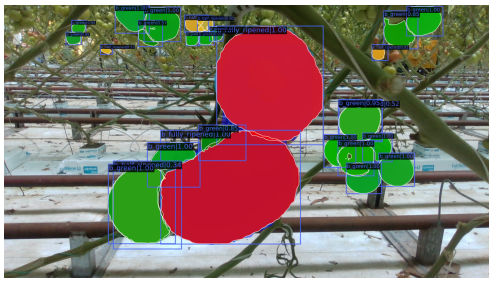


(d) Another Example of Size Measurement Result

(e) Original Image for Ripeness Measurement

(f) Ripeness Measurement Result

(g) Another example of Original Image for Ripeness Measurement



(h) Another example of Ripeness Measurement Result

Figure 17 Size and Ripeness Measurement Results

Figure17 show example output from the AI Processing Client compare to the original images, these model enables automated size and ripeness tracking of tomato across multiple plants simultaneously, offering farmers a comprehensive view of crop maturity without the need for manual inspection.

The user experience is a critical component of such systems, as effective adoption hinges

on intuitive and efficient interfaces. To assess the usability and performance of the web interface, a user survey was conducted, yielding overwhelmingly positive feedback. The interface received an average rating of 4.7 out of 5, indicating exceptional user experience across multiple dimensions, including navigation ease, data visualization clarity, and design intuitiveness. These results underscore the potential of the AI-based approach to enhance agricultural management efficiency while minimizing manual labor requirements.

5. Discussion

The findings of this project demonstrate the significant potential for integrating AI-driven systems into smart agriculture to enhance efficiency, productivity, and sustainability. The development of a real-time monitoring system for plant growth utilizing computer vision models shows considerable promise. The system effectively measures tomato size and ripeness, providing farmers with actionable insights without manual intervention. This work aligns with existing research showcasing how AI and IoT technologies can enhance agricultural management through task automation and real-time data provision for optimized decision-making processes.

Benchmarking against related systems highlights the project's strengths, such as real-time image processing and a modular architecture allowing for future scalability. The modular design facilitates easy integration of additional AI models and potential application across various agricultural contexts. When compared to other AI-based plant monitoring platforms, this system demonstrates advantages in its utilization of open-source tools and practical validation through real-world tests in a greenhouse setting.

However, there are limitations exist, because the research is constrained by the relatively small scale of its application, focusing solely on tomato plants in a controlled environment. Scaling the system for use with other crops or larger agricultural settings may introduce unforeseen challenges, particularly in handling larger datasets and varying environmental conditions. While the AI models employed in this project proved effective, there remains potential to further refine and train these models with more extensive datasets to enhance accuracy and robustness.

Regarding the research questions and objectives, the project successfully demonstrates the applicability of AI-powered image processing to smart farming. The objectives of developing a user-friendly web application and integrating it with real-time data from IoT sensors have been achieved. The system addresses the need for more efficient crop management solutions, offering a viable framework extensible to broader agricultural applications.

6. Conclusion

Our project affirms the effectiveness of generating automated plant monitoring methods in smart agriculture using applications including AI and IoT sensors to enhance decision-making. The primary contribution of my research lies in the development of a modular, scalable architecture that integrates AI-driven image analysis with real-time environmental monitoring. This system enables more precise management of crop growth, allowing farmers to optimise yield while reducing labour costs. This modular system also ensured that developers can easily integrate other AI models by creating their own client.py Python file connected to their AI model processing system.

Future research directions could focus on expanding the system's scope to encompass a

wider range of crops and environmental factors, such as implementing tomato leaf disease detection or ripeness analysis for diverse crop varieties. Enhancement of the Backend System through the implementation of robust encryption and decryption protocols for API communications between servers and clients would significantly bolster overall system security. Furthermore, exploration of methods to augment the system's real-time processing capabilities is warranted, particularly to accommodate larger-scale farming operations. These advancements would substantially contribute to the system's scalability and security, positioning it for broader adoption across diverse agricultural settings. The integration of such improvements would not only enhance the system's versatility but also reinforce its potential as a comprehensive solution for modern precision agriculture.

References

[1] Food and Agriculture Organization, *The future of food and agriculture*. Rome, Italy: Food & Agriculture Organization of the United Nations (FAO), Jun. 2017.

[2] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, "Machine learning in agriculture: A review," *Sensors*, vol. 18, no. 8, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/8/2674

[3] S. Wolfert, L. Ge, C. Verdouw, and M.-J. Bogaardt, "Big data in smart farming - a review," *Agricultural Systems*, vol. 153, pp. 69–80, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0308521X16303754

[4] M. Bacco, A. Berton, E. Ferro, C. Gennaro, A. Gotta, S. Matteoli, F. Paonessa, M. Ruggeri, G. Virone, and A. Zanella, "Smart farming: Opportunities, challenges and technology enablers," in *2018 IoT Vertical and Topical Summit on Agriculture - Tuscany (IOT Tuscany)*, 2018, pp. 1–6.

[5] A. Walter, R. Finger, R. Huber, and N. Buchmann, "Smart farming is key to developing sustainable agriculture," *Proceedings of the National Academy of Sciences*, vol. 114, no. 24, pp. 6148–6150, 2017. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.1707462114

[6] A. Faid, M. Sadik, and E. Sabir, "An agile ai and iot-augmented smart farming: a cost-effective cognitive weather station," *Agriculture*, vol. 12, no. 1, p. 35, 2021.

[7] S. S. L. Chukkapalli *et al.*, "Ontologies and artificial intelligence systems for the cooperative smart farming ecosystem," *IEEE Access*, vol. 8, pp. 164 045–164 064, 2020.

[8] Y. Murakami, "ifarm: Development of web-based system of cultivation and cost management for agriculture," in *2014 Eighth International Conference on Complex, Intelligent and Software Intensive Systems*, 2014, pp. 624–627.

[9] M. J. O'Grady and G. M. O'Hare, "Modelling the smart farm," *Information processing in agriculture*, vol. 4, no. 3, pp. 179–187, 2017.

[10] S. A. Bhat and N.-F. Huang, "Big data and ai revolution in precision agriculture: Survey and challenges," *IEEE Access*, vol. 9, pp. 110 209–110 222, 2021.

[11] S. A. Osinga, D. Paudel, S. A. Mouzakitis, and I. N. Athanasiadis, "Big data in agriculture: Between opportunity and solution," *Agricultural Systems*, vol. 195, p. 103298, 2022.

[12] N. C. Eli-Chukwu, "Applications of artificial intelligence in agriculture: A review." *Engineering, Technology & Applied Science Research*, vol. 9, no. 4, 2019.

[13] H. Tian, T. Wang, Y. Liu, X. Qiao, and Y. Li, "Computer vision technology in agricultural automation-a review," *Information Processing in Agriculture*, vol. 7, no. 1, pp. 1–19, 2020.

[14] W. B. Demilie, "Plant disease detection and classification techniques: A comparative study of the performances," *Journal of Big Data*, vol. 11, no. 5, 2024.

[15] J. R. Pangilinan, J. Legaspi, and N. Linsangan, "Inceptionv3, resnet50, and vgg19 performance comparison on tomato ripeness classification," in *Proceedings of the 5th International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*. Yogyakarta, Indonesia: IEEE, 2022, pp. 619–624.

[16] A. Nourbakhsh, E. B. Mehrabani, S. Faraji, F. A. Shirazi, and M. Hatefi, "Tomato ripeness evaluation and localization using mask r-cnn and dbscan clustering," in *Proceedings of the 11th RSI International Conference on Robotics and Mechatronics (ICRoM)*. Tehran, Iran: IEEE, 2023, pp. 714–719.

[17] P. Ramos, F. Prieto, E. Montoya, and C. Oliveros, "Automatic fruit count on coffee branches using computer vision," *Computers and Electronics in Agriculture*, vol. 137, pp. 9–22, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016816991630922X

[18] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, "Machine learning in agriculture: A review," *Sensors*, vol. 18, no. 8, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/8/2674

[19] R. Dagar, S. Som, and S. K. Khatri, "Smart farming–iot in agriculture," in *2018 International Conference on Inventive Research in Computing Applications (ICIRCA)*. IEEE, 2018, pp. 1052–1056.

[20] G. Sushanth and S. Sujatha, "Iot based smart agriculture system," in *2018 international conference on wireless communications, signal processing and networking (WiSPNET)*. IEEE, 2018, pp. 1–4.

[21] S. Jaiganesh, K. Gunaseelan, and V. Ellappan, "Iot agriculture to improve food and farming technology," in *2017 Conference on Emerging Devices and Smart Systems (ICEDSS)*, 2017, pp. 260–266.

[22] M. Rahman, M. S. R. Kohinoor, and A. A. Sami, "Enhancing poultry farm productivity using iot-based smart farming automation system," in *Proceedings of the 26th International Conference on Computer and Information Technology (ICCIT)*. Cox's Bazar, Bangladesh: IEEE, 2023, pp. 1–6.

[23] ——, "Enhancing poultry farm productivity using iot-based smart farming automation system," in *2023 26th International Conference on Computer and Information Technology (ICCIT)*, 2023, pp. 1–6.

[24] O. M. Mico, P. B. M. Santos, and R. B. Caldo, "Web-based smart farm data monitoring system: A prototype," *J. Eng. Comput. Stud.*, vol. 3, no. 3, 2016.

[25] R. Dolci, "Iot solutions for precision farming and food manufacturing: Artificial intelligence applications in digital food," in *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, 2017, pp. 384–385.

[26] J. Jung, J. Lee, and H. Noh, "Web-based data analysis service for smart farms," *KIPS Transactions on Software and Data Engineering*, vol. 11, no. 9, pp. 355–362, 2022.

[27] S. Wolfert and G. Isakhanyan, "Sustainable agriculture by the internet of things - a practitioner's approach to monitor sustainability progress," *Computers and Electronics in Agriculture*, vol. 200, p. 107226, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169922005403