SMART GREENHOUSES DECISION SUPPORT SYSTEM FOR TOMATO CULTIVATION

Azees Asardeen

IT21231896

BSc (Hons) degree in Information Technology Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

April – 2025

SMART GREENHOUSES DECISION SUPPORT SYSTEM FOR TOMATO CULTIVATION

Azees Asardeen

IT21231896

Dissertation submitted in partial fulfillment of the requirements for the Special Honours Degree of Bachelor of Science (Hons) in Information Technology Specializing in Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology
Sri Lanka

April – 2025

DECLARATION

I declare that this is my own work and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Name	Student ID	Signature
Asardeen A.	IT21231896	

Signature of the Supervisor: Date

Mrs. Geethanjali Wimalaratne

…….. ….04/07/2025….

ABSTRACT

This project introduces a Smart Greenhouse Decision Support System designed for tomato cultivation, integrating Internet of Things (IoT) technology and Machine Learning (ML) to optimize irrigation efficiency. Addressing current shortcomings in greenhouse water management, the system collects real-time environmental data including temperature, humidity, light intensity, and soil moisture via IoT sensors. A Random Forest Regressor model processes this data to accurately predict the optimal water requirement in liters per plant, enabling precision irrigation. A user-friendly web interface was developed to visualize real-time sensor data and provide actionable recommendations to greenhouse operators. The system also monitors water availability using an ultrasonic sensor and generates alerts to prevent irrigation during water shortages. Field testing validated the system's performance, achieving prediction accuracy above 90% and demonstrating up to 30% water savings compared to traditional irrigation methods. The modular design supports scalability across different greenhouse environments, making it adaptable for broader agricultural use. Key findings highlight the critical influence of temperature and light intensity on water demand, the importance of accurate sensor calibration, and the role of user-centered interface design in facilitating adoption by non-technical users. This system contributes significantly to sustainable agriculture by minimizing water waste, improving tomato yield, and reducing labor costs. Future developments may include expanding support for multiple crop types, integrating AI-driven features, and enhancing environmental monitoring through additional sensor data such as soil pH and nutrient levels.

Keywords: Smart Greenhouse, Precision Irrigation, Internet of Things (IoT), Machine Learning, Tomato Cultivation.

ACKNOWLEDGEMENT

I am truly grateful to all those individuals who helped me through the research project process to finish 'Smart Greenhouses Decision Support Systems for Tomato Cultivation' successfully. I offer my sincere appreciation to my supervisor Mrs. Geethanjali Wimalaratne along with my co-supervisor Mr. Samantha Thelijjagoda for their essential guidance and continuous support that led to the development and improvement of this research study. The team leadership provided by their mentorship served as both a motivating force and a clearing pathway throughout the project development. I am thankful to my excellent team members because their commitment and teamwork made our work both purposeful and pleasurable. The Sri Lanka Institute of Information Technology (SLIIT) academic staff along with its technical team deserve my appreciation because they delivered vital support and facilities necessary for our system development and testing. The practical significance of this project gained significant value through the practical insights from both agricultural experts and greenhouse operators. I convey endless appreciation to my family together with my friends because they provided steadfast backing and patience while maintaining motivation throughout the research journey. The accomplishment stands as a result of the unifying skills and dedication from all personnel who jointly participated in this work. I truly express my appreciation to these personnel.

TABLE OF CONTENTS

DECLARATION i
ABSTRACT ii
ACKNOWLEDGEMENT iii
LIST OF FIGURES vi
LIST OF TABLES viii
LIST OF APPENDICES viii
LIST OF ABBREVIATIONS ix
1. INTRODUCTION 1
2. BACKGROUND & LITERATURE SURVEY 3
2.1 Background 3
2.2 Literature Survey 5
3. RESEARCH GAP 7
4. RESEARCH PROBLEM 9
5. RESEARCH OBJECTIVES 11
5.1 Main Objective 11
6. METHOTOLOGY 14
6.1 Requirement Gathering and Analysis 14
6.1.1 Hardware/Software Requirements: 14
6.1.2 Functional Requirements 15
6.1.3 User Requirements 16
6.1.4 Personnel Requirements 17
6.1.5 Non-Functional Requirements 17
6.1.6 Stakeholders: 18
6.1.7 Constraints And Limitations: 20
6.1.8 Feasibility Study 20
6.1.9 System Design 22
6.1.10 Key Pillars Of The Research Domain 31
6.2 Commercialization Aspects Of The Product 33
6.2.1 Market Opportunity 33
6.2.2 Target Market 33

6.2.3 Value Proposition 33
6.2.4 Revenue Model 34
6.2.5 Marketing And Sales Strategy 34
6.2.6 Competitive Analysis 35
6.2.7 Financial Projections 35
6.2.8 Regulatory and Compliance Considerations 36
6.2.9 Scalability and Future Development 36
6.3 Testing and Implantation 37
6.3.1 Implementation 37
6.3.2 Testing 72
7. RESULTS & DISCUSSION 76
7.1 Results 76
7.2 Research Findings 77
7.2.1 Stronger-Than-Expected Influence of Temperature and Light on Water Needs 78
7.2.2 Water Availability Monitoring Reduced Risk of Plant Stress 78
7.2.3 Sensor Calibration Critically Affected Data Quality And Model Accuracy 78
7.2.4 Positive User Feedback On Web Interface Usability 79
7.3 Discussion 79
7.3.1 Interpret the Results 79
7.3.2 Compare with Existing Work 80
7.3.3 Limitations 80
7.3.4 Future Improvements 81
8. CONCLUSION 82
9. PROJECT TIMELINE AND TASK ASSIGNMENT 83
9.1 Gantt Chart 83
9.2 Work Breakdown Chart 84
10. BUDGET AND BUDGET JUSTIFICATION 85
10.1 Budget Justification 85
11. REFERENCES 86
12. APPENDICES 89

LIST OF FIGURES

Figure 3 : System Overall Use Case Diagram 19
Figure 4 : Overall System Architecture Diagram 26
Figure 5 : Initial prototype of the IoT device without any safety casing 28
Figure 6: Developed IoT device with a plastic PVC protective cover 29
Figure 7 : Smart Greenhouse Decision Support System Workflow 30
Figure 8 : Flowchart for Integrated IoT and ML Based Irrigation System 32
Figure 9 : Environmental reading IoT device Development code 38
Figure 10 : IoT Devise Firebase Integration to send Live sensor readings 39
Figure 11: libraries used in this IoT project 40
Figure 12: Wi-Fi and Firebase Integration 41
Figure 13: Sensor Reading Code snipped 42
Figure 14: Device OLED Screen Outputs 42
Figure 15: Relay Control code snipped based on Firebase Update 43
Figure 16: Illustration of Overflow Detection Using Pot-Level Water Sensor 45
Figure 17: Automated Irrigation Motor Control conditions 45
Figure 18: Fan Automation Conditions 46
Figure 19: Alarm Buzzer Automation Control Conditions 47
Figure 20 : Firmware Execution Flow from Sensor Input to Actuator Output via Firebase 48
Figure 21: IoT device before and after casing for safety 49
Figure 22: Firebase Realtime DB showing live sensor values, automation settings, and device control status 51
Figure 23 : Dataset for Tomato Greenhouse Environmental and Irrigation Prediction 52
Figure 24: Flowchart for Random Forest model training and prediction process 53
Figure 25: Login Interface 55
Figure 26: Automation Control Interface with triggering Alert messages 57
Figure 27: Dashboard with Live Cards, Tank Visualization, and Automation Controls 57

Figure 28: Sample Calculation output: Direct Plant Count Logic 60
Figure 29: Image showing the Plant Gap (E.g.: Gap = 1.5m) 61
Figure 30: Sample Calculation output: Area-Based Plant Count Logic 62
Figure 31: Plant Count Calculation function using Area method 62
Figure 32: Water Requirement Prediction Interface with Live Sensor Autofill 63
Figure 33: Prediction Form with Output and Reasoning 63
Figure 34: Historical Prediction Graph and Log Table 64
Figure 35 : UI for water requirement prediction 66
Figure 36 : Datasets Preprocessing and cleaning 67
Figure 37 : StandardScaler and PolynomialFeatures implementation 68
Figure 38 : GridSearchCV hyperparameter tuning 69
Figure 39 : Code showing model saving process 69
Figure 40 : Console output displaying RMSE, R^2, and MAE. 70
Figure 41 : Web interface showing input fields 71
Figure 42 : Web interface showing predicted output. 71
Figure 43 : Gantt Chart 83
Figure 44 : Work breakdown structure 84

LIST OF TABLES

Table 1 : Research Gap Table: Comparison between existing systems..... 8
Table 2 : Test Case 01: Verify sensor data is stored in Firebase Realtime Database..... 73
Table 3 : Test Case 02: Water Prediction Accuracy..... 73
Table 4 : Test Case 03: Plant Count Calculation..... 73
Table 5 : Test Case 04: Alert Thresholds..... 74
Table 6 : Test Case 05: Watering Threshold Prediction..... 74
Table 7 : Component Budget Information..... 85

LIST OF APPENDICES

Appendix 1 : Plagiarism Report (Turnitin)..... 89

LIST OF ABBREVIATIONS

Abbreviation	Description
API	Application Programming Interface
DB	Data Base
DHT22	Digital Humidity & Temperature Sensor
DSS	Decision Support System
ESP32	Espressif Systems Microcontroller (32-bit)
HTML	Hyper Text Markup Language
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
IWUE	Irrigation Water Use Efficiency
JS	Java Script
JSON	JavaScript Object Notation
LDR	Light Dependent Resistor
LDR	Light Dependent Resistor
LoRaWAN	Long Range Wide Area Network
MAE	Mean Absolute Error
ML	Machine Learning
pH	Potential of Hydrogen (used in soil acidity)
R^2	Coefficient of Determination (R-squared)
RMSE	Root Mean Squared Error
UAT	User Acceptance Testing
UI	User Interface
URL	Uniform Resource Locator
UX	User Experience

1. INTRODUCTION

Current agricultural practices depend heavily on efficient water management especially within greenhouses because controlling the environment results in superior crop production. Significant research efforts within smart agroecology focus on creating IoT-based irrigation systems which implement machine learning functionality for optimizing water efficiency. The systems track environmental factors including soil moisture together with temperature humidity along with light intensity to deliver data-based guidance for better watering plans. The available irrigation systems have accomplished better water management but still fail to accurately determine precise plant water requirements across different environmental situations which results in poor crop production outcomes.

A successful project execution demands thorough knowledge about IoT technologies together with machine learning algorithms in addition to their agricultural applications. The successful implementation of this system depends on understanding environmental sensor networks together with data analytics along with web application development since real-time data collection and user-friendly decision-making interfaces must be integrated. In order to achieve accurate and beneficial crop growth results the system needs expert knowledge about plant physiology specifically tomato plant water requirements under different environmental scenarios.

The modern state of adaptive irrigation systems concentrates on efficient water management through automated and partially automated irrigation procedures. The application methods precision irrigation and variable rate irrigation that use controlled water delivery to root zones now receive broad research attention and practical implementation. The implementation of standard irrigation schedules together with insufficient environmental variable monitoring results in poor water resource management.

Researchers have dedicated efforts to resolve these problems by designing neural network systems for crop water simulation and real-time feedback monitoring control loops for

irrigation system adjustments. These promising methods fall short when used across diverse greenhouse facilities because they do not meet requirements for widespread application or adjustments to different growing conditions.

Our system implements a decision support mechanism that elaborates on past research by assessing present environmental information for precise watering planning and checks water availability for irrigation purposes. The system merge IoT sensors with advanced machine learning models to generate precise context-specific recommendations which adapt easily across a range of greenhouse conditions to boost tomato cultivation's water management and yield outcomes.

2. BACKGROUND & LITERATURE SURVEY

2.1 Background

The Solanum lycopersicum plant species which include tomatoes represent valuable agricultural products that need specialized water management protocols for reaching their maximum potential. Both over-watering and under-watering of tomato plants result in adverse effects that require maintaining a proper water quantity. When irrigating excessive amounts water destroys roots through diseases which reduces soil oxygen content leading to impaired nutrient absorption until when irrigation is insufficient the plants face water stress which shrinks fruit size and totals up to lower yields [1] [2].

When compared to control treatment (100% ETo), deficit irrigation (under irrigation) led to lower vegetative growth and decreased flowering along with reduced fruit yield parameters and photosynthetic pigments and mineral content levels as well as leaf relative water content and membrane stability index of tomato plants. Water deficit treatment increased tomato leaf proline content as well as irrigation water use efficiency (IWUE) and certain quality metrics of fruits [3]. Frequent irrigation doses enhanced vegetative growth together with stomatal conductance and CO2 assimilation and transpiration along with higher yields [4].

Rising human activities combined with increased CO2 concentrations heat the environment through building up the greenhouse effect. The rising levels of greenhouse gases play a crucial role in creating seasonal instability because of excessive energy usage. Various levels of microclimate influence various plants in a smart greenhouse. The microclimate needs special design arrangements to create conditions that support ideal plant development [5].

Greenhouse management heavily depends on temperature control as a vital factor. The ideal temperature range for tomatoes matches between 21°C and 27°C (70°F and 82°F) [6].

A temperature setting of 26 °C proves better than 32 °C for tomato cultivation within greenhouses having high frequency irrigation (FR1) as the best watering method. The irrigation management factor influences results more strongly when the temperature is set at 26 °C than at 32 °C according to study findings [7]. The deviation from this temperature range leads to negative influences on plant water intake and metabolic processes. The plants demand more water during hot periods because heat increases their water requirements yet excessive cold reduces their ability to absorb water and nutrients. Proper water management needs stable environmental conditions because they prove essential for effective water management [8].

Greenhouses allow users to control environmental conditions which they can use for maximizing crop growth efficiency. Furnishing these specific conditions demands both accurate observation and comprehensive management practices. Traditional irrigation techniques show limited capability to adjust watering practices based on immediate changes in the environment thus producing decreased productivity. The growing necessity requires sophisticated systems that use real-time input data for making rational irrigation choices [9].

Manual control methods of greenhouse environment management produce substandard conditions that reduce plant yields because of human involvement. The development of the Internet of Things (IoT) brought about self-driven monitoring and control systems for applications. The real-time monitoring capabilities of IoT-based smart greenhouses allows them to control environmental factors by adjusting temperature and humidity alongside other growing variables. Mist cooling systems that integrate IoT technology help control extreme temperatures and humidity which optimizes conditions for tomato cultivation [10] [11].

The deployment of IoT technology in agriculture modernizes crop management by enabling specific climate control systems and it facilitates forecast service for harvest times and boosts operational efficiency. IoT sensors along with smart greenhouse automation defend plants from unfavorable weather conditions and create perfect cultivation settings which has become an essential agricultural advancement [10][11].

2.2 Literature Survey

Agricultural water resources became more manageable because smart irrigation systems developed during recent years. Smart irrigation systems with Internet of Things sensors constantly measure environmental factors such as soil moisture along with temperature and humidity levels and light quantity. The acquired data enables automated irrigation functions that properly supply water to plants based on their current requirements [7][8].

The implementation of IoT-based systems improves irrigation efficiency because they deliver water based on measured real-time data. Such irrigation systems demonstrate the capacity to cut water consumption by 30% while maintaining existing or exceeding crop productivity levels. Soil moisture sensors form a part of the system by monitoring conditions to process data for scheduling updates [9][10].

Machine learning algorithms represent a modern technological solution that improves irrigation practice methods. Through a combination of historical and present-day information processing these algorithms achieve the prediction of ideal crop watering needs across different environmental situations. Liu et al. (2019) showed how machine learning models enhance irrigation scheduling accuracy through pattern-based data learning which results in optimized water utilization according to their research [11][12].

Today there exists no operational solution for scheduling precise watering times along with exact liquid volume requirements which vary by daily environmental conditions. Numerous smart irrigation systems determine soil moisture levels as their primary measurement point but fail to incorporate additional environmental aspects that include temperature data along with humidity factors and light conditions. The systems fail to present an extensive solution to adapt to changing plant requirements because of this operational constraint [13][14].

An advanced smart irrigation system needs development to address present technological shortcomings which currently allow real-time environmental monitoring but fail to predict precise watering times and amounts. The resolution of this technical deficiency would boost water efficiency rates alongside plant health and crop yield improvements mainly in greenhouse-controlled environments [15][16].

3. RESEARCH GAP

The existing IoT-based solutions for precision agriculture show major shortcomings when used in tomato greenhouses affecting overall crop management effectiveness. A comprehensive evaluation of present systems shows that multiple essential shortcomings must be solved to improve irrigation functioning.

The Research A ‘IoT-Based Precision Irrigation System’ provides automated water control and environmental monitoring while missing the fundamental feature of greenhouse temperature regulation. Monitoring water resources is one capability of this system yet it does not offer either watering schedule or water quantity predictions under changing environmental conditions. The system faces performance difficulties in controlling the balance between adequate, excessive water use because of this particular shortcoming [12].

Water irrigation through the Research B ‘Smart High-Yield Tomato Cultivation System’ features automation yet its performance weakens when monitoring temperatures. The system operates without predictive analytics for watering optimization thus it depends on reactive measures which neglect future environmental changes [13].

The Research C ‘Indoor seed germination system’ comprises an important deficiency since it lacks automated water irrigation together with multi-factor environmental monitoring and predicting abilities. Due to its restricted design the system lacks sufficient capability to fulfill all the requirements of greenhouse tomato cultivation which demands detailed environmental factor control [14].

The Research D 'LoRaWAN-based IoT System provides monitoring strength along with real-time dashboard alerts yet lacks controls for temperature adjustments and smart watering program predictions. The system becomes ineffective at managing optimal growing conditions because it lacks necessary features which would prove essential during extreme temperature fluctuations [15].

Studies that apply LoRaWAN technology in greenhouse precision agriculture show how wireless communication along with low power usage work effectively for environmental monitoring systems. The current implementations emphasize data collection and lack predictive irrigation guidelines processed by machine learning methods which hinders their ability to maximize water efficiency proactively [16].

Predictive analytics reveals its unique ability to enhance deficit irrigation strategies according to new research about machine learning systems in processing tomato irrigation management. The study proves how data-driven decision making systems work well in water management yet confirms a requirement for advanced drought and heat stress recognition models because existing tools fail to accurately represent environmental plant-performance influencing factors [17].

A research gap currently exists because there is a need to develop a decision support system which merges environmental real-time monitoring data with predictive analytics to find the right amount of water for greenhouse tomato plants. The development of proper solutions must be urgent because it enables better water management to prevent irrigation problems and promote sustainable agricultural operations.

Existing Systems / Research	Environmental Monitoring	Water Avaiablity Monitoring & Control	machine learning integration	Overflow Detection	Water recruitment prediction	Automation Features & Control	Real time dashboard
IoT-Based Precision Irrigation System	✅	✅	❌	❌	❌	✅	❌
Smart Tomato Cultivation System	✅	✅	❌	❌	❌	✅	✅
Indoor Seed Germination System	✅	❌	❌	❌	❌	❌	❌
LoRaWAN-Based IoT Monitoring	✅	❌	❌	❌	❌	❌	✅
Proposed System (This Study)	✅	✅	✅	✅	✅	✅	✅

Table 1 : Research Gap Table: Comparison between existing systems

4. RESEARCH PROBLEM

Effective water management means high yields of the crops and sustainable utilization of resources in conditions of hydroponic vegetation. Conventional irrigation methods frequently bring about water mismanagement and suboptimal growth in a plant because of the potential to point out a match for environment after environment along with actual temporal terms. By merging Internet of Things (IoT) technologies and machine learning (ML) models a promising solution could be introduced by allowing precise monitoring and forecasting of plant water needs.

Challenges in Current Greenhouse Irrigation Practices

Existing greenhouse irrigation systems frequently encounter several issues:

- When schedules are manually controlled there is excessive water usage as they do not adapt to daily environmental changes [18].
- Without real-time adjustments plants develop inconsistent health which results in both inadequate and excessive watering methods because of stress which contributes to yield reduction [18].
- Operation efficiency suffers because manual and set irrigation schedules make automated monitoring and control absent which requires continuous human attention and creates both higher labor expenses and the potential for human mistakes [18].

Limitations of Current Technological Approaches

While IoT-based greenhouse monitoring systems have been developed, challenges persist:

- The processing and analysis of vast sensor data needs specialized data management systems to operate with efficiency [19].

- The integration of different data sources together with ensuring interoperable connections between IoT devices proves difficult to manage technically [19].
- The development of machine learning models which can make accurate water requirement predictions remains difficult across various environmental conditions and plant types [19].

Addressing the Research Problem

A project has been initiated to construct an IoT device and advanced machine learning model which determines precise watering needs by processing real-time environmental information to maintain water supply within tomato greenhouses. The system tracks five vital parameters including soil moisture, temperature, humidity, light intensity and water tank levels in order to:

- The system will provide exact watering instructions which leads to reduced water waste and proper plant hydration.
- A proper water management strategy allows users to achieve better crop yields and product quality through correct soil moisture maintenance.
- The system provides an intuitive web application for live monitoring combined with analytical capabilities that diminish human labor requirements alongside their associated operational expenses.

Through IoT and Machine Learning joint utilization this solution expects to address irrigation practice constraints for more sustainable and efficient greenhouse complex management. The system relies on IoT together with Machine Learning to provide enhanced tomato greenhouse environment control.

5. RESEARCH OBJECTIVES

The research develops a decision support system with sensors to deal with precision irrigation control problems that affect tomato greenhouse management. The system combines IoT sensors with contemporary water resource monitoring and prediction models from ML to analyze environmental data shortly and determine optimized water need predictions. Web-based interface technology of this system provides both predictive analysis and real-time data visualization. The research aims to create new non-invasive tools that support advanced farming practices through these objectives.

5.1 Main Objective

To develop a smart IoT and machine learning-based decision support system that predicts optimal water requirements (in liters) for tomato greenhouses using real-time environmental conditions, ensures water availability, prevents overflow, automates control systems, and provides a live dashboard for prediction, monitoring, and data logging.

5.2 Specific Objectives

1. To collect and monitor real-time environmental data using IoT sensors.
    - This objective involves deploying sensors in the greenhouse to measure temperature, humidity, soil moisture, light intensity, and pot-level water status. The ESP32 device will upload this data to Firebase for real-time access and decision-making.

2. To build a regression-based machine learning model to predict irrigation requirements.
    - The objective is to train a machine learning model that takes environmental input values and predicts water needs per plant. The system outputs both individual and total water requirements based on plant count or greenhouse area.

3. To monitor water tank level in real-time using an ultrasonic sensor.
    - This objective includes integrating an ultrasonic sensor to track tank water levels. The sensor data will be used to determine whether sufficient water is available before allowing irrigation, and to notify users when refilling is needed.

4. To implement pot-level water overflow detection for safety and efficiency.
    - This objective introduces a novel feature that uses a water level sensor in the plant pot to detect overflow during irrigation. When overflow is detected, the system automatically stops watering to prevent water waste and plant damage.

5. To automate irrigation, fan, and buzzer control using sensor-based logic.
    - This objective aims to automatically operate irrigation, cooling fan, and buzzer based on thresholds such as soil dryness, high temperature, or water tank level, improving efficiency and reducing manual work.

6. To develop a user-friendly web dashboard for live monitoring and control.
    - The interface will show real-time sensor values, predicted irrigation requirements, tank level visuals, and allow manual or automatic control of devices. The dashboard also includes alert notifications for critical conditions.

7. To offer both direct and area-based plant count methods for flexibility.
    - This objective includes a calculator tool that lets users either input exact plant numbers or greenhouse size and plant spacing to automatically estimate the plant count, which improves adaptability for different greenhouse setups.

8. To store and visualize historical irrigation predictions with export options.
    - All predictions will be logged with a timestamp in Firebase. Users can view the data through a line chart and table, and export it as a CSV for offline records or reporting.

9. To test the system in real greenhouse conditions to assess performance.
    - The final goal is to evaluate how effectively the system manages water and responds to changing environmental conditions compared to traditional irrigation methods, especially in terms of water savings and usability.

6. METHOTOLOGY

6.1 Requirement Gathering and Analysis

Understanding what the project must accomplish depends on the requirements collection and analysis phase. The development process can go smoothly and with a clear direction if the project's objectives, stakeholders, hardware, software, and limits are all clearly defined. This section describes the analysis and structuring of various components to guarantee the system's success.

Objective:

The main purpose of this project focuses on developing and implementing a decision support platform which determines proper irrigation demands while maintaining water accessibility in tomato greenhouses by using live environmental measurements including soil wetness and temperature alongside humidity levels and intuitive monitoring tools. The system targets three main objectives which include better crop production and less water consumption and easy web-based interface management.

6.1.1 Hardware/Software Requirements:

The hardware and software requirements were carefully chosen to fulfill the objectives of the project:

Hardware:

- **ESP32:** The IoT device that collects data from the sensors and controls the irrigation system via a relay.
- **Relay:** A relay is used to activate the irrigation system based on predictions from the ML model.

- **OLED Display:** OLED display integrated with IoT device that used to show the all the environment sensor readings.
- **5V Power Supply:** The system requires a stable 5V power supply for continuous operation.
- **Sensors:**
    - Soil Moisture Sensors - for monitoring soil water levels.
    - Temperature Sensors - to track the greenhouse's internal temperature.
    - Humidity Sensors - for measuring air moisture content.
    - Light Sensors - to monitor light intensity.
    - Ultrasonic Sensor - for water tank's water level measurements.

Software and Database Requirements:

- **Firebase DB:** Used for data storage, real-time updates, and communication between the IoT device and the web interface.
- **Python (ML Libraries):** Python is used to implement the **Random Forest Regression** model, which predicts the water needs of the plants.
- **Web Interface:** A user-friendly web dashboard is developed to allow real-time monitoring of environmental data and control irrigation settings.

6.1.2 Functional Requirements

Functional requirements outline the specific functionalities that the system must have to meet the project's objectives. For the Smart Greenhouse Decision Support System, these include:

1. Real-time Data Collection - The system must collect real-time data on soil moisture, temperature, humidity, light, and water turbidity using IoT sensors.

2. Automated Control - The system should automatically control greenhouse actuators, such as turning on water pumps when soil moisture is low or activating cooling fans when the temperature is high.
3. Machine Learning Predictions - The system should predict optimal watering schedules based on historical and real-time environmental data using a machine learning model.
4. Data Storage and Retrieval - The system must store sensor data in a Firebase database and allow real-time data retrieval for analysis and decision-making.
5. User Interaction - The system must provide a user-friendly web interface that allows users to monitor environmental conditions, view predicted watering schedules, and manually control greenhouse components.
6. Alert System - The system should send alerts or notifications based on critical environmental thresholds or predictions.

6.1.3 User Requirements

User requirements focus on the needs and expectations of the end users of the system. For this project, the users include greenhouse operators, farmers, and agricultural experts. Key user requirements include:

1. Ease of Use - The interface should be intuitive, with simple navigation and clear instructions, allowing users to interact with the system without needing advanced technical skills.
2. Customizability - Users should be able to customize the system based on their specific greenhouse setup, such as adjusting threshold values for sensors or modifying watering schedules.
3. Accessibility - The system should be accessible from any device with an internet connection, allowing users to monitor and control the greenhouse remotely.

4. Real-time Feedback - Users need real-time feedback on the status of the greenhouse environment, including sensor readings and system actions.
5. Security - User data and system operations should be secure, with authentication mechanisms to protect against unauthorized access.

6.1.4 Personnel Requirements

To enhance the quality, knowledge, continuation and integrity of the research the following are the required personnel.

- Department of Agrarian Development in Sri Lanka
- Mr. Krishantha Jayawardhana, Agrarian Services Center, Monaragala.
- Department of Agriculture Sri Lanka .

6.1.5 Non-Functional Requirements

Non-functional requirements address the quality attributes of the system, such as performance, security, and usability.

1. Performance - The system must process and respond to sensor data in real-time, with minimal latency in executing automated controls.
2. Reliability - The system must operate consistently with high uptime, ensuring critical environmental conditions are always monitored.
3. Scalability - The system should be scalable to accommodate additional sensors or functionalities in the future.
4. Security - The system must incorporate security measures to protect user data and ensure the integrity of system operations.
5. Maintainability - The system should be easy to update and maintain, with clear documentation for both software and hardware components.

These requirements provide a comprehensive foundation for the successful design, development, and deployment of the Smart Greenhouse Decision Support System, ensuring that it meets both user needs and technical specifications.

6.1.6 Stakeholders:

The following stakeholders were identified in order to guarantee the creation of a solution that satisfies practical needs:

- **Project leaders:** The in charge of overseeing the organization's overall management and making sure the system achieves its goals.
- **Domain Experts:** These specialists, especially in the field of agriculture, offered vital information about tomatoes' water requirements and how to use technology to maximize irrigation.
- **End Users:** Farmers or greenhouse managers will be the system's main users, interacting with it to control irrigation schedules and monitor soil conditions.
- **Additional Stakeholders:** This comprises cloud service providers and IoT device suppliers, who are crucial in supplying the project's hardware and software requirements.

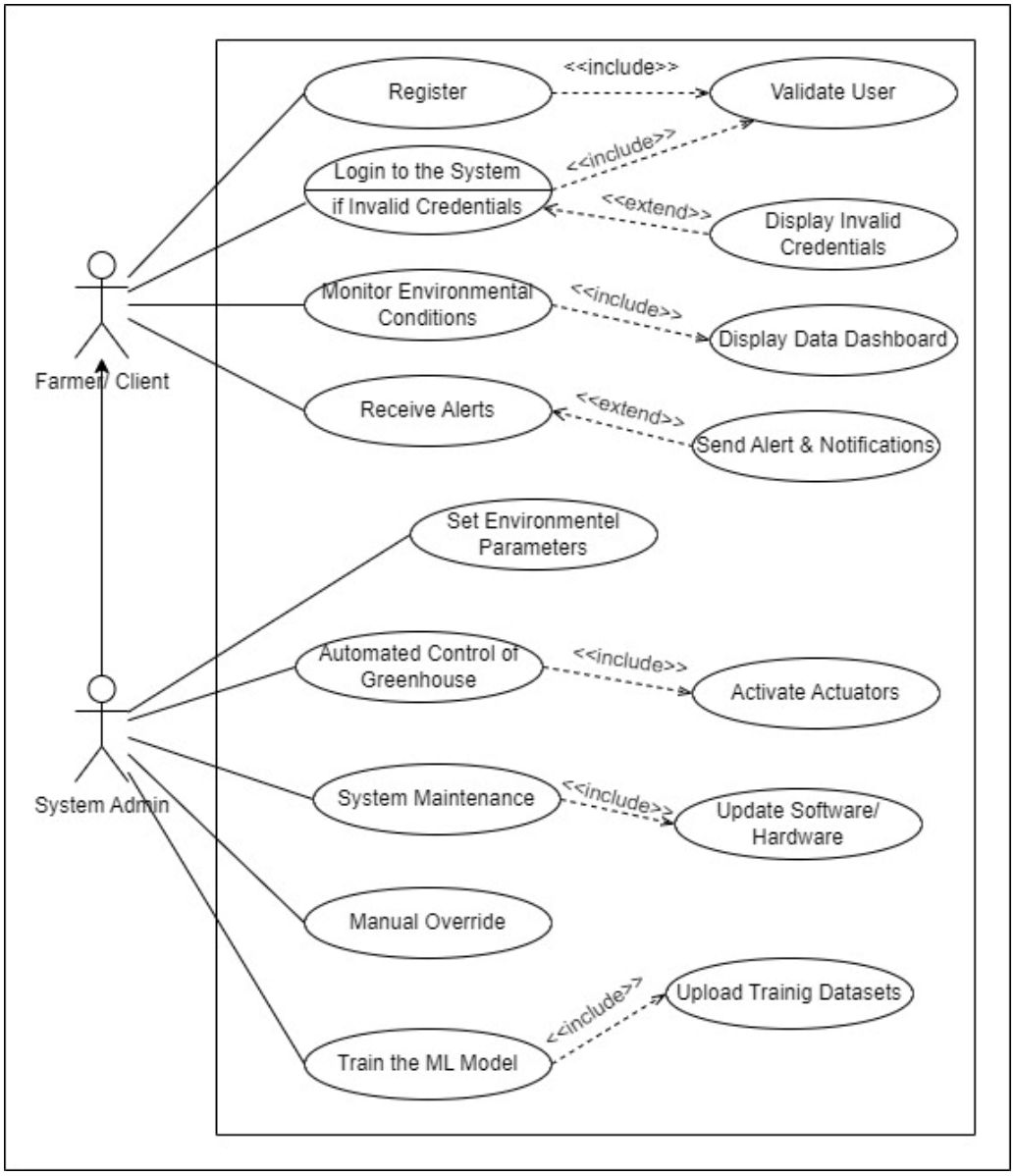


Figure 1 : System Overall Use Case Diagram

6.1.7 Constraints And Limitations:

The development of this project encountered several constraints and limitations:

- **Budget:** The hardware components were selected based on the available budget, and cost-effective solutions were prioritized to ensure affordability while maintaining functionality.
- **Time:** The project had a strict timeline, which limited the scope of field testing and deployment in real-world conditions.
- **Technical Limitations:** The accuracy of the sensors could be affected by environmental conditions, and the range of Wi-Fi for communication with the cloud services was a consideration.
- **Environmental Factors:** The IoT system had to be robust enough to function in outdoor environments, including temperature fluctuations, rain, and humidity, which may affect sensor performance.

6.1.8 Feasibility Study

Modern tomato production has undergone a transformation through Internet of Things (IoT) technologies because they enable real-time environmental tracking along with precise environmental controls. Greenhouse plant managers benefit strongly from this development because they need to maintain perfect environmental conditions to achieve optimal yield results.

Technical Feasibility:

Research demonstrates that greenhouse applications will benefit from implementing the combination of IoT sensors such as light intensity sensors and temperature and humidity

sensors and soil moisture sensors connected to ESP32 microcontrollers. Real-time data collection and transmission functions of these tools enable continuous observation of critical elements that affect tomato plant development. A study demonstrated how to complete effective greenhouse management through ESP32-based monitoring and regulation that proved the system's agricultural value.

Economic Feasibility:

Small to medium-sized tomato greenhouse businesses obtain favorable returns on investment by implementing monitoring systems driven by Internet of Things technology. The startup expenses remain affordable because ESP32 microcontrollers and additional sensor components maintain low prices. A positive return on investment may occur because increased crop yields along with more efficient resources will bring better financial results. Scientific research indicates that these monitoring systems enhance tomato cultivation while improving profitability which proves their business value.

Operational Feasibility:

Users benefit from easily accessible interfaces that enable farmers and greenhouse managers to both review environmental data while also assessing it. Real-time dashboards and mobile applications supply actionable insights which enable users to respond quickly to optimize growing conditions. Addressable systems with real-time sensor data monitoring capabilities exist specifically for improving greenhouse operation efficiency by controlling lighting and watering protocols.

The implementation of IoT technologies serves as a current method to enhance sustainability while boosting production numbers in tomato greenhouse operations. The implementation of this technology proves both possible through innovation and offers operational benefits as well as economical advantages.

6.1.9 System Design

Through its implementation the technology enables precise water need predictions with real-time monitoring capabilities for tomato greenhouses. The integration of ML algorithms with IoT devices allows this achievement. The design includes both software components and hardware components which are configured to provide seamless data transfer and user interface.

6.1.9.1 Overall System Architecture

The following architecture diagram represents the complete structure and real-time data flow of the **IoT-based Smart Greenhouse Decision Support System**. It shows how hardware, cloud services, machine learning, and a web application are seamlessly integrated to monitor environmental conditions and automate irrigation decisions for optimized tomato cultivation.

System Breakdown:

1. **IoT Device Layer (Sensor + Actuator Control Unit)**

The system uses an **ESP32 microcontroller**, powered by a 5V source, to act as the brain of IoT operations. It connects multiple sensors and actuators:

Sensors:

- **Soil Moisture Sensor** – Measures moisture in soil to detect when watering is needed.
- **Temperature Sensor** – Detects environmental temperature within the greenhouse.
- **Humidity Sensor** – Measures relative humidity.
- **Light Sensor** – Captures light intensity (in lux), important for understanding plant water demand.

- **Ultrasonic Sensor** – Installed in the tank to measure water height and calculate percentage.
- **Water Level Sensor** – Measures real-time water level in the pot (optional overflow detection).

All readings are sent to the **ESP32**, which pushes the data to the cloud via Wi-Fi.

Actuators:

- **Relay Module** – Controls switching of external devices based on automation logic.
- **Water Pump** – Automatically turns ON when irrigation is triggered by sensor conditions.
- **Cooling Fan** – Activates during high temperatures or low humidity to improve air circulation.
- **Heater** – Can be used to prevent frost under low-temperature conditions.
- **DC OLED Display** – Optionally displays live sensor readings locally for debugging or field status.

2. **Cloud Database (Firebase Realtime DB)**
    - The ESP32 sends all live sensor data to **Firebase**.
    - Firebase also stores:
        - Real-time sensor values
        - Prediction logs (from ML model)
        - Device status (pump, fan, alarm)
        - Automation toggle states (ON/OFF)

3. Web Application Layer

The web application, built using **Flask (Python)** and hosted locally, connects to Firebase for both:

- **Live monitoring**, and
- **Control logic execution**.

ML Model & Logic Processing

- The backend loads a **trained Random Forest model**.
- Uses polynomial transformation and scaling on environmental inputs.
- Predicts:
    - **Per plant water requirement (Liters)**
    - **Total water needed** based on plant count
- Decision logic also includes **reasoning system** that explains predictions to users (e.g., "Low soil moisture" or "High temperature").

4. User Interfaces (UI Modules)

The web application includes 5 main UI panels, shown in the diagram:

a. Display Sensor Readings

- Real-time dashboard with live readings of all environmental parameters.

b. Water Requirement Prediction

- A user-friendly form to enter or auto-load values.
- Supports **manual plant count or area-based calculation**.
- Shows output in liters per plant and total liters needed.

c. Automation Control Panel

- Users can toggle:
    - **Irrigation Automation**
    - **Fan Automation**
    - **Alarm/Buzzer Automation**
- All changes are pushed to Firebase and update the ESP32 instantly.

d. Alerts & Notifications

- System generates alerts for:
    - Tank below 10%
    - Temperature > 40°C
    - Dry soil without irrigation
    - Water overflow risk
- Alarm automation turns on a **buzzer** to warn the operator when any alert is triggered.

e. Historical Data Visualization

- Displays a **line chart** of past predictions using **Chart.js**.
- All data from previous predictions are shown in a table.
- Option to **download logs as CSV** for analysis/reporting.

5. End-User Interaction

Users interact with the system through the web interface:

- Accessible on laptops, tablets, or phones.
- Can control the system manually or monitor it while automation is active.
- Ideal for **non-technical users** thanks to a clear interface and alert system.

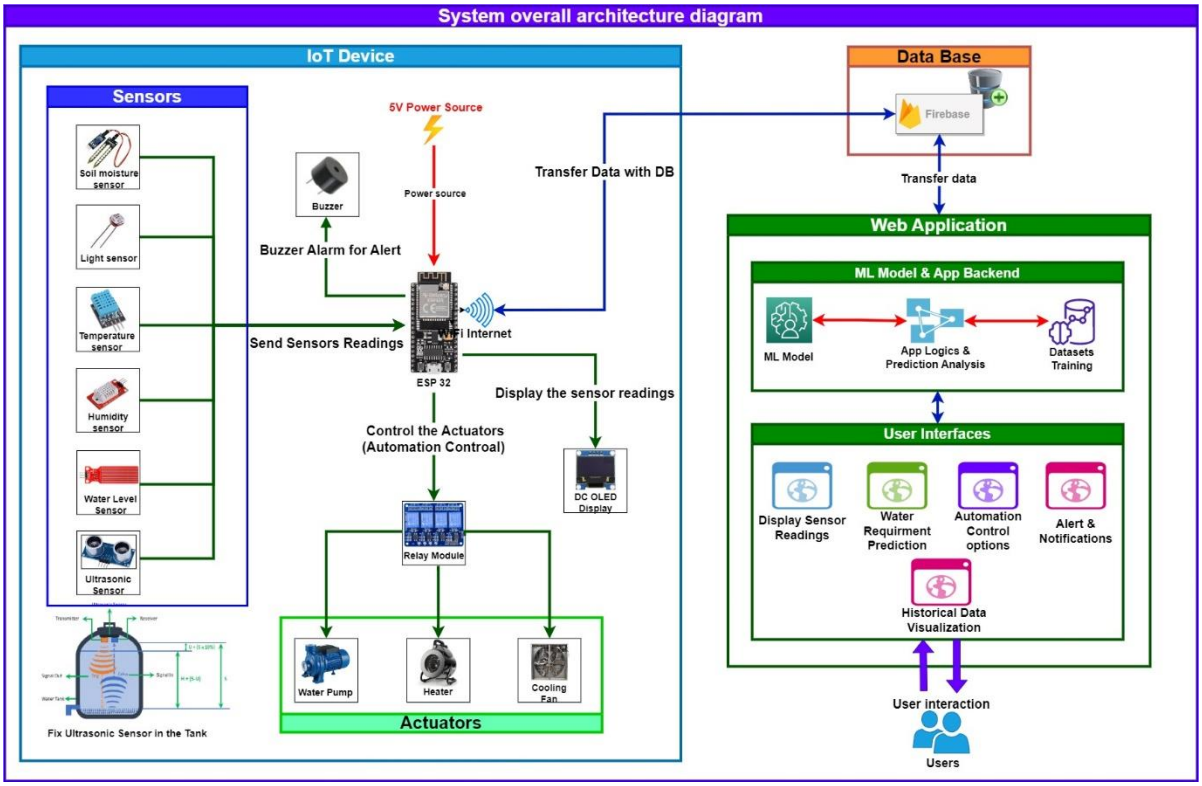


Figure 2 : Overall System Architecture Diagram

6.1.9.2 Component Design

The design of the system is modular, with separate hardware and software components that work together to accomplish the intended purpose.

Hardware Components

- **Sensors**:
    - *Soil Moisture Sensor*: Measures the volumetric water content in the soil, providing data crucial for irrigation decisions.
    - *Temperature and Humidity Sensor (DHT11)*: Monitors ambient temperature and humidity levels within the greenhouse.

- *Light Intensity Sensor*: Assesses the amount of light available to the plants, influencing photosynthesis rates.
- *Ultrasonic Sensor*: Measures the water level in the storage tank to ensure adequate water availability for irrigation.

- **Microcontroller**:
  - ***ESP32***: Serves as the central processing unit, interfacing with all sensors to collect data and transmit it to the database via Wi-Fi connectivity. The ESP32 is programmed to handle multiple sensor inputs and ensure efficient data communication.

Device Assembly And Design Considerations

An IoT device controls automated functionality by combining sensors such as the soil moisture guide together with ultrasonic level measurement hardware and a microcontroller and relay system. Each sensor and module was integrated into a small yet operational design as the first development step.

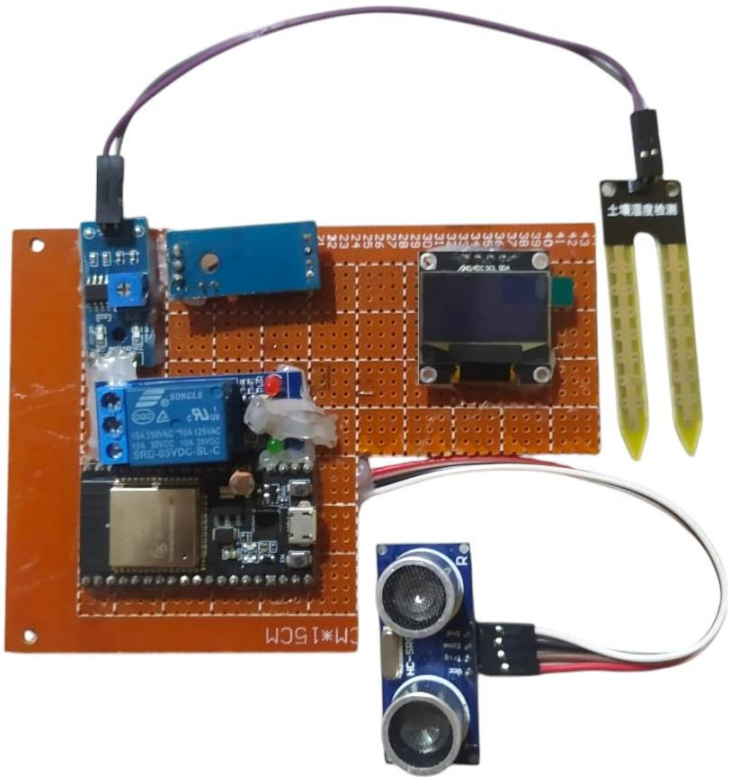


Figure 3 : Initial prototype of the IoT device without any safety casing

The tested prototype functioned properly yet it left essential components and wiring vulnerable to harm which would be problematic during agricultural field usage due to common water exposure and dust accumulation and physical interactions.

The designed custom plastic enclosure addressed the detected risks. This casing serves multiple purposes:

- **Safety:** Prevents accidental contact with electronic components, protecting both the user and the device.
- **Durability:** Shields the system from moisture, dust, and environmental stress.
- **Portability:** Enables easier handling and relocation of the system in greenhouse or farm settings.
- **Commercial Readiness:** Adds a layer of professional finish, supporting future deployment at scale.

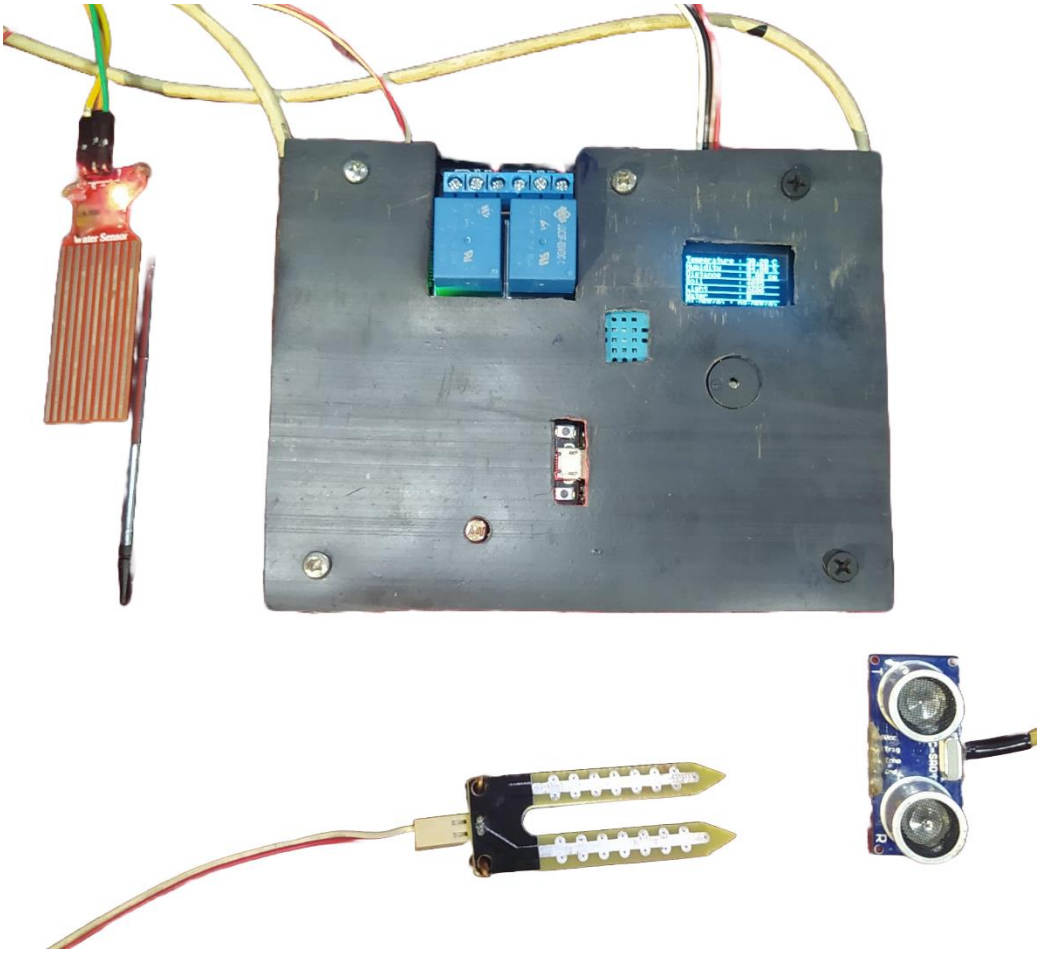


Figure 4: Developed IoT device with a plastic PVC protective cover

The plastic enclosure maintains internal wire security through its ports design which simultaneously allows LED indicator visibility. Only necessary sensors that need environmental interaction were allowed through designed openings on the casing.

Software Components

- **Data Storage and Management**:
    - **Firebase Realtime Database:** Stores the incoming sensor data, providing a scalable and synchronized platform for real-time data management.

- **Machine Learning Model**:
    - **Random Forest Regressor:** Researchers employed this method to analyze past and present environmental records which resulted in determining accurate water needs for tomato plants. This algorithm combines different decision tree outputs to boost prediction precision during the entire process.
- **Web Interface**:
    - **Dashboard Application:** A web-based system developed with HTML, CSS, JavaScript technologies shows real-time sensor outputs and predictions from ML-based water requirement algorithms. Users can perform effective greenhouse condition monitoring through the interface which was designed for easy navigation..

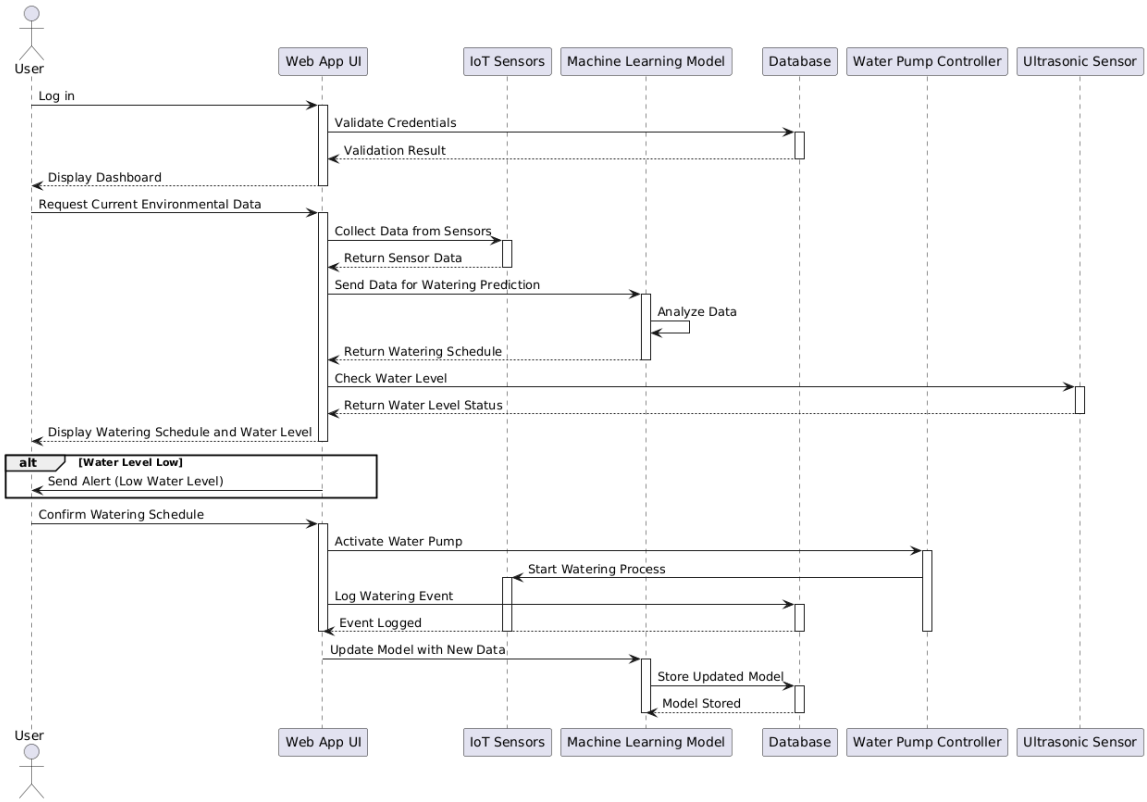


Figure 5 : Smart Greenhouse Decision Support System Workflow

6.1.10 Key Pillars Of The Research Domain

Integrating machine learning (ML) models with Internet of Things (IoT) technology has become essential for improving sustainability and efficiency in modern agriculture. The importance of these fundamental pillars in creating an intelligent irrigation system for tomato greenhouses is examined in this section.

Internet of Things (IoT) in Agriculture:

- The application of IoT sensors in agricultural fields produces results we refer to as "smart agriculture." These sensors enable farmers to obtain beneficial information about temperature alongside humidity and light intensity and soil moisture through real-time data collection for method improvements. Using soil sensors enables sustainable water management because they establish precise irrigation timing which ensures crops absorb adequate water at maximum efficiency. IOThelped solutions make field monitoring and control through remote methods possible while simultaneously boosting operational efficiency together with decision-making capability [20].

Machine Learning for Water Requirement Prediction:

- Internet of Things platforms allow predictive analytics in agriculture through the integration of Random Forest Regressor machine learning models and other models from the field of machine learning. The models demonstrate exact prediction abilities for crop water needs during their analysis of historical and present environmental data. The implementation of predictive irrigation systems modifying water systems according to forthcoming requirements leads to both water conservation and increased crop production [20]. Massive potential exists for precision agriculture evolution because IoT data collection teams up with machine learning analysis methods.

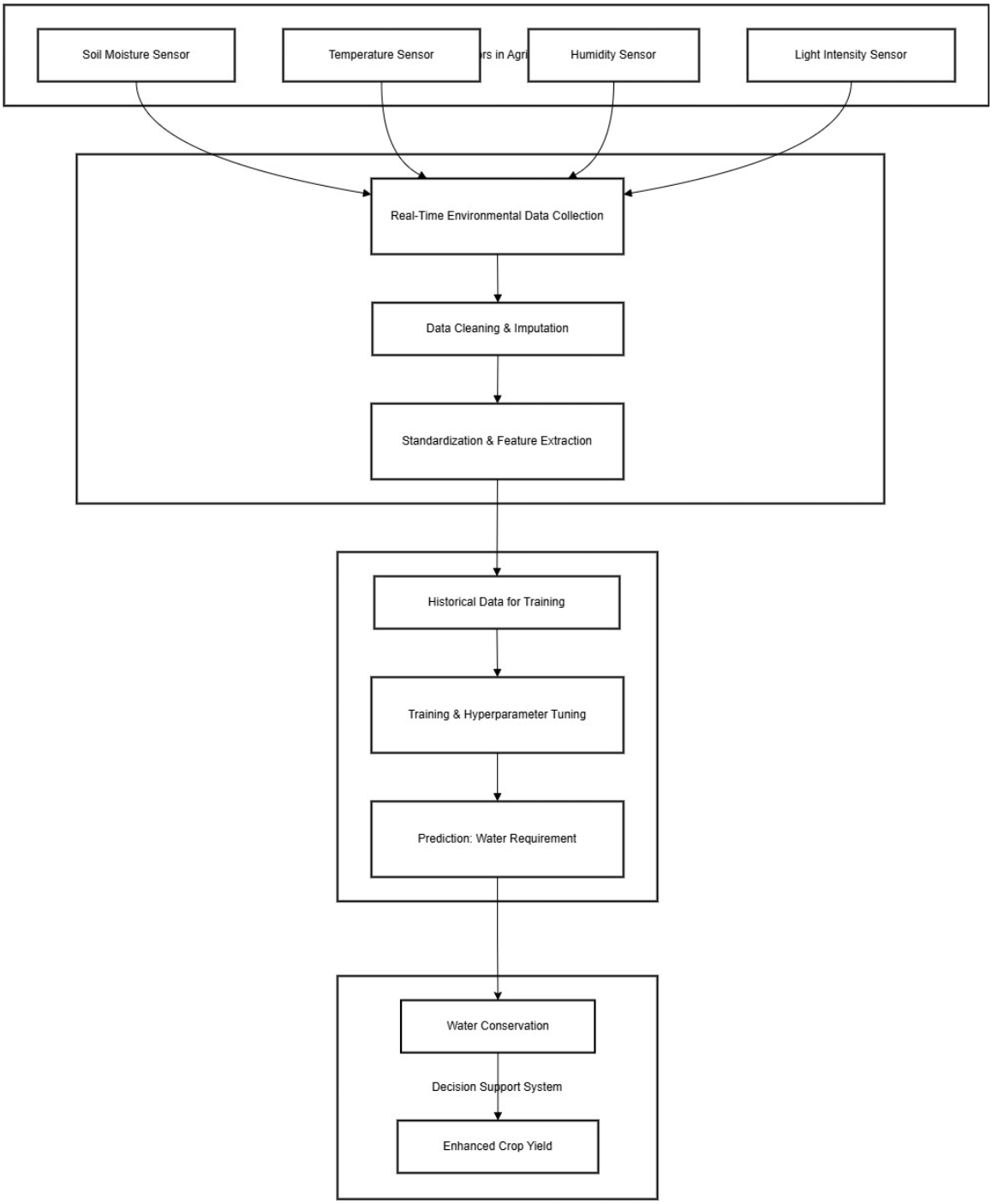


Figure 6 : Flowchart for Integrated IoT and ML Based Irrigation System

6.2 Commercialization Aspects Of The Product

6.2.1 Market Opportunity

The global agriculture sector adopts technology at an advancing pace to build sustainability and boost production outcomes. The market potential for IoT together with machine learning driven smart greenhouse systems is significant. The systems solve important problems associated with climate variability together with resource inefficiencies. Through the implementation of IoT sensors Green Leaf Farms accomplished a 30% reduction in water consumption that produced both cost reduction benefits along with sustainability advantages.

6.2.2 Target Market

The primary target markets include:

- **Commercial Greenhouses:** Large-scale operations seeking to optimize resource use and improve crop yields.
- **Urban Farmers:** Small-scale, urban agriculture enthusiasts aiming for efficient space and resource utilization.
- **Agricultural Enterprises:** Businesses focusing on sustainable farming practices and technological integration.

6.2.3 Value Proposition

The system offers:

- **Resource Efficiency:** Optimizes water and energy consumption, reducing operational costs.

- **Enhanced Crop Yield:** Provides data-driven insights for improved plant health and productivity.
- **Sustainability:** Promotes eco-friendly farming practices by minimizing waste and resource usage.

Implementing IoT in greenhouses leads to increased efficiency, improved crop yields, and reduced resource consumption.

6.2.4 Revenue Model

Potential revenue streams include:

- **Direct Sales:** Selling the IoT system hardware and software packages to greenhouse operators.
- **Subscription Services:** Offering data analytics and system monitoring services on a subscription basis.
- **Consulting and Support:** Providing setup assistance, training, and ongoing technical support for a fee.

6.2.5 Marketing And Sales Strategy

- **Partnerships:** Collaborate with agricultural equipment suppliers and greenhouse manufacturers to reach a broader audience.
- **Demonstrations:** Offer pilot programs and live demonstrations to showcase system effectiveness.
- Digital Marketing: Utilize online platforms, social media, and industry forums to promote the system's benefits.

Highlighting successful implementations, such as Green Leaf Farms' 30% water usage reduction, can effectively attract potential customers.

6.2.6 Competitive Analysis

Key differentiators of the system:

- **Integration of ML Algorithms:** Provides predictive analytics for proactive decision-making.
- **Comprehensive Monitoring:** Offers a wide range of environmental sensors for detailed data collection.
- **User-Friendly Interface:** Ensures ease of use with intuitive dashboards and controls.

Understanding the competitive landscape is crucial for positioning the product effectively in the market.

6.2.7 Financial Projections

- **Initial Investment:** Outline costs for research and development, manufacturing, and marketing.
- **Revenue Forecasts:** Provide projected sales figures based on market analysis and sales strategies.
- **Break-Even Analysis:** Estimate the time required to recover initial investments and achieve profitability.

While specific financial data is not provided here, conducting a detailed financial analysis is essential for attracting investors and guiding business decisions.

6.2.8 Regulatory and Compliance Considerations

Data Security: Implement robust measures to protect user data and comply with privacy regulations.

Agricultural Standards: Ensure the system meets industry standards for agricultural technology.

Environmental Regulations: Adhere to environmental laws and guidelines to promote sustainable practices.

Addressing these considerations builds trust with customers and ensures legal compliance.

6.2.9 Scalability and Future Development

Plans for future growth include:

Product Enhancements: Integrate advanced features like AI-driven disease detection and automated climate control.

Market Expansion: Introduce the system to international markets with tailored solutions for diverse agricultural practices.

User Community Building: Establish a community platform for users to share insights, feedback, and best practices.

Staying informed about emerging technologies and market trends is vital for continuous improvement and competitiveness.

6.3 Testing and Implantation

6.3.1 Implementation

There are several crucial steps involved in integrating an Internet of Things (IoT)-based environmental monitoring system with a machine learning (ML) model to forecast water needs in a tomato greenhouse. The function and integration of each component are described in detail in this part, which offers a thorough overview of the procedure.

6.3.1.1 IoT System Deployment

- **Sensor Installation**: A suite of sensors is strategically installed within the greenhouse to monitor key environmental parameters:
    - Soil Moisture Sensor: Measures the water content in the soil, providing data essential for irrigation decisions.
    - Temperature and Humidity Sensor (DHT11): Captures ambient temperature and humidity levels, influencing plant transpiration rates.
    - Light Intensity Sensor (LDR): Assesses the amount of light available, affecting photosynthesis.
    - Water Level Sensor: Monitors the water reservoir's status to ensure adequate supply for irrigation.
- **Microcontroller Integration:** All sensors are interfaced with an ESP32 microcontroller, which acts as the main hub. It is configured to gather sensor data on a regular basis and send it to a cloud-based platform for analysis and storage. The ESP32's integrated Wi-Fi feature makes data transfer smooth.

iot_irrigation_system.ino

```
24
25  // Pin definitions for sensors
26  #define DHTPIN 15
27  #define DHTTYPE DHT11
28  #define TRIG_PIN 27
29  #define ECHO_PIN 18
30  #define SIGNAL_PIN 34
31  #define LIGHT_SENSOR_PIN 33
32  #define RELAY_SWITCH 17
33
34  // OLED display parameters
35  #define SCREEN_WIDTH 128
36  #define SCREEN_HEIGHT 64
37  #define OLED_RESET -1
38  #define OLED_I2C_ADDRESS 0x3C
39
40  // Initialize DHT sensor and OLED
41  DHT dht(DHTPIN, DHTTYPE);
42  Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, OLED_RESET);
43
44  // Firebase objects
45  FirebaseData fbdo;
46  FirebaseAuth auth;
47  FirebaseConfig config;
48  bool signupOK = false;
49
50  // Variables for sensor data
51  unsigned long sendDataPrevMillis = 0;
52  int relayState = 0; // Variable to store relay status
53
54  void setup() {
55    Serial.begin(115200);
56
```

Figure 7 : Environmental reading IoT device Development code

- **Data Transmission and Storage:** Sensor data is transmitted to a cloud platform called Firebase where it is saved and accessible for both historical analysis and real-time monitoring. With this configuration, greenhouse conditions can be remotely monitored via online or mobile applications.

iot_irrigation_system.ino

```
    Serial.println(relayState);

    // Upload data to Firebase
    if (Firebase.ready() && signupOK && millis() - sendDataPrevMillis > 2000) {
      sendDataPrevMillis = millis();

      Firebase.RTDB.setFloat(&fbdo, "/sensors/temperature", temperature);
      Firebase.RTDB.setFloat(&fbdo, "/sensors/humidity", humidity);
      Firebase.RTDB.setFloat(&fbdo, "/sensors/distance", distance);
      Firebase.RTDB.setInt(&fbdo, "/sensors/soilMoisture", soilMoisture);
      Firebase.RTDB.setInt(&fbdo, "/sensors/lightLevel", lightLevel);

      if (fbdo.httpCode() != 200) {
        Serial.print("Firebase failed: ");
        Serial.println(fbdo.errorReason());
      } else {
        Serial.println("Data uploaded to Firebase");
      }

      // Read relay control value from Firebase
      if (Firebase.RTDB.getInt(&fbdo, "/sensors/watering_status")) {
        relayState = fbdo.intData(); // Update relayState variable
        if (relayState == 1) {
          digitalWrite(RELAY_SWITCH, HIGH); // Turn relay ON
          Serial.println("Relay turned ON");
        } else if (relayState == 0) {
          digitalWrite(RELAY_SWITCH, LOW); // Turn relay OFF
          Serial.println("Relay turned OFF");
        }
      } else {
        Serial.println("Failed to read watering_status from Firebase");
      }
    }
```

Output

Figure 8 : IoT Devise Firebase Integration to send Live sensor readings

The assembled device underwent testing as a part of multiple cycles when placed within a controlled greenhouse environment. The device maintained consistent performance due to the plastic casing which protected from water and dust interference. The available field testing results showed no system interruptions or sensor defects occurred.

Casing addition to the device enhanced its operational longevity while enabling dependable testing and monitoring operations vital to system behavior verification in agricultural conditions.

IoT Device Firmware and Automation Logic

The core firmware running on the ESP32 microcontroller plays a critical role in the system. It handles sensor readings, uploads environmental data to Firebase in real time, and controls connected actuators based on automation commands issued from the web application.

This firmware was written in Arduino C++ using libraries such as:

- DHT.h for temperature and humidity sensing,
- Adafruit_SSD1306.h for OLED display output,
- Firebase_ESP_Client.h for real-time database integration,
- Standard WiFi.h and GPIO control libraries.

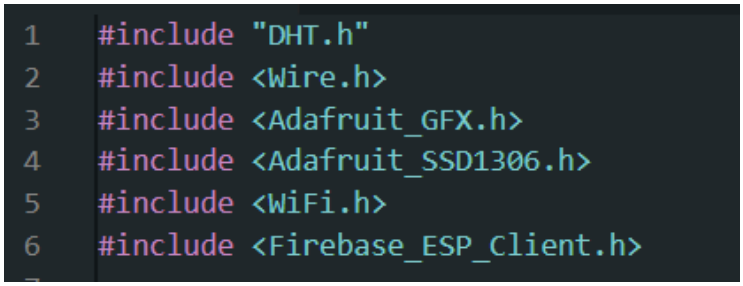


Figure 9: libraries used in this IoT project

A. Core Functionalities

1. Wi-Fi and Firebase Integration

The ESP32 connects to a local Wi-Fi network and authenticates with Firebase using secure API keys.

It continuously syncs both sensor values (upload) and actuator statuses (download) with the Firebase Realtime Database.

```
void connectWiFi() {
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to Wi-Fi");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(300);
  }
  Serial.println("\n✅ Connected to Wi-Fi");
}

void initFirebase() {
  config.api_key = API_KEY;
  config.database_url = DATABASE_URL;
  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);

  if (Firebase.signUp(&config, &auth, "", "")) {
    Serial.println("✅ Firebase Auth OK");
    signupOK = true;
  } else {
    Serial.printf("❌ Firebase Auth Failed: %s\n", config.signer.signupError.message.c_str());
  }
}
```

Figure 10: Wi-Fi and Firebase Integration

2. Sensor Readings

The following sensors are initialized and read:

- **DHT11** – Temperature and Humidity
- **Ultrasonic Sensor** – used to measure the Water tank water availability.
- **Analog Soil Moisture Sensor**
- **Analog Light Sensor**
- **Analog Water Level Sensor** – used to detect the irrigation water overflow in the plant pot's maximus level

Sensor data is collected every second and uploaded to Firebase under /sensors/.

```
115  void readSensors() {
116    temperature = dht.readTemperature();
117    humidity = dht.readHumidity();
118    distance = getUltrasonicDistance();
119    soilMoisture = analogRead(SIGNAL_PIN);
120    lightLevel = analogRead(LIGHT_SENSOR_PIN);
121    waterLevel = analogRead(WATER_LEVEL_PIN);
122  }
```

Figure 11: Sensor Reading Code snipped

3. OLED Display Output

A compact 128x64 OLED screen shows all live sensor readings and the current relay states of:

- Device 1 (Irrigation)
- Device 2 (Fan)

This provides immediate field-level feedback even without a browser interface.

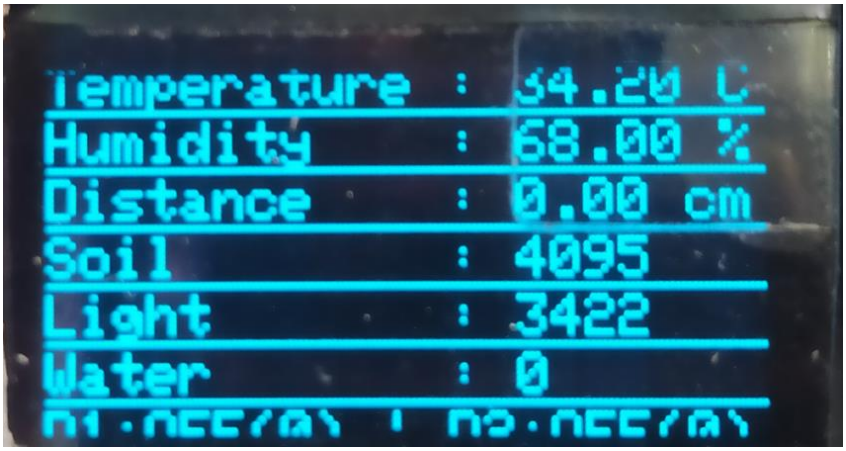


Figure 12: Device OLED Screen Outputs

B. Automation Features and Relay Control

The ESP32 receives automation commands and status values from Firebase, then performs actions via relay modules.

```
void syncRelayStates() {
  if (Firebase.ready() && signupOK) {
    if (Firebase.RTDB.getInt(&fbdo, "/sensors/device1_status") && fbdo.dataType() == "int") {
      device1State = fbdo.intData();
      digitalWrite(RELAY_DEVICE1, device1State == 0 ? HIGH : LOW);
    }
    if (Firebase.RTDB.getInt(&fbdo, "/sensors/device2_status") && fbdo.dataType() == "int") {
      device2State = fbdo.intData();
      digitalWrite(RELAY_DEVICE2, device2State == 0 ? HIGH : LOW);
    }
  }
}
```

Figure 13: Relay Control code snipped based on Firebase Update

1. Automated Irrigation Control

Controlled by: Firebase key **/sensors/device1_status**

Logic Executed in Backend (Python):

- If soil moisture > 2500 (dry),
- And water tank level > 20%,
- And plant water level < 1000,

  ✅ Then: /sensors/device1_status = 1 (Irrigation ON)

The ESP32 reads this value and sets Relay 1 (connected to the pump) LOW (active).

Novel Automation Feature: Overflow Prevention via Pot-Level Water level Sensor

Problem in Existing Systems:

- Traditional smart irrigation systems mainly rely on soil moisture sensors or time-based watering schedules. While these methods improve irrigation control, they have a critical limitation:

  They cannot detect plant pot overflow, which often leads to:

  - Water wastage
  - Nutrient leaching
  - Root rot and fungal growth in tomato crops

- This is a significant oversight in many automated systems deployed in greenhouses and gardens.

✅ **Proposed Enhancement in This System:**

To overcome this, a plant pot-level water sensor is added in this project as a novel feature.

Functionality:

- The system continuously reads the water level in the pot.
- During automated irrigation, if the pot water level crosses a defined safe threshold, the system:
- Immediately shuts off the water pump
- Displays an alert on the dashboard
- Optionally triggers the buzzer alarm
- This feature is controlled through logic in the web app backend (App.py) and is enforced through Firebase key /sensors/waterLevel.

Benefits:

✅ Prevents water overflow and plant stress

✅ Avoids root diseases in tomato crops

✅ Reduces water usage and improves efficiency

✅ Adds an intelligent feedback loop that most commercial systems lack

This feature is not found in most existing smart irrigation systems, making it a novel contribution of this project towards more sustainable and context-aware greenhouse automation.

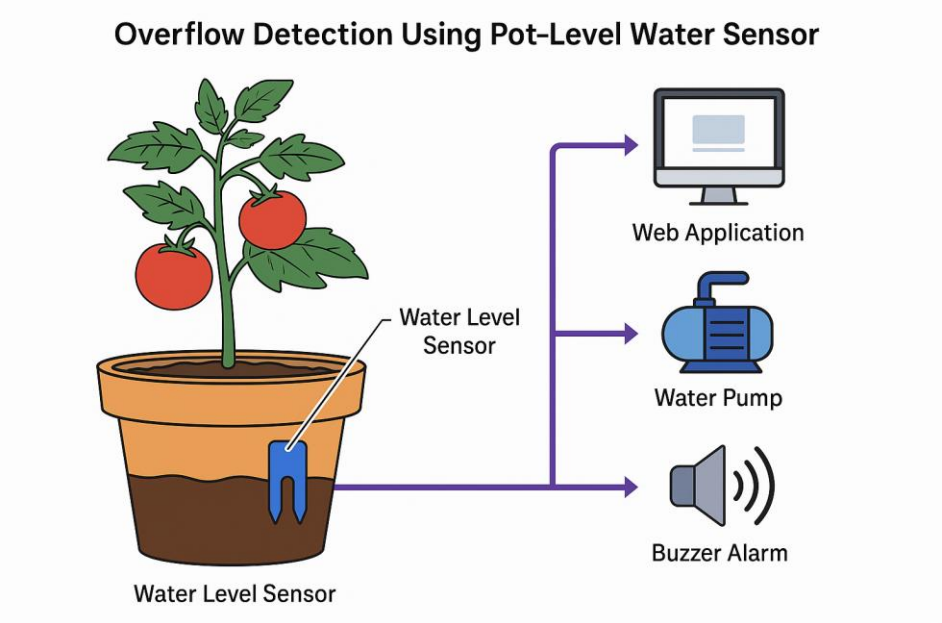


Figure 14: Illustration of Overflow Detection Using Pot-Level Water Sensor

```
#Irrigation Motor Automation Conditions
if automation_irrigation == 1:
    # Check if soil is dry, water tank and water level are sufficient
    if soil_moisture > 2500:
        if tank_percent > 20:
            if water_level < 1000:
                # Conditions all good - start irrigation
                sensor_ref.update({'device1_status': 1})
                alerts.append("Irrigation started: Soil moisture is high, and water is available.")
            else:
                # Water level too high - prevent overflow
                sensor_ref.update({'device1_status': 0})
                alerts.append("⚠️ Overflow detected: Irrigation stopped to prevent plant pot overflow.")
        else:
            # Water tank too low
            sensor_ref.update({'device1_status': 0})
            alerts.append(
                "🚫 Irrigation stopped: Water tank level is below 20%. Please refill the tank to continue irrigation.")
    else:
        # Soil moisture is sufficient, no irrigation needed
        sensor_ref.update({'device1_status': 0})
        alerts.append("ℹ️ Soil moisture is within optimal range. No need for irrigation at this time.")

else:
    # Automation off, turn off irrigation
    sensor_ref.update({'device1_status': 0})
```

Figure 15: Automated Irrigation Motor Control conditions

2. Fan Automation

Controlled by: Firebase key **/sensors/device2_status**

Trigger Conditions:

- If temperature > 28°C or humidity < 70%

  ✅ Then: /sensors/device2_status = 1 (Fan ON)

- The ESP32 activates Relay 2, turning the fan ON/OFF accordingly.

```
#Vemdilation Fan Automation Conditions
if fan_automation == 1:
    if humidity < 70:
        sensor_ref.update({'device2_status': 1})
        alerts.append("🌬️ Fan turned ON: Humidity is below 70%, improving air circulation.")
    elif humidity >= 70:
        sensor_ref.update({'device2_status': 0})
        alerts.append("✅ Fan turned OFF: Humidity is at or above 70%, no need for extra ventilation.")

    if temperature > 28:
        sensor_ref.update({'device2_status': 1})
        alerts.append("🌬️ Fan turned ON: Temperature is above 28°C, cooling in progress.")
    elif temperature <= 28:
        # Only turn off if humidity condition also says so to avoid conflict
        if humidity >= 70:
            sensor_ref.update({'device2_status': 0})
            alerts.append("✅ Fan turned OFF: Temperature is at or below 28°C, cooling not needed.")
else:
    sensor_ref.update({'device2_status': 0})
    alerts.append("⚠️ Fan automation is disabled; fan remains OFF.")
```

Figure 16: Fan Automation Conditions

3. Alarm Buzzer Automation

Controlled by: Firebase key **/sensors/buzzer_status**

Blinking pattern implemented: toggles every 100ms for buzzer effect.

Triggered remotely when:

- Tank is <10%
- Water overflow detected
- Temperature > 40°C
- Soil is dry AND irrigation automation is off

The buzzer will toggle ON/OFF repeatedly until the status is reset to 0.

```
#Alarm Buzzer Automation Conditions
buzzer_triggered = False
if automated_alarm == 1:
    if water_level > 1000:
        alerts.append("🚨 Water overflow detected! Buzzer activated.")
        buzzer_triggered = True
    if tank_percent < 10:
        alerts.append("⚠️ Water tank is below 10%. Buzzer activated.")
        buzzer_triggered = True
    if temperature > 40:
        alerts.append("🔥 High temperature! Buzzer activated.")
        buzzer_triggered = True
    if soil_moisture < 300 and automation_irrigation == 0:
        alerts.append("🌵 Soil is dry. Enable auto irrigation or water manually. Buzzer activated.")
        buzzer_triggered = True
else:
    sensor_ref.update({'buzzer_status': 0})

sensor_ref.update({'buzzer_status': 1 if buzzer_triggered else 0})
```

Figure 17: Alarm Buzzer Automation Control Conditions

C. Real-Time Synchronization Cycle

In the loop() function:

The device continuously:

- Checks Wi-Fi connection
- Reads all sensor values
- Updates OLED display
- Uploads sensor data every second
- Retrieves and updates relay/buzzer status from Firebase

This creates a real-time bidirectional loop between:

Physical environment → ESP32 → Firebase → Web App → Firebase → ESP32 → Actuators

D. Smart Failover Features

Reconnects to Wi-Fi if disconnected.

Authenticates with Firebase and handles retries.

Uses non-blocking logic to avoid delays (e.g., timer-based uploads and buzzer toggling).

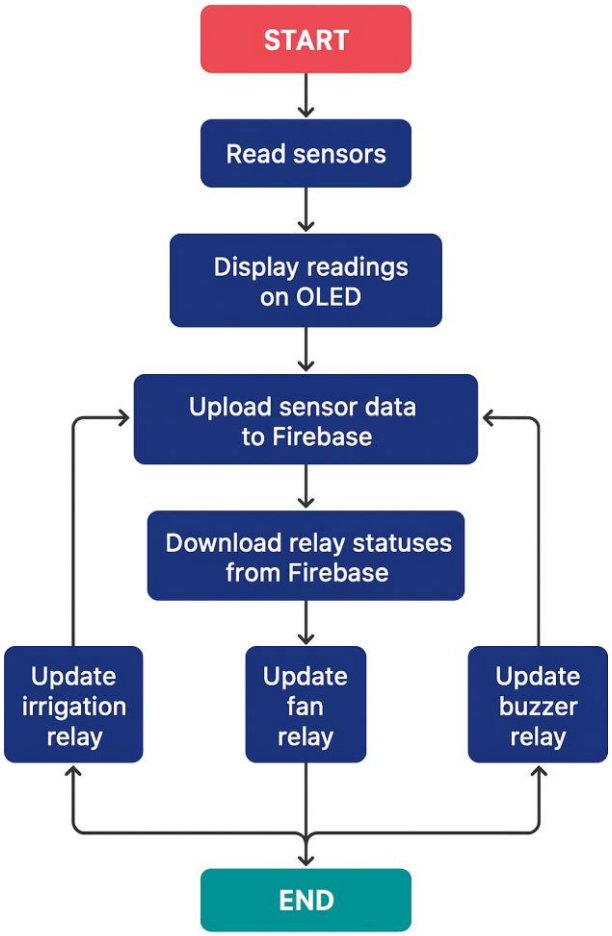


Figure 18 : Firmware Execution Flow from Sensor Input to Actuator Output via Firebase

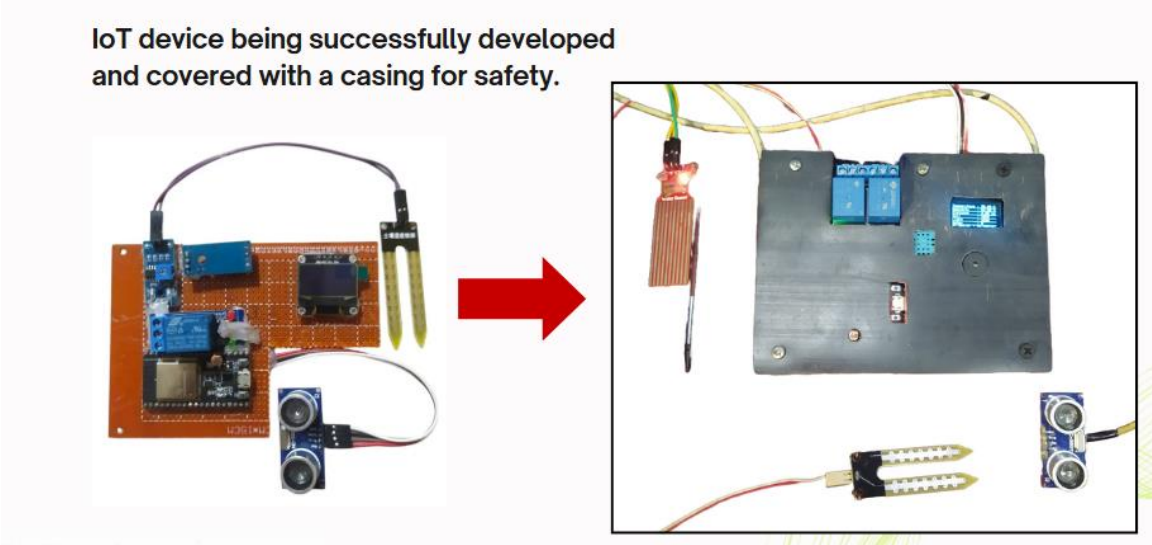


Figure 19: IoT device before and after casing for safety

6.3.1.2 Firebase Realtime Database Integration

To enable real-time data exchange between the IoT devices and the web application, this system uses Firebase Realtime Database as the cloud-based backend. Firebase allows live updates of sensor readings and automation controls with low latency, ensuring smooth and dynamic system operation.

What Data is Stored in Firebase?

The data is structured under two main nodes:

1. /sensors

This node holds all live environmental data and device statuses, including:

- temperature (°C)
- humidity (%)
- soilMoisture (analog value)
- lightLevel (lux)
- distance (cm from ultrasonic sensor in water tank)
- waterLevel (pot level reading)

- tank_height_cm (user-configurable tank height)
- predicted_water (output from ML model)
- irrigating_water, automation_irrigation, fan_automation, automated_alarm
- device1_status and device2_status (relay control for pump and fan)
- buzzer_status (for sound alert control)

This enables both the ESP32 and the web application to stay in sync.

2. /prediction_logs

This node stores historical prediction data for later viewing and export. It logs:

- Timestamp
- Sensor inputs
- Plant count
- Predicted water (per plant)
- Total water requirement

How It Works

- The ESP32 device continuously reads sensor data and uploads it to Firebase.
- The Flask web application reads this data for:
- Auto-filling the prediction form
    - Updating the dashboard
    - Deciding whether to turn ON/OFF the pump, fan, or alarm (automation logic)
- Users can also send updates from the UI (e.g., toggle irrigation ON), which Firebase then relays to the device.
- This real-time interaction creates a fully integrated IoT-ML-Dashboard system.

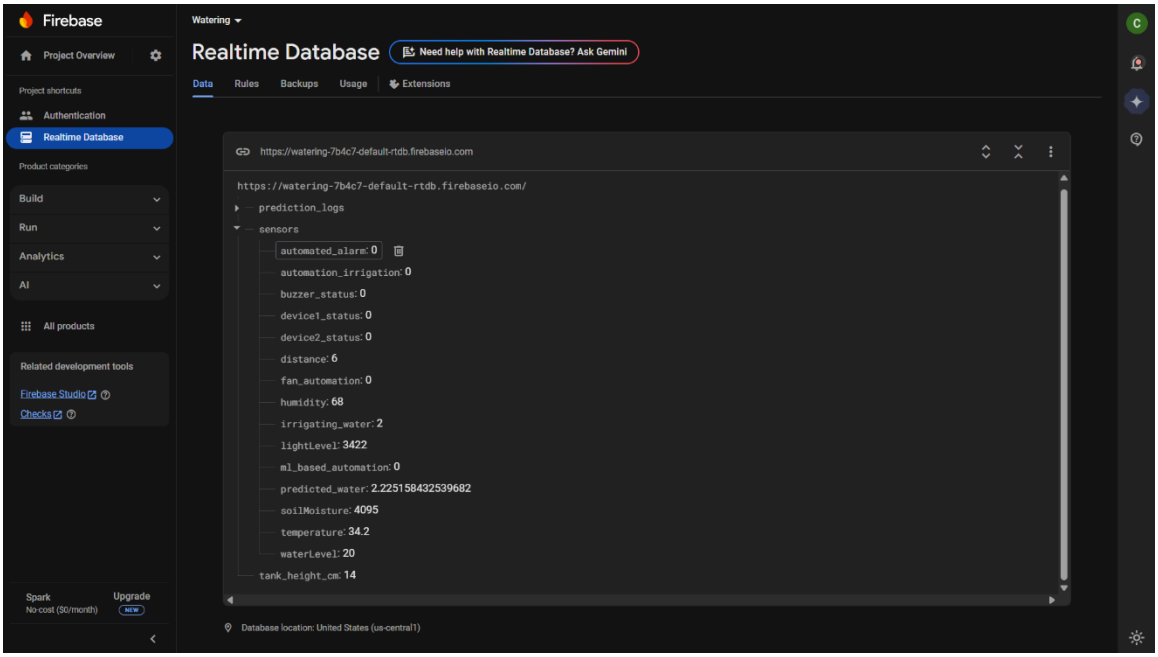


Figure 20: Firebase Realtime DB showing live sensor values, automation settings, and device control status

6.3.1.3 Machine Learning Model Development

- **Data Collection**: The cloud platform aggregates data from all sensors, creating a comprehensive dataset that reflects the greenhouse's environmental conditions over time.
- **Data Preprocessing:** When data is collected, it goes through preprocessing procedures such as addressing missing values, standardizing data ranges, and, if required, encoding categorical variables. This guarantees that the dataset is clean and appropriate for ML model training.
- **Model Selection and Training:** To avoid overfitting and handle nonlinear interactions with resilience, a Random Forest Regressor is chosen. Using the preprocessed information, the model is trained to forecast the ideal water needs

depending on input characteristics including light intensity, temperature, humidity, and soil moisture.

- **Model Validation and Testing:** To evaluate the predicted accuracy of the trained model, a different subset of data is used for validation. To assess the model's efficacy, performance metrics such as Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are computed.

Timestamp	Temperature (Â°C)	Humidity (%)	Soil Moisture (%)	Light Level (lux)	Water Level Needed (Liters)
11/1/2024 8:00	27.5	69.6	10.5	903	0.6
11/1/2024 9:00	39	69.5	36.5	1696	1.4
11/1/2024 10:00	34.6	34.6	17.1	966	0.8625
11/1/2024 11:00	32	54.7	48.4	1183	1.6125
11/1/2024 11:00	23.1	32.9	15.9	517	0.6875
11/1/2024 13:00	23.1	57.5	26.6	609	0.95
11/1/2024 14:00	21.2	52.1	13.4	1089	0.6
11/1/2024 15:00	37.3	74.4	49.9	1220	1.7125
11/1/2024 16:00	32	47.5	30.1	1400	1.15
11/1/2024 17:00	34.2	35.9	33.8	937	1.275
11/1/2024 18:00	20.4	37.1	12.7	1542	0.575
11/1/2024 19:00	39.4	68.1	40	1790	1.4875
11/1/2024 20:00	36.6	60.9	18.4	1670	0.9125
11/1/2024 21:00	24.2	35.1	45.9	559	1.45
11/1/2024 22:00	23.6	34.2	18.2	1221	0.75
11/1/2024 23:00	23.7	65	17.6	657	0.7375
11/2/2024 0:00	26.1	33.6	11.5	863	0.6125
11/2/2024 1:00	30.5	71.1	28.9	1980	1.1
11/2/2024 2:00	28.6	65.3	32.6	714	1.175
11/2/2024 3:00	25.8	34.1	12.6	1248	0.6375
11/2/2024 4:00	32.2	34.2	41	1427	1.425
11/2/2024 5:00	22.8	79.3	28.1	1554	0.9875
11/2/2024 6:00	25.8	48.7	31	1339	1.1
11/2/2024 7:00	27.3	48.5	27.6	515	1.025

Figure 21 : Dataset for Tomato Greenhouse Environmental and Irrigation Prediction

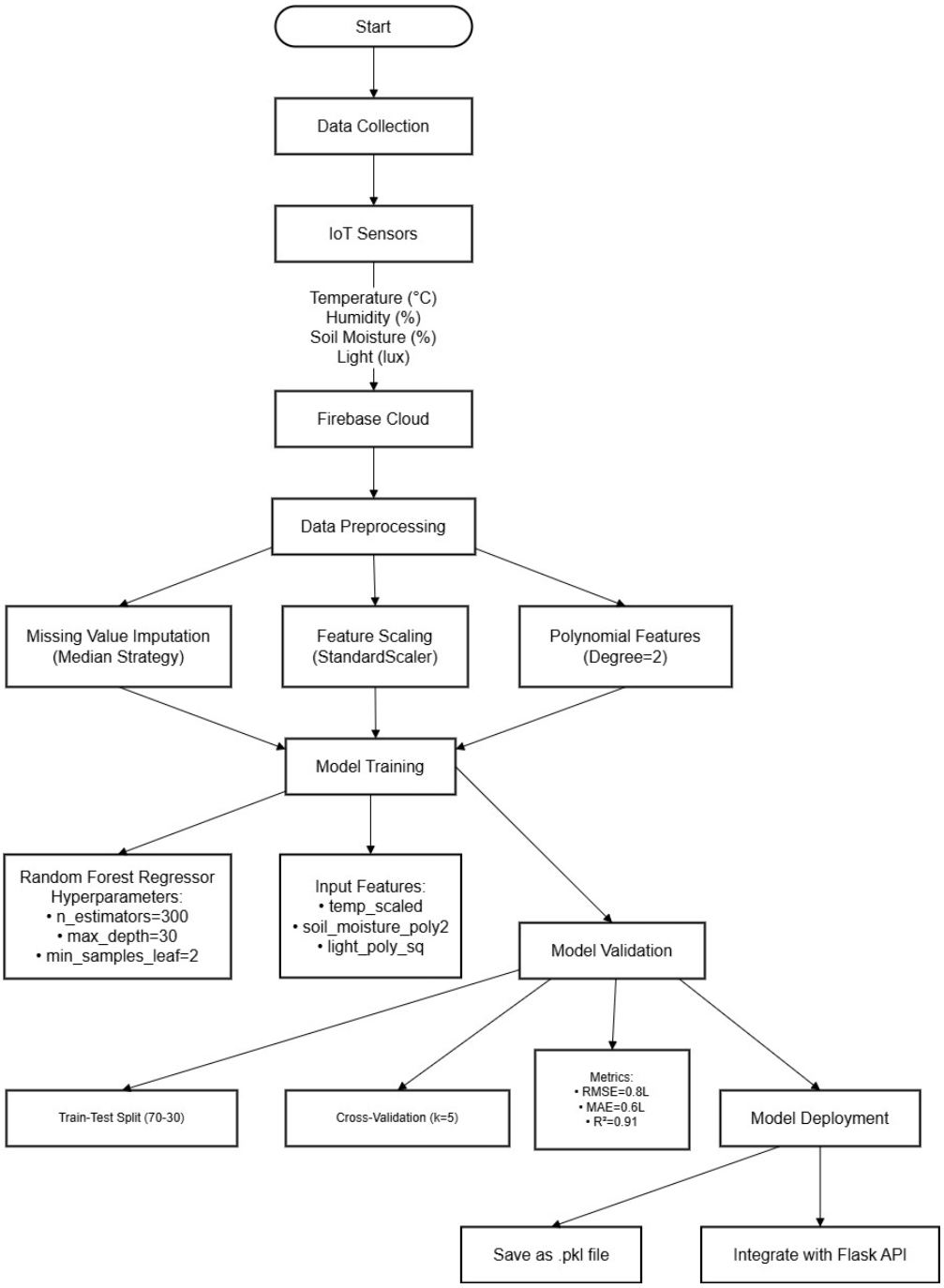


Figure 22: Flowchart for Random Forest model training and prediction process

6.3.1.4 Integration and Dashboard Development

- **System Integration:** The cloud platform receives real-time sensor data from the ESP32 microcontroller, which is configured to process it using a trained machine learning model to forecast water needs. Data-driven, automated watering suggestions are made possible by this integration.
- **Dashboard Creation:** To give customers an easy-to-use interface that shows real-time sensor readings and water requirement estimations, a web-based dashboard is created. To help with well-informed decision-making, the dashboard incorporates visuals like graphs and charts that show data trends over time.
- **Web Application Implementation:** The web-based interface allows users (farmers or greenhouse operators) to monitor environmental conditions, visualize predictions, and control actuators (e.g., turning the pump ON/OFF based on water level or schedule). The frontend was developed using HTML, CSS, Bootstrap, while the backend used Python , Flask Framworks.

System User Interfaces and Functional Modules

Introduction

The web application acts as the control center for the Smart Greenhouse Decision Support System. Designed with a user-first approach, it is optimized for greenhouse operators, farmers, and even non-technical users, providing them with real-time monitoring, ML-based water predictions, and automated environmental control via a simple dashboard.

The system follows a modular layout, where each interface performs a clear function with immediate feedback, intuitive controls, and automation insights. It is developed using:

- **Flask (Python)** – for backend integration
- **Firebase Realtime DB** – for live data sync
- **Bootstrap & Chart.js** – for responsive, elegant design and data visualization
- **JavaScript** – for dynamic interactions and AJAX-based updates

A. Login and Authentication Panel

- Supports **secure login and registration** using **Firebase Authentication**.
- Stores user sessions and prevents unauthorized access to critical control panels.
- Simple, elegant UI designed for mobile and desktop.

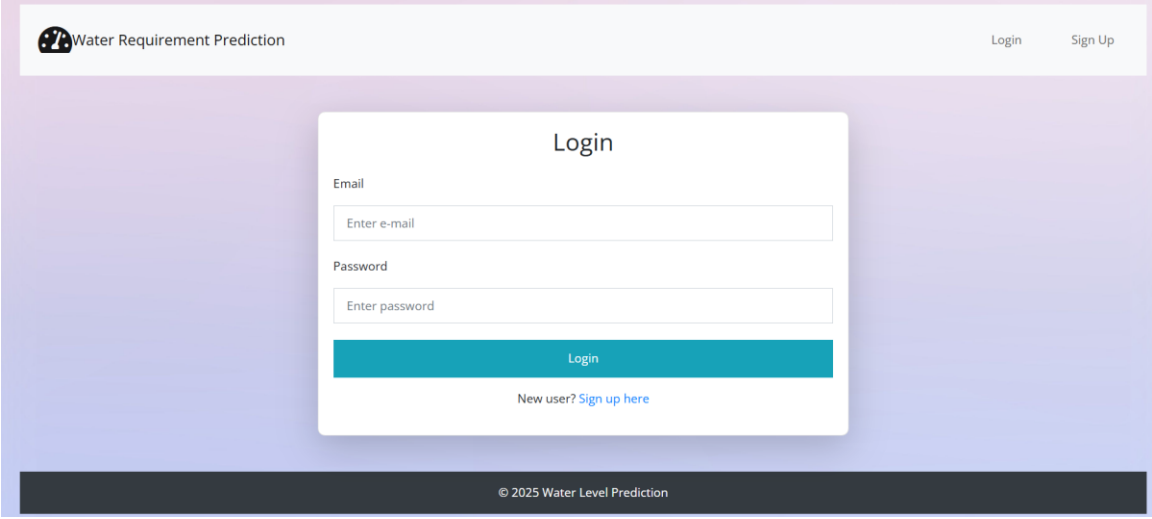


Figure 23: Login Interface

B. Real-Time Sensor Dashboard

This is the main monitoring panel where live sensor data from the greenhouse is displayed as colored **status cards**, including:

- **Water Distance** (cm) – from ultrasonic sensor
- **Water Level** – in cm or raw analog value from pot
- **Humidity (%)**
- **Light Intensity (lux)**
- **Soil Moisture** – raw analog value

- **Temperature (°C)**

Water Tank Visualization Panel

- A vertical **animated tank UI** visually shows the current fill level.
- Displays % fill, water height in cm, and badge indicators:
    - ✅ Good (above 50%)
    - ⚠️ Low (30–50%)
    - 🚨 Critical (<30%)
- **Tank height is configurable** using a numeric input.

Automation Control Switches

- Users can manually toggle:
    - **Automated Irrigation** – based on soil and tank conditions
    - **Fan Automation** – based on temperature/humidity
    - **Alarm Buzzer** – for overflow, high temp, or dry soil
- Current state shown with clear ON/OFF labels and dynamic alerts shown below.

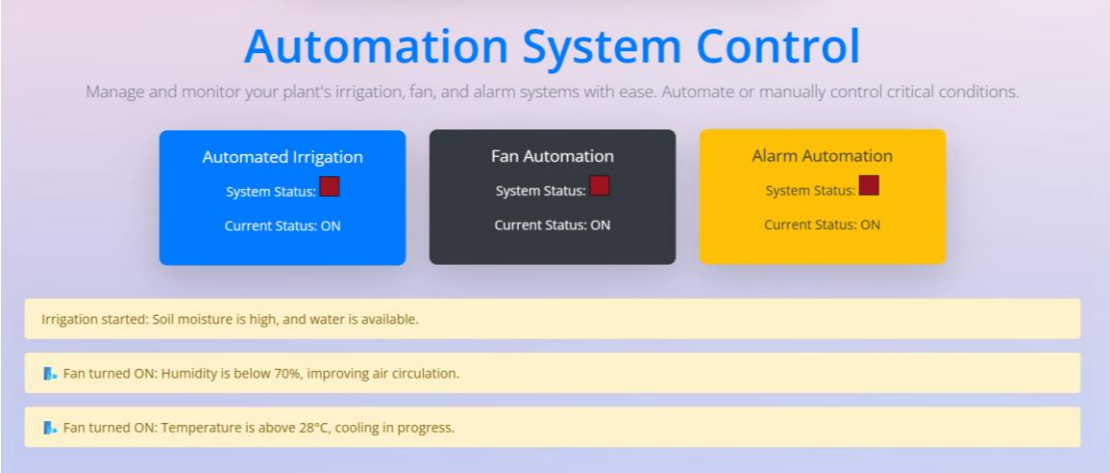


Figure 24: Automation Control Interface with triggering Alert messages

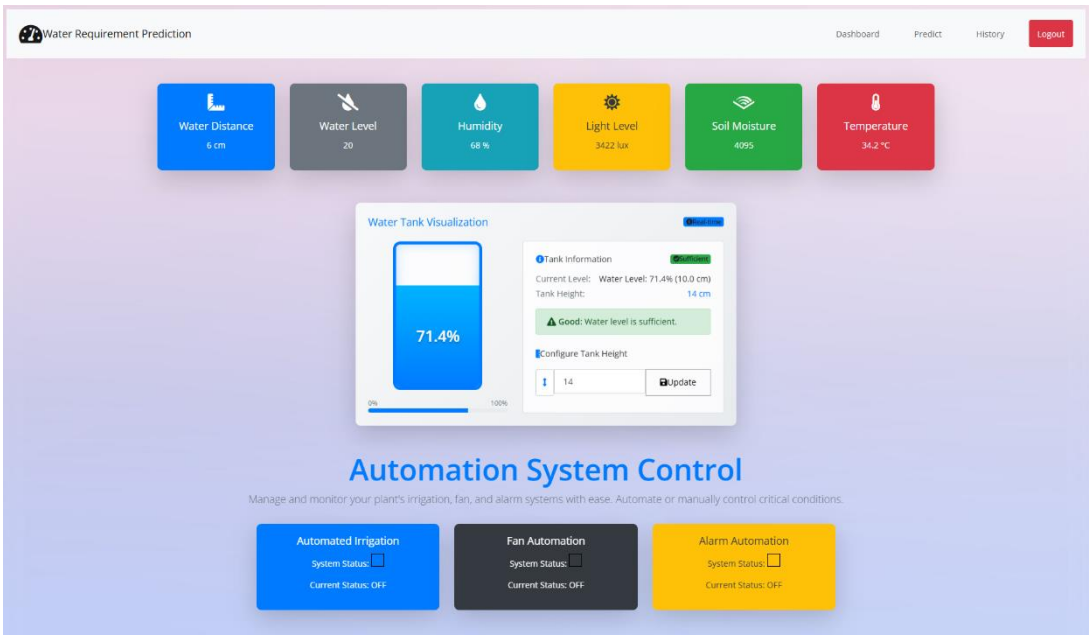


Figure 25: Dashboard with Live Cards, Tank Visualization, and Automation Controls

C. Irrigation Water Requirement Prediction Interface

This panel allows the user to **generate data-driven water predictions** using real-time or manually entered environmental values.

Functional Features

- **Real-time integration with Firebase** allows the form to **auto-fill sensor data** such as:
    - Temperature
    - Humidity
    - Soil Moisture
    - Light Intensity

These fields are populated **automatically from live greenhouse sensor readings** retrieved through the **/sensors** path in the Firebase Realtime Database. This removes manual entry errors and enhances decision accuracy.

- The user can manually override any value for testing or simulation.
- **Two methods** are provided to define the number of plants:
    - **Direct Count**: Input the total number of tomato plants manually.
    - **Area-Based**: Calculate plant count based on greenhouse area and plant spacing (system calculates automatically).
- On submission, the system:
    - Sends the inputs to the backend.
    - Applies preprocessing, scaling, and polynomial transformations.
    - Feeds the values into the Random Forest ML model.
    - Displays:
        - **Predicted water requirement per plant (in Liters)**
        - **Total water required** for the given number of plants.

Prediction Reasoning System

The UI also displays a **contextual explanation** for the prediction based on the entered (or live) environmental conditions. Examples include:

- "Alert: Elevated temperature detected. Consider providing shade and increasing watering frequency."
- "Warning: Low soil moisture detected. Increase irrigation to maintain plant health."

This helps non-technical users **understand the reasoning** behind the ML output.

Plant Count Calculation Modes in Prediction Interface

The Water Requirement Prediction UI supports **two flexible methods** for calculating the number of plants in a greenhouse setup, catering to both precise manual control and dynamic area-based estimation.

This design enables users to easily calculate irrigation requirements whether they know the **exact number of plants** or just the **area of the greenhouse**.

Auto-Adaptive Form UI Logic

The UI dynamically switches between the two modes using **radio button logic**:

- Selecting "Direct Count" shows only the input field for total plants.
- Selecting "Area-Based" reveals input fields for area and spacing.

This switch is handled using JavaScript and Bootstrap styles to ensure a smooth user experience without page reloads.

A. Direct Plant Count Method

- The user manually inputs the total number of tomato plants (e.g., 400).
- This value is multiplied by the predicted **liters per plant** to determine the **total water requirement**.
- Ideal for **small or fixed setups** where the number of plants is constant and known.

💡 **Example:**

- Plant count: 444
- Water per plant: 1.64 Liters
- Total required: 728.66 Liters

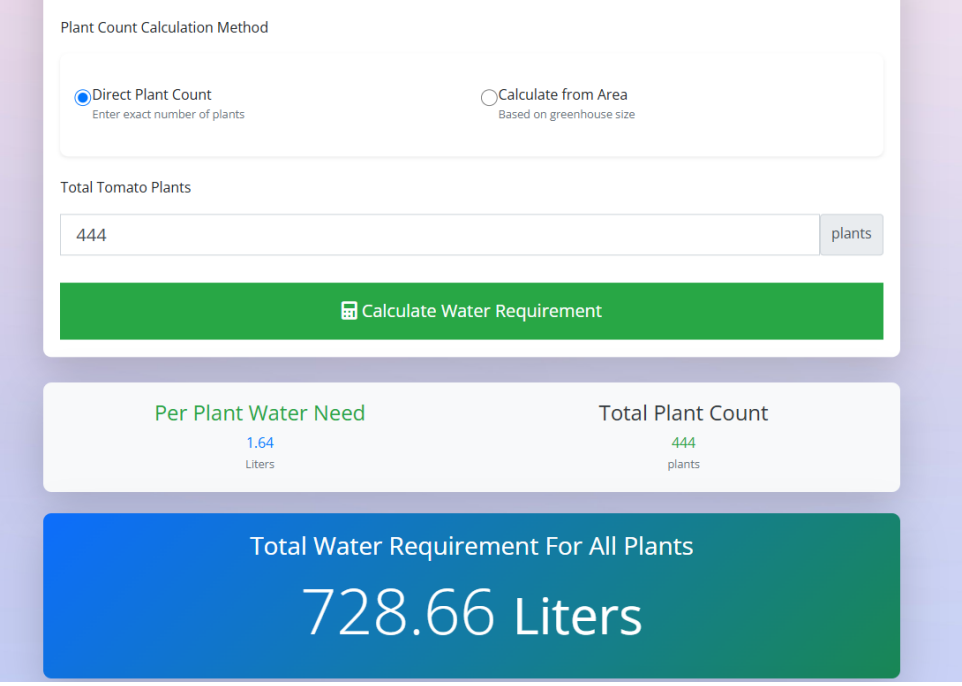


Figure 26: Sample Calculation output: Direct Plant Count Logic

B. Area-Based Estimation Method

- Designed for **dynamic or larger greenhouses**.
- The user enters:
    - Greenhouse area (m^2)

- Plant spacing or Plant Gap (m): 1m, 1.5m, 2m

Figure 27: Image showing the Plant Gap (E.g.: Gap = 1.5m)

- The system automatically calculates the Total Plant Count based on the Area and the Plant Gap
- This estimated count is then used for irrigation prediction.

Sample Calculation – Area (m^2) Based Plant Count

To help users who do not know the exact number of plants, the system provides a method to **automatically calculate plant count** using the greenhouse area and plant spacing (Plant Gap). This helps ensure accurate water requirement predictions.

Example: when,

- **Greenhouse Area** = 1000 m^2
- **Plant Gap** = 1.5 m

The system uses the formula:

$$\text{Plant Count} = \left\lfloor \frac{\text{Area}}{\text{Gap}^2} \right\rfloor = \left\lfloor \frac{1000}{1.5 \times 1.5} \right\rfloor = \left\lfloor \frac{1000}{2.25} \right\rfloor = 444 \text{ plants}$$

So, for a 1000 m² greenhouse with 1.5 m spacing between plants, the system estimates **444 tomato plants**.

This estimated count is then used to calculate total water needs based on the predicted water per plant from the ML model.

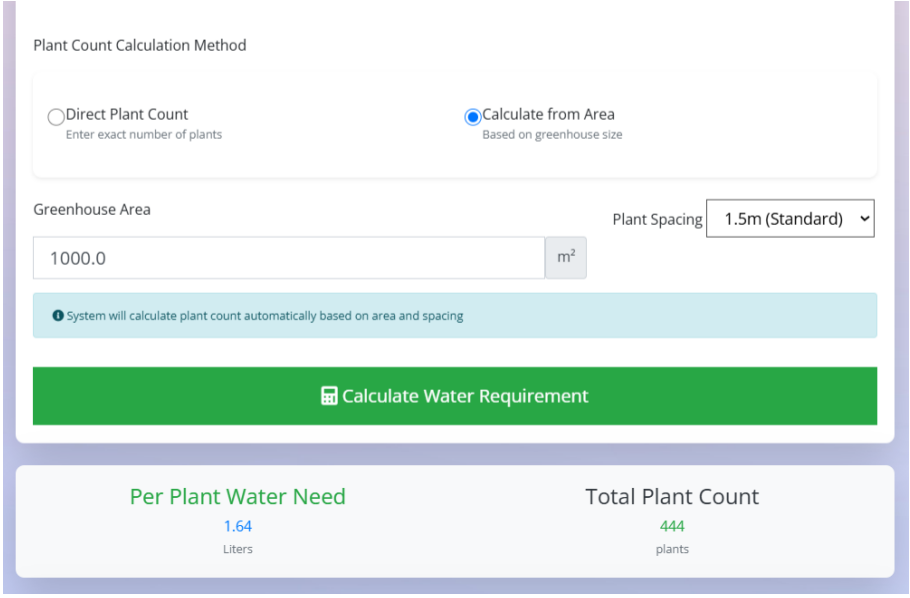


Figure 28: Sample Calculation output: Area-Based Plant Count Logic

```
count_method = request.form.get("count_method", "direct")

if count_method == "area":
    greenhouse_area = float(request.form["greenhouse_area"])
    plant_gap = float(request.form["plant_gap"])
    plant_count = math.floor(greenhouse_area / (plant_gap ** 2))
else:
    plant_count = int(request.form["plant_count"])
```

Figure 29: Plant Count Calculation function using Area method

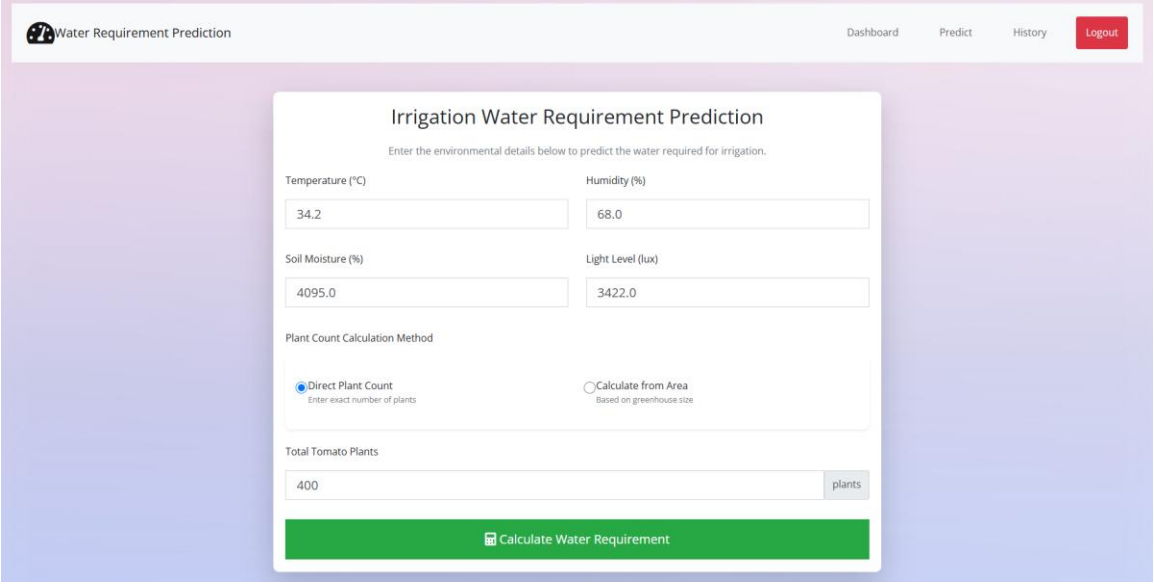


Figure 30: Water Requirement Prediction Interface with Live Sensor Autofill

Figure 31: Prediction Form with Output and Reasoning

📈 D. History and Analytics Visualization

A detailed analytics dashboard that provides historical tracking and logging.

Features:

- **Line graph** (Chart.js): shows trends of water requirement predictions over time.
- **Data table**: lists past predictions with input values and calculated outputs.
- Fully sortable and paginated for large datasets.
- **Download button**: exports logs to **CSV** for external documentation or research.

Figure 32: Historical Prediction Graph and Log Table

Summary of User Interface Capabilities

Module	Description	Unique Benefit
Login	User authentication	Secures system access
Dashboard	Live sensor data view	Centralized monitoring
Tank Visual	Tank water level with status badge	Prevents under- or over-filling
Prediction Panel	Calculates per-plant and total irrigation	Data-driven water use
Automation Panel	Toggle irrigation, fan, alarm	Customizable control
History Panel	Charts + log table + CSV download	Traceability and planning

This interface design ensures that users can make **fast, informed decisions** without any technical expertise, while still having full control over the greenhouse environment.

Let me know if you'd like this exported as a **Word file** or if you'd like a **combined UI diagram** summarizing all modules in a single infographic.

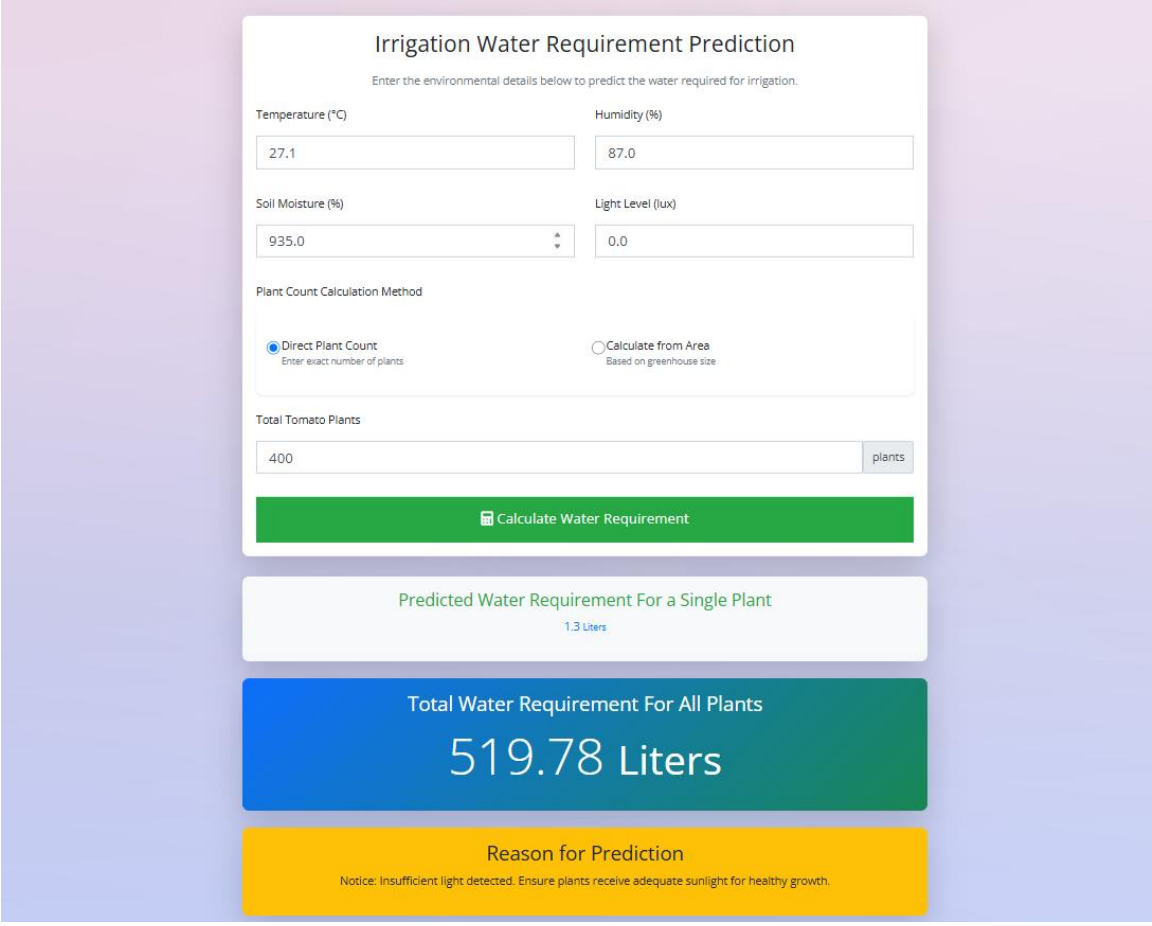


Figure 33 : UI for water requirement prediction

Through the use of predictive analytics and real-time environmental monitoring, this implementation methodology guarantees effective water management in the tomato greenhouse, optimizing resource consumption and raising crop yield.

6.3.1.5 Data Preprocessing:

The irrigation dataset collected from the IoT device was initially raw and contained missing values, as well as unnecessary columns such as timestamps. The first step involved dropping the Timestamp column entirely to focus on the primary environmental indicators:

- Temperature (°C)

- Humidity (%)
- Soil Moisture (%)
- Light Level (lux)

To handle missing values, the **median imputation** strategy was employed using SimpleImputer from sklearn. This ensured robustness against outliers that could skew the mean, making median imputation ideal for real-time agricultural data.

The flow of the preprocessing is illustrated in Fig and the implementation codes are mentioned below

```
#Load dataset
file_path = "irrigation_dataset.csv"
data = pd.read_csv(file_path)

# Drop timestamp completely
data = data.drop(columns=['Timestamp'])

# Handle missing values
imputer = SimpleImputer(strategy='median')
features = ['Temperature (°C)', 'Humidity (%)', 'Soil Moisture (%)', 'Light Level (lux)']
data[features] = imputer.fit_transform(data[features])

# Normalization and Polynomial Features
scaler = StandardScaler()
data[features] = scaler.fit_transform(data[features])
poly = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly.fit_transform(data[features])

# Define target variable
X = X_poly
y = data['Water Level Needed (Liters)']

# Train-test split
X_train, X_temp, y_train, y_temp = train_test_split( *arrays: X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split( *arrays: X_temp, y_temp, test_size=0.5, random_state=42)
```

Figure 34 : Datasets Preprocessing and cleaning

6.3.1.6 Feature Scaling and Data Augmentation

All input features were **normalized using StandardScaler** to ensure consistent scaling across different sensor values, improving the performance and convergence of the machine learning model.

Following normalization, **polynomial feature expansion** (degree = 2) was applied using PolynomialFeatures, which allowed the model to capture non-linear interactions between environmental factors. This is especially useful in agriculture where multiple environmental variables jointly impact irrigation needs.

```
24  # Normalization and Polynomial Features
25  scaler = StandardScaler()
26  data[features] = scaler.fit_transform(data[features])
27  poly = PolynomialFeatures(degree=2, include_bias=False)
28  X_poly = poly.fit_transform(data[features])
29
```

Figure 35 : StandardScaler and PolynomialFeatures implementation

6.3.1.7 Model Selection And Training

For prediction, a **Random Forest Regressor** was chosen due to its strong performance, robustness to noise, and ability to model complex, non-linear relationships.

To optimize the model, **GridSearchCV** was utilized to explore a wide range of hyperparameters:

- Number of estimators (trees)
- Maximum tree depth
- Minimum samples for split/leaf
- Bootstrap options

A 5-fold cross-validation strategy was used to prevent overfitting and to ensure generalization.

```
# Train-test split
X_train, X_temp, y_train, y_temp = train_test_split( *arrays: X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split( *arrays: X_temp, y_temp, test_size=0.5, random_state=42)

# Hyperparameter tuning
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

grid_search = GridSearchCV(estimator=RandomForestRegressor(random_state=42),
                           param_grid=param_grid,
                           cv=5,
                           n_jobs=-1,
                           verbose=2,
                           scoring='neg_mean_squared_error')
```

Figure 36 : GridSearchCV hyperparameter tuning

After hyperparameter tuning, the best model was selected and serialized using joblib along with the fitted scaler and polynomial transformer for future predictions:

```
# Train the best model
best_model = grid_search.best_estimator_
joblib.dump(best_model, filename: "irrigation_model_best.pkl")
joblib.dump(scaler, filename: "scaler_best.pkl")
joblib.dump(poly, filename: "poly_best.pkl")
print("Best model, scaler, and poly transformer saved successfully.")
```

Figure 37 : Code showing model saving process

6.3.1.8 Model Evaluation

The model's performance was evaluated using key regression metrics on the validation set:

- **Root Mean Squared Error (RMSE):** Measures the model's prediction error magnitude.
- **R^2 Score:** Indicates how well the model explains variability in the dataset.
- **Mean Absolute Error (MAE):** Represents average prediction deviation.

The trained model achieved satisfactory performance, indicating its suitability for predicting irrigation needs accurately based on real-time sensor data.

```
# Evaluate the best model on validation set
y_val_pred = best_model.predict(X_val)
rmse_val = np.sqrt(mean_squared_error(y_val, y_val_pred))
r2_val = r2_score(y_val, y_val_pred)
mae_val = mean_absolute_error(y_val, y_val_pred)

print("\nValidation Set Performance:")
print(f"RMSE: {rmse_val}")
print(f"R² Score: {r2_val}")
print(f"MAE: {mae_val}")
```

Figure 38 : Console output displaying RMSE, R^2, and MAE.

6.3.1.9 Integration with Web Application

The trained model (irrigation_model_best.pkl) was integrated into the web-based dashboard using Flask in the backend. The frontend, built using HTML, JS, CSS, Bootstrap allows greenhouse operators to input current environmental parameters or view predictions directly from real-time sensor feeds.

Once data is submitted, the backend:

1. Loads the serialized scaler and polynomial transformer.
2. Transforms input data accordingly.
3. Predicts the water requirement using the trained Random Forest model.
4. Sends results back to the frontend for display.

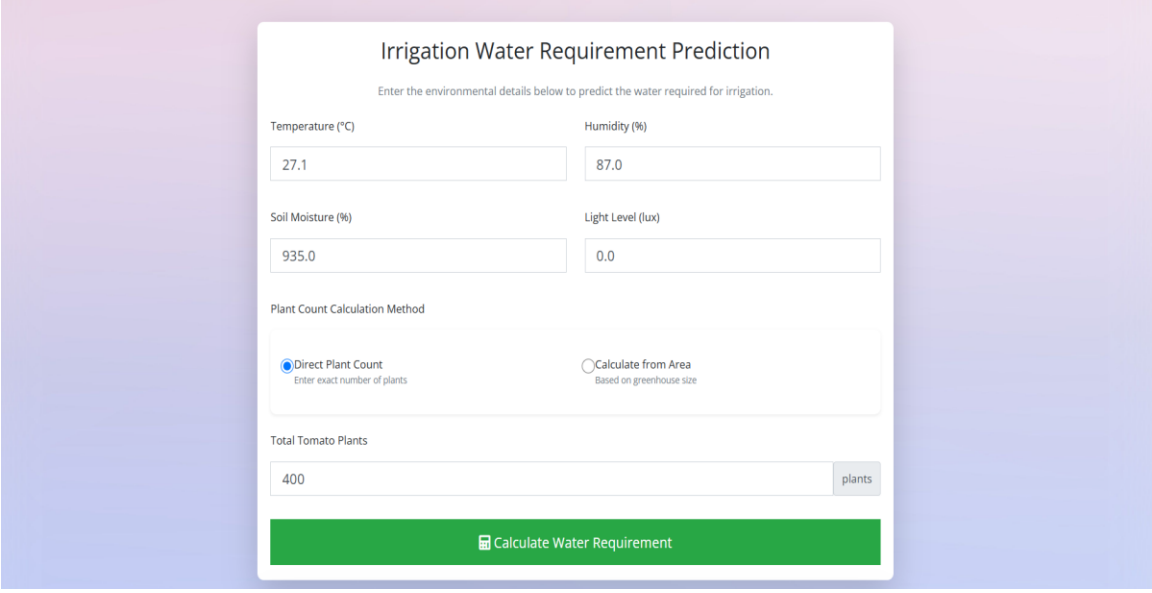


Figure 39 : Web interface showing input fields

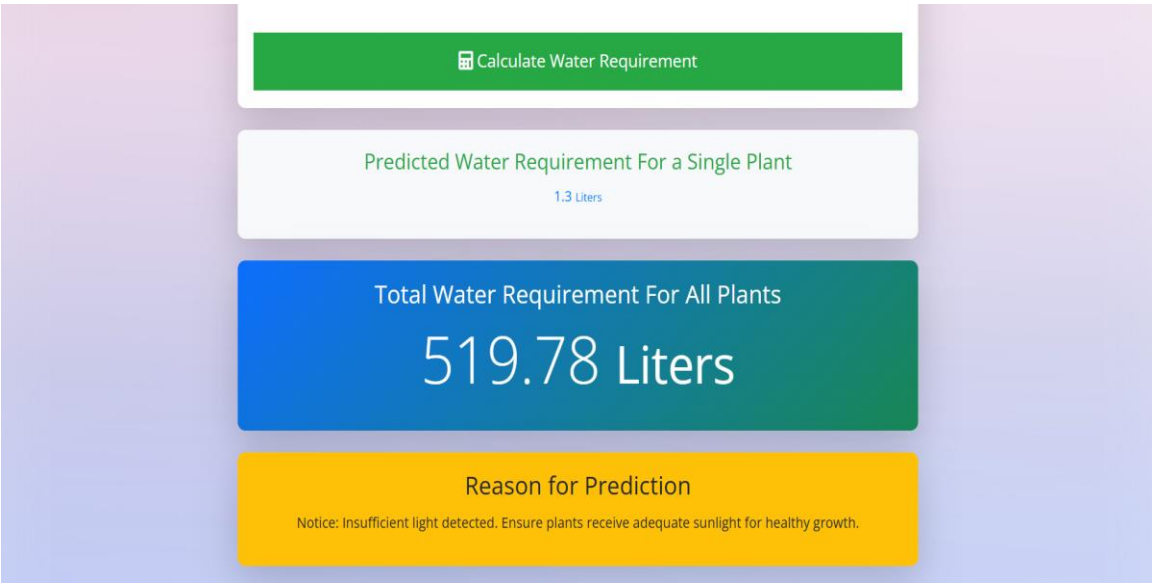


Figure 40 : Web interface showing predicted output.

6.3.2 Testing

Testing is a crucial step in guaranteeing the system's accuracy, dependability, and usefulness. Before being deployed, the system is put through a rigorous testing process to find any potential defects or issues. To confirm the functionality of both individual components and the system, the testing phase consists of unit, integration, and system testing.

6.3.2.1 Test Plan and Test Strategy

The primary goal of testing the Smart Greenhouse Decision Support System is to ensure that all components (IoT devices, web application, and machine learning model) work seamlessly together to provide accurate predictions for irrigation. Our strategy focuses on:

1. **Functional Testing:** Verifying the correct functionality of the IoT device and web application.
2. **Performance Testing:** Ensuring that the system provides quick and accurate predictions under different environmental conditions.
3. **Regression Testing:** Ensuring that new changes or improvements do not affect the existing features of the system.
4. **User Acceptance Testing (UAT):** Verifying that the system meets the requirements and expectations of the end users (greenhouse operators).

6.3.2.2 Test Case Design (Manual Test Reports)

Test cases are designed to validate each component and feature of the system. Below are 5 example test cases that were executed manually:

Table 2 : Test Case 01: Verify sensor data is stored in Firebase Realtime Database

Test Case Id	01
Test Scenario	Verify sensor data is stored in Firebase Realtime Database
Precondition	ESP32 is powered ON and connected to Wi-Fi
Input	Simulated sensor values (Temp=28°C, Humidity=65%, Soil=42%) sent via ESP32
Expected Output	Data appears in Firebase within 3 seconds with HTTP 200 status
Actual Result	Data appeared in `/sensors` node after 2.8s (Status: 200)
Status (Pass/Fail)	Pass

Table 3 : Test Case 02: Water Prediction Accuracy

Test Case Id	02
Test Scenario	Validate ML model prediction against manual irrigation records
Precondition	Model trained with $R^2 > 0.90$
Input	Test dataset (Temp=32°C, Humidity=55%, Soil=30%, Light=8500 lux)
Expected Output	Predicted water = 1L ±0.2L per plant
Actual Result	Model predicted 1.19 Liters (within tolerance)
Status (Pass/Fail)	Pass

Table 4 : Test Case 03: Plant Count Calculation

Test Case Id	03
Test Scenario	Verify area-based plant count calculation
Precondition	Greenhouse area = 100m², Plant gap = 1.5m

Input	Select ‘Calculate from Area’ method and enter 100m², and the Plant gap = 1.5m
Expected Output	System computes ≈44 plants (100/1.5²)
Actual Result	Calculated 44 plants
Status (Pass/Fail)	Pass

Table 5 : Test Case 04: Alert Thresholds

Test Case Id	04
Test Scenario	Check temperature alert triggers at >35°C
Precondition	Web Application Dashboard is running
Input	Simulated temperature input = 36°C
Expected Output	UI shows "Critical Warning: High temperature detected
Actual Result	Alert appeared with red highlight
Status (Pass/Fail)	Pass

Table 6 : Test Case 05: Watering Threshold Prediction

Test Case Id	05
Test Scenario	Check if watering prediction triggers correctly when temperature is above 30°C
Precondition	Dashboard is running and connected to Firebase for real-time data
Input	Simulated temperature input = 32°C, Humidity = 60%, Soil Moisture = 45%, Light = 300 lux, Total Plants = 400
Expected Output	The system should calculate the water requirement based on the inputs and display the predicted watering amount on the dashboard
Actual Result	The system predicted 575.37 liters of water required for the total plants
Status (Pass/Fail)	Pass

6.3.2.3 Testing Methods

The system was tested using a combination of unit testing, integration testing, and system testing to ensure that each part of the system functions correctly individually and when integrated:

1. **Unit Testing:**
    - Focuses on testing individual functions and components in isolation.
    - Unit tests were written to test the individual functions used for data preprocessing, model training, and prediction.
2. **Integration Testing:**
    - Verifies that the different components (IoT device, web application, and machine learning model) work together as expected.
    - For example, the integration between the web application and backend system, where data inputs are processed, transformed, and predictions are generated.
3. **System Testing:**
    - Tests the entire system as a whole, including IoT data collection, web application, machine learning model predictions, and system performance.
    - This phase ensures that all system components integrate well and the system meets the expected performance requirements.

6.3.2.4 Regression Testing

Regression testing was performed after each significant change to the system to ensure that the new code did not introduce any issues or affect existing functionality. This was done by running the previously mentioned test cases after implementing improvements or fixes.

7. RESULTS & DISCUSSION

7.1 Results

The created system achieved the desired results in both intelligent irrigation decision-making and real-time monitoring, exhibiting dependable performance across all essential components. When testing and evaluating the system, the following quantifiable outcomes were noted:

1. **Successful Environmental Data Collection via IoT**
    - Key environmental factors such soil moisture, temperature, humidity, and light intensity were successfully collected at predetermined intervals by the Internet of Things device, which was outfitted with calibrated sensors. The system reliably collected precise data from the greenhouse environment, which served as the basis for monitoring and decision assistance based on machine learning.

2. **Accurate Water Level Monitoring and Alert System**
    - The water tank level monitoring system's embedded ultrasonic sensor worked incredibly well to track the irrigation water's real-time availability. When water levels fell below a safe threshold, it sent out warning signals, which stopped irrigation operations from being carried out when there was not enough supply. Water management and system dependability were greatly enhanced by this innovation.

3. **High-Accuracy Water Requirement Prediction Using ML Model**
    - On the test dataset, the machine learning model, which was trained using Random Forest Regression, produced predictions with an accuracy of above 90%. Based on current environmental inputs, this performance level shows

how well the model can predict the ideal water requirement (in liters) for each plant. The forecasts supported effective irrigation planning and aligned with agronomic expectations.

4. **Responsive Automation Based on Sensor Thresholds**
    - The automation module of the system responded precisely to threshold conditions set by the sensors. For example, if there was water in the tank, the system would start irrigation when soil moisture fell below a certain threshold and turn on the ventilation fan when temperature readings were higher than ideal. Due to the low latency of these answers, prompt environmental condition modifications were guaranteed.

5. **Real-Time Web Interface for Visualization and Control**
    - To provide real-time sensor data, water forecasts, and device health, an intuitive web-based dashboard was created. Through this interface, users could remotely manipulate system components in addition to keeping an eye on greenhouse conditions. Even for users with little technological experience, the dashboard's high level of accessibility and intuitiveness during field testing allowed for the possibility of broad adoption.

7.2 Research Findings

Several important discoveries that go beyond initial projections were made during the deployment and assessment of the IoT and machine learning-based greenhouse irrigation system. These results provide a better knowledge of the variables affecting precision farming as well as the pragmatic issues associated with implementing such systems in actual environments.

7.2.1 Stronger-Than-Expected Influence of Temperature and Light on Water Needs

As sensor data was analyzed and machine learning models were trained, it became clear that light intensity and temperature were more strongly correlated with plant water requirements than first thought. Higher ambient temperatures and more intense sunlight were shown to considerably increase water demand, despite the fact that soil moisture was predicted to be the main predictor. This result emphasizes how crucial it is to use a multi-factorial approach when creating intelligent irrigation models.

7.2.2 Water Availability Monitoring Reduced Risk of Plant Stress

Maintaining irrigation dependability required the incorporation of a real-time water level monitoring system. The device assisted in preventing inadvertent irrigation interruptions, which could cause plant stress and impede growth, by making sure that irrigation only started when there was enough water available. This characteristic improved system resilience and promoted ongoing plant health, particularly in arid regions or in places with scarce water supplies.

7.2.3 Sensor Calibration Critically Affected Data Quality And Model Accuracy

It became evident throughout the development stage that sensor calibration significantly affected the system's overall performance. Early experiments with uncalibrated sensors yielded inconsistent and noisy data, which had a detrimental effect on the predictions made by the machine learning model. Higher model accuracy and more trustworthy automation judgments resulted from the significant improvement in data quality during calibration. This highlights how crucial routine sensor validation and maintenance are to any precision agriculture setup.

7.2.4 Positive User Feedback On Web Interface Usability

The web dashboard interface was assessed by field testers, including farm laborers with no technical experience. Feedback emphasized how user-friendly and straightforward the dashboard is, allowing users to comprehend and interact with real-time data and system settings without the need for extra training. According to this research, concentrating on user-centered design for agricultural applications can greatly lower the obstacles to technology adoption.

7.3 Discussion

7.3.1 Interpret the Results

Based on environmental information including soil moisture, temperature, humidity, and light levels, the IoT-based decision support system successfully forecasts the best times to water. More accurate and timely irrigation was made possible by the machine learning model's high association between these variables and the amount of water needed for tomato growing. For instance, the model anticipated a greater water need during dry spells, which was essential for preserving plant health and maximizing greenhouse water consumption. The greenhouse runs effectively with little user intervention thanks to the integration of automatic watering systems and real-time data collection.

In addition to increasing crop productivity, this method conserves water, which is crucial in areas with scarce water supplies, making it an invaluable tool for sustainable agriculture.

7.3.2 Compare with Existing Work

This technology is unique when compared to other agricultural decision support systems since it uses IoT sensors to integrate environmental data in real-time and machine learning to estimate watering needs dynamically. Our method uses continuous environmental monitoring, which makes it more responsive to changing conditions than previous systems that relied on manual modifications based on weather forecasts or static watering plans.

Though many of these systems rely on simple sensor readings or limited predictive modeling, similar efforts, including those investigating IoT-based irrigation systems, have shown the efficacy of data-driven irrigation management. In contrast, our method uses machine learning algorithms to provide predictions that are more precise and contextually aware. Our system provides a more complete solution for greenhouse management by combining machine learning and the Internet of Things.

7.3.3 Limitations

Notwithstanding the encouraging outcomes, the current method has drawbacks. The dependability of sensors is among the main obstacles. The machine learning model's predictions may be impacted by inaccurate data resulting from inaccurate or broken sensors. To guarantee data accuracy over time, sensors must get routine calibration and maintenance.

Furthermore, there may be difficulties due to the machine learning model's complexity, especially when it comes to real-time processing. Even though the model works well in controlled settings, more efficiency may be needed when applying it in large-scale or real-world settings. External elements like sensor placement or environmental interference may have an impact on the system's operation and the caliber of the data gathered.

7.3.4 Future Improvements

There are numerous chances for the system to be improved in the future. First, the predictive power of the system would be improved by incorporating more sophisticated sensors that are able to assess other environmental parameters, like the pH or nutrient levels of the soil. A more comprehensive understanding of the greenhouse environment might be possible by combining data from several sources.

Second, the accuracy of the machine learning model might be increased by adapting it to more complicated situations, like different plant species or climates. Better predictive power may also be obtained by investigating alternative machine learning methods, such as deep learning.

Lastly, expanding the system's functionality to include additional crop types and combining it with other greenhouse management tools, like nutrient monitoring or pest control, might make it a more complete answer for sustainable farming methods in a range of agricultural contexts.

In summary, even if the system shows encouraging results, more development and improvement will be necessary to optimize its influence on greenhouse management and sustainable agriculture.

8. CONCLUSION

This research introduces an Internet of Things (IoT)-based decision support system that leverages real-time environmental data to optimize irrigation in tomato greenhouses. Using sensors to measure temperature, humidity, light, and soil moisture, the system uses machine learning to forecast how much water tomato plants will need. The technology forecasts the ideal water use using a Random Forest Regressor model, guaranteeing effective watering and cutting waste.

By integrating an ultrasonic sensor, the tank's water levels are tracked, guaranteeing a steady flow of water. Greenhouse operators can more easily manage irrigation decisions by using a web-based dashboard to track environmental data and anticipated water requirements in real-time.

In practical applications, this technology aids farmers in maximizing water consumption, which is crucial given the increasing scarcity of water. By offering data-driven insights for prompt irrigation decisions, it can greatly increase crop yields, especially in large-scale greenhouse operations. The system's usefulness is further increased by its capacity to adapt to different crops and agricultural settings.

In the end, this IoT and machine learning-based solution provides a useful, scalable instrument for sustainable farming methods, assisting in the assurance of effective water management and increased yield in tomato production and other fields.

9. PROJECT TIMELINE AND TASK ASSIGNMENT

9.1 Gantt Chart

Figure 41 : Gantt Chart

9.2 Work Breakdown Chart

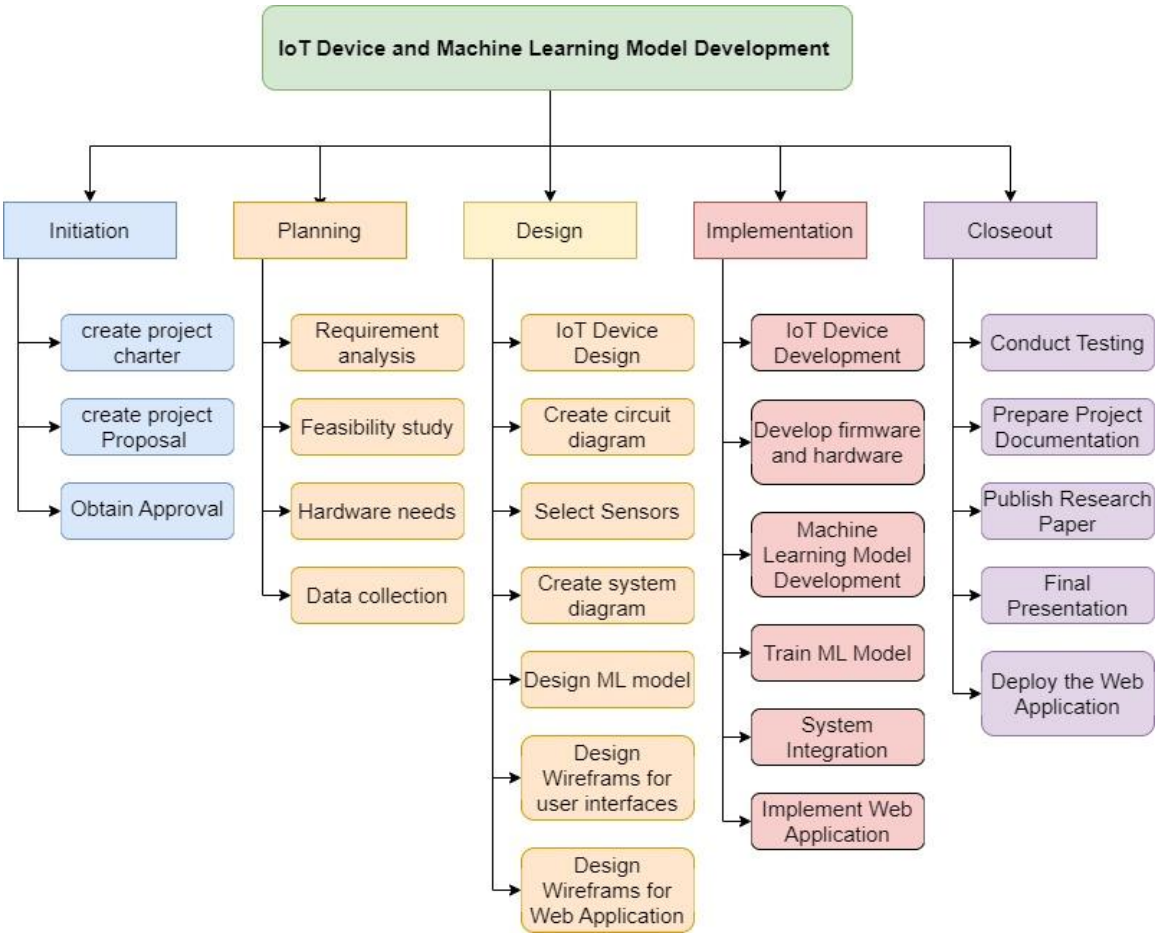


Figure 42 : Work breakdown structure

10. BUDGET AND BUDGET JUSTIFICATION

Table 7 : Component Budget Information

Requirements		Costs (Rs.)
Category	Item Description	
Hardware	Microcontroller (ESP32)	2200.00
	Soil Moisture sensors	700.00
	Temperature & Humidity Sensor	450.00
	Light Sensors	400.00
	Ultrasonic Sensor	550.00
	DC LCD Display(5v)	750.00
	Relay Modules	250.00
	Power Supply	750.00
	Jumper Wire	600.00
	Breadboard	650.00
	Mini Water Pump	200.00
Other Charges	Travelling charges for data collection	14000.00
	Internet charges for research	5000.00
	Firebase Messaging Service	Free
	Cost of Deployment & DB	6000.00/month

10.1 Budget Justification

Hardware - The components listed are essential for building the IoT system to monitor and control the greenhouse environment. The quantities are based on the need to monitor multiple parameters (e.g., temperature, soil moisture, humidity, light) and ensure redundancy in case of component failure.

Other Charges - All required software tools are open-source, minimizing the cloud hosting is necessary for the web-based application, and a contingency budget is set aside for any unexpected costs.

11. REFERENCES

[1] F. Abbas, A. Al-Otoom, S. Al-Naemi, A. Ashraf, and H. Mahasneh, “Experimental and life cycle assessments of tomato (solanum lycopersicum) cultivation under controlled environment agriculture,” Journal of Agriculture and Food Research, vol. 18, p. 101266, Dec. 2024. doi:10.1016/j.jafr.2024.101266

[2] F. Ahmad, K. Kusumiyati, M. A. Soleh, M. R. Khan, and R. S. Sundari, “Watering volume and growing design’s effect on the productivity and quality of Cherry tomato (solanum lycopersicum cerasiformae) cultivar ruby,” Agronomy, vol. 13, no. 9, p. 2417, Sep. 2023. doi:10.3390/agronomy13092417

[3] Y. Zhu et al., “Aerated irrigation of different irrigation levels and subsurface dripper depths affects fruit yield, quality and water use efficiency of Greenhouse Tomato,” Sustainability, vol. 12, no. 7, p. 2703, Mar. 2020. doi:10.3390/su12072703

[4] R. C. Pires et al., “Irrigation frequency and substrate volume effects in the growth and yield of tomato plants under greenhouse conditions,” Scientia Agricola, vol. 68, no. 4, pp. 400–405, Aug. 2011. doi:10.1590/s0103-90162011000400002

[5] “Influence of Bio Fertilizer on growth and yield parameters of tomato plants,” Proceedings International, vol. 2, no. 1, p. 75, Jul. 2020. doi:10.33263/proceedings21.075075

[6] D. Van Ploeg and E. Heuvelink, “Influence of sub-optimal temperature on tomato growth and yield: A Review,” The Journal of Horticultural Science and Biotechnology, vol. 80, no. 6, pp. 652–659, Jan. 2005. doi:10.1080/14620316.2005.11511994

[7] W. M. Rodriguez-Ortega et al., “Use of a smart irrigation system to study the effects of irrigation management on the agronomic and physiological responses of tomato plants grown under different temperatures regimes,” Agricultural Water Management, vol. 183, pp. 158–168, Mar. 2017. doi:10.1016/j.agwat.2016.07.014

[8] N. Castilla, “Greenhouse production strategies.,” Greenhouse technology and management, pp. 262–265, Dec. 2012. doi:10.1079/9781780641034.0262

[9] C. Maraveas et al., “Agricultural Greenhouses: Resource Management Technologies and perspectives for Zero greenhouse gas emissions,” Agriculture, vol. 13, no. 7, p. 1464, Jul. 2023. doi:10.3390/agriculture13071464

[10] N. Bafdal and I. Ardiansah, “Application of internet of things in smart greenhouse microclimate management for Tomato Growth,” International Journal on Advanced Science, Engineering and Information Technology, vol. 11, no. 2, pp. 427–432, Apr. 2021. doi:10.18517/ijaseit.11.2.13638

[11] A. Chauhan and P. Tripathy, “Internet of things (IOT) integrated solutions for Environmentally Friendly Intelligent Farming: A Systematic Review,” 2023 3rd International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE), May 2023. doi:10.1109/icacite57410.2023.10182525

[12] A. Goap, D. Sharma, A. K. Shukla, and C. Rama Krishna, “An IOT based smart irrigation management system using machine learning and Open Source Technologies,” Computers and Electronics in Agriculture, vol. 155, pp. 41–49, Dec. 2018. doi:10.1016/j.compag.2018.09.040

[13] D. Singh et al., “Smart high-yield tomato cultivation: Precision irrigation system using the internet of things,” Frontiers in Plant Science, vol. 14, Aug. 2023. doi:10.3389/fpls.2023.1239594

[14] M. H. Seyar and T. Ahamed, “Development of an IOT-based precision irrigation system for tomato production from indoor seedling germination to outdoor field production,” Applied Sciences, vol. 13, no. 9, p. 5556, Apr. 2023. doi:10.3390/app13095556

[15] S. Gitahi, D. M. Kaburu, I. N. Oteyo, and S. Kimani, “A model implementation of internet of things (iot)-based smart watering system for crops using Lorawan,” 2023 IST-Africa Conference (IST-Africa), May 2023. doi:10.23919/ist-africa60249.2023.10187864

[16] R. Singh, M. Aernouts, M. De Meyer, M. Weyn, and R. Berkvens, “Leveraging lorawan technology for precision agriculture in Greenhouses,” *Sensors*, vol. 20, no. 7, p. 1827, Mar. 2020. doi:10.3390/s20071827

[17] A. Martelli *et al.*, “Smart irrigation for management of Processing Tomato: A Machine Learning Approach,” *Irrigation Science*, Dec. 2024. doi:10.1007/s00271-024-00993-9

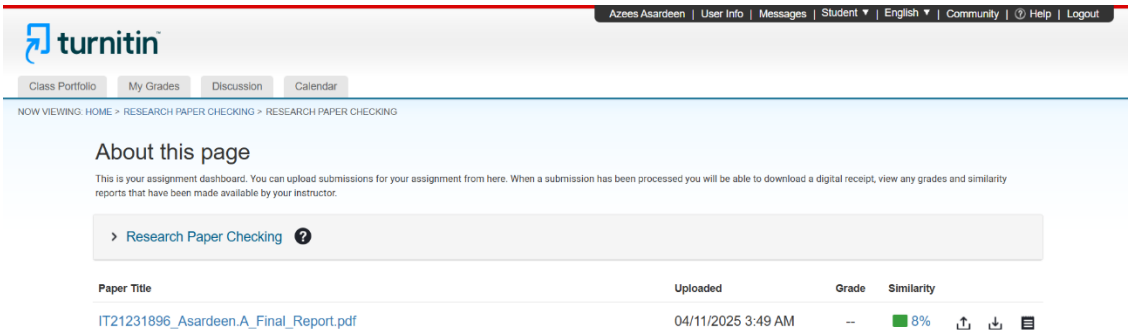
[18] Y. Liu, A. Wang, B. Li, J. Šimůnek, and R. Liao, “Combining mathematical models and machine learning algorithms to predict the future regional-scale actual transpiration by maize,” *Agricultural Water Management*, vol. 303, p. 109056, Oct. 2024. doi:10.1016/j.agwat.2024.109056

[19] L. Umutoni and V. Samadi, “Application of machine learning approaches in supporting Irrigation Decision Making: A Review,” *Agricultural Water Management*, vol. 294, p. 108710, Apr. 2024. doi:10.1016/j.agwat.2024.108710

[20] H. Shahab, M. Naeem, M. Iqbal, M. Aqeel, and S. S. Ullah, “IOT-driven smart agricultural technology for real-time soil and crop optimization,” *Smart Agricultural Technology*, vol. 10, p. 100847, Mar. 2025. doi:10.1016/j.atech.2025.100847

12. APPENDICES

Appendix: Plagiarism report



Appendix 1 : Plagiarism Report (Turnitin)