Design Web Application For IoT Enabled Agriculture Sensor Systems

by

YASH SHETH
202111046

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of

MASTER OF TECHNOLOGY
in
INFORMATION AND COMMUNICATION TECHNOLOGY
to

DHIRUBHAI AMBANI INSTITUTE OF INFORMATION AND COMMUNICATION TECHNOLOGY

May, 2023

Declaration

I hereby declare that

i) the thesis comprises of my original work towards the degree of Master of Technology in Information and Communication Technology at Dhirubhai Ambani Institute of Information and Communication Technology and has not been submitted elsewhere for a degree,

ii) due acknowledgment has been made in the text to all the reference material used.

[illegible]

Yash Sheth

Certificate

This is to certify that the thesis work entitled INSERT YOUR THESIS TITLE HERE has been carried out by INSERT YOUR NAME HERE for the degree of Master of Technology in Information and Communication Technology at *Dhirubhai Ambani Institute of Information and Communication Technology* under my/our supervision.

[illegible] [illegible]

Dr. Vinay S. Palaparthy
Thesis Supervisor

Dr. Saurabh Tiwari
Thesis Co-Supervisor

Acknowledgments

First and foremost, I am deeply grateful to my supervisor Dr. Vinay S. Palaparthy, whose expertise, dedication, and constant encouragement have been instrumental in shaping this thesis. I am indebted to him for his unwavering guidance, insightful feedback, and patience throughout the entire process. He has been an incredible mentor and has inspired me to strive for excellence in my academic pursuits.

I would like to acknowledge the Dhirubhai Ambani Institute of Information and Communication Technology for providing me with the resources, facilities, and opportunities necessary to carry out this research. The academic environment and support from the faculty and staff have been vital in shaping my understanding of the subject matter and facilitating the completion of this thesis.

I would also like to express my gratitude to my family and friends for their constant support, encouragement, and understanding throughout this journey. Their unwavering belief in me has strengthened and motivated me during challenging times.

Lastly, I would like to thank all those individuals who may not be mentioned here but have provided assistance, encouragement, and inspiration in various ways. Their contributions, no matter how small, have played a significant role in successfully completing this thesis.

In conclusion, I extend my heartfelt appreciation to all those who have supported me in this endeavor. Your contributions have been immeasurable, and I am truly grateful for your presence in my life. Thank you for believing in me and for being a part of my academic and personal growth.

Contents

Abstract

Application Design plays a crucial role in Internet of Things (IoT) systems, serving as the data repository for IoT nodes. Additionally, web applications are widely utilized for data analysis and offer remote accessibility. In this study, we have developed a web application for an in-house IoT node. This application collects data from various sensors and uploads it to the server. To achieve this, we have configured a static IP and utilized a standard desktop as the server. The uniqueness of our application design lies in its capability to collect data from over 100 nodes, each containing 16 fields. This stands in contrast to existing open-source applications like ThingSpeak^{TM}, which only support four nodes and eight fields for free account users. We opted for 16 fields to incorporate sensor information and self-diagnostic signals provided by our IoT system. Furthermore, our application also displays the status of each IoT node (active or inactive), which is not a feature in ThingSpeak^{TM}.

List of Figures

CHAPTER 1

Introduction

Agriculture is pivotal in India's economy, supporting over 70% of the population and contributing significantly to the Gross Domestic Product (GDP)[1]. However, the country faces challenges in meeting the demands of its vast population, leading to significant grain and agro-product imports. To address these challenges and improve the efficiency of farming practices, the modernization of the agricultural sector becomes imperative [2].

Plant diseases, weeds, and pests account for roughly 36% of crop loss. This causes a sharp drop in crop productivity, which puts farmers in dangerous and epidemic conditions. The reason for the crop loss is a result of inconsistently used farming techniques, the late discovery of plant disease, and attacks by weeds and pests. Researchers have concentrated on sensor-based technology, the Internet of Things (IoT), machine learning, and artificial intelligence algorithms to reduce crop loss [3]. It has offered a sensible and cost-efficient answer that would improve farming and guarantee its sustainability and financial success. Concerning a variety of applications, including skin hydration, environmental monitoring, explosive detection, biomedicine, and agriculture, sensor-based technology has been crucial in enabling expert decision-making. Many farmers utilize an expert decision system with in-situ innovative sensor-based technologies to increase agricultural output and decrease crop loss from plant disease.

Water is a vital input for plant growth and plays a crucial role in food security in agricultural production. However, unregulated irrigation leads to soil waterlogging, which hampers the germination process by slowing down seed growth due to excessive soil moisture. Insufficient air supply under waterlogged conditions inhibits proper root growth, and excessive irrigation can cause crops to collapse due to strong winds. Uncontrolled irrigation damages crops and diminishes both the quantity and quality of agricultural production.

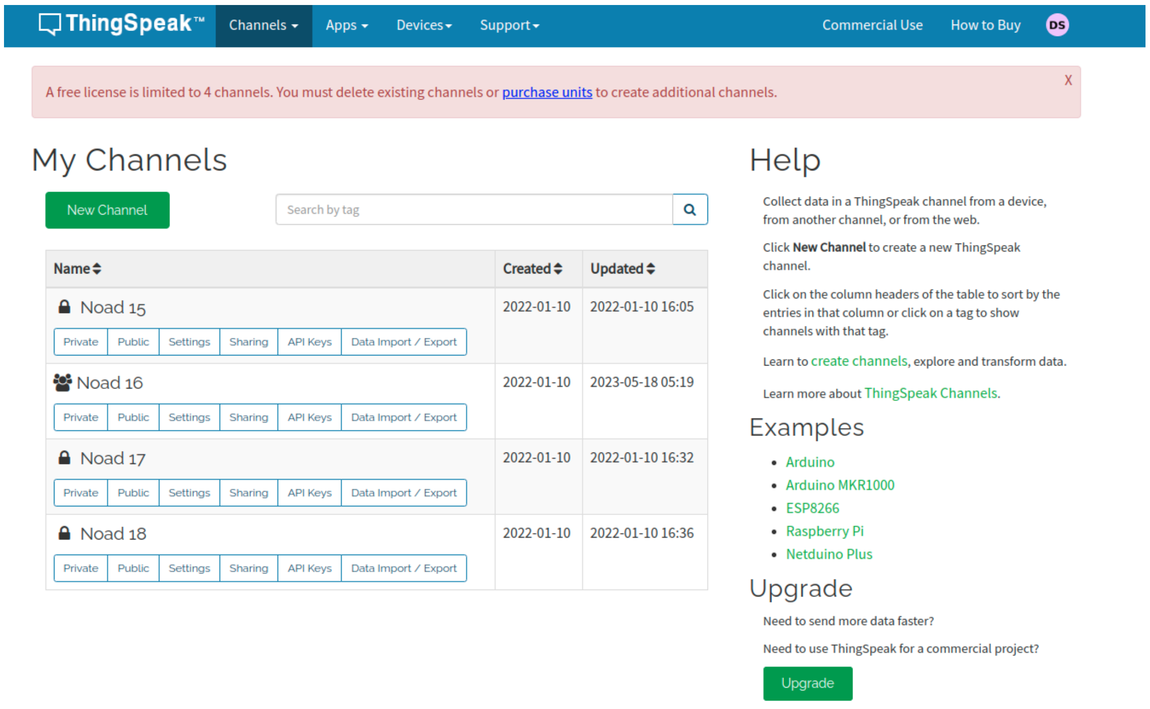
Researchers worldwide have developed disease detection and warning models to reduce the massive crop loss caused by illnesses. For this purpose, soil moisture, soil temperature, ambient temperature, and relative humidity are a few essential elements in developing a model using these sensor data [4], and a properly developed system significantly helps predict the disease that harms the crop. We can achieve early plant disease prediction using machine learning approaches, and because of its incredibly precise forecasts and amity, We may use a variety of datasets - numeric, image, etc. Deep Neural Networks include Artificial Neural Networks(ANN) [5] and Convolutional Neural Networks(CNN) [6]. ANN is suitable for classification and regression problems, and datasets such as tabular, image, and text datasets can be used. In contrast, CNN is the most popular for image datasets.

Many techniques are available, and a lot of work is going on in this field. Still, available solutions are currently inefficient from a usability perspective and are not very cost-effective. Now, known solutions are challenging to purchase in developing nations. Some solutions have individual hardware and software setups for disease prediction and water irrigation. Those are very costly for the majority of farmers. At present many researchers have used the cloud-based platform to store the sensor data. The reported work to date uses IoT-enabled technology to store the data on the cloud platform. For this purpose, various open-source cloud platforms have been used, such as Thingspeak [7] and ThingsBoard [8].

A few third-party systems support the development of these techniques; for example, ThingSpeak provides cloud storage to store the sensor data directly from the IoT sensors [7]. As a free user account, they provide up to four channels with eight data fields. Likewise, ThingsBoard offers similar features. However, one of the major limitations is the number of nodes, and the number of fields per account is limited to 4 and 8, respectively. Below Figure 1.1 shows that there are four nodes created in the free account, and there is a warning when trying to create a new node.

This study aims to create a smart system that can identify plant illness and enhance irrigation by utilizing some key metrics, including Leaf Wetness Sensor, Soil Moisture Sensor, Soil Temperature, Ambient Temperature, and Relative Humidity.

In our work, the developed hardware generates about 16 fields, which comprise sensor data and signals pertaining to the system healing mechanism. Besides this, the number of systems deployed in the field is about 10. Thus, we need an application compatible with our developed hardware to collect copious data from the field. Considering the features offered by the aforementioned open-source platform, there is a dearth of the server, which can collect more than 16



A free license is limited to 4 channels. You must delete existing channels or purchase units to create additional channels.

My Channels

New Channel

Search by tag

Name	Created	Updated
Noad 15 (Private, Public, Settings, Sharing, API Keys, Data Import / Export)	2022-01-10	2022-01-10 16:05
Noad 16 (Private, Public, Settings, Sharing, API Keys, Data Import / Export)	2022-01-10	2023-05-18 05:19
Noad 17 (Private, Public, Settings, Sharing, API Keys, Data Import / Export)	2022-01-10	2022-01-10 16:32
Noad 18 (Private, Public, Settings, Sharing, API Keys, Data Import / Export)	2022-01-10	2022-01-10 16:36

Help

Collect data in a ThingSpeak channel from a device, from another channel, or from the web.

Click **New Channel** to create a new ThingSpeak channel.

Click on the column headers of the table to sort by the entries in that column or click on a tag to show channels with that tag.

Learn to create channels, explore and transform data.

Learn more about ThingSpeak Channels.

Examples

- Arduino
- Arduino MKR1000
- ESP8266
- Raspberry Pi
- Netduino Plus

Upgrade

Need to send more data faster?

Need to use ThingSpeak for a commercial project?

Upgrade

Figure 1.1: Things Speak Dashboard

inputs from the single node (required for the self-healing circuits). A digital system with a self-healing mechanism is becoming very promising, and it is a kind of system upon which one can rely. It can detect failures or faults in the digital system and fix them through self-healing or repair. The systems having such kind of mechanism can recompense failures. Further, these open-source platform does not provide any notifications for system failure, which hinders the system deployed in remote locations.

CHAPTER 2
Literature Review and Research Objective

The literature [9]. presents the challenges faced by the agricultural sector in India, such as excessive importation of grains and agro products due to the demand of the country's 1.2 billion population. The need for modernization becomes evident to improve productivity and self-sufficiency in agricultural production. Several architectural and technological improvements have been suggested in the past to address these challenges.

Various architectural and technological improvements have been suggested and adopted over the years to enhance productivity. Among the challenges, proper monitoring of soil health, environment and optimized irrigation are crucial for achieving sustainable agricultural practices. Traditionally, manual observation-based techniques have been used, resulting in less efficient and less productive crops. This literature survey aims to explore IoT-based solutions for modernizing agriculture and improving crop productivity in India [2].

The survey [10] focuses on the role of IoT in revolutionizing agriculture. IoT enables the interconnection of physical sensing devices in the field with cloud-based platforms, providing real-time data collection and analysis capabilities. This connectivity and data analysis offer farmers immense insights into their agricultural fields, enabling informed decision-making for improved irrigation and crop care.

A significant difficulty in farming is managing irrigation systems, environmental monitoring, and soil health. IoT is used in Paper to discuss a strategy for taking control of these problems. It links irrigation control systems and physical sensing equipment to the cloud. This aids in architecture analysis. Its final findings show that low latency can attenuate real-time data. Agriculture has experienced a significant setback due to migration.

In the literature [11], it states the key aspect of modernizing agriculture is the proper monitoring of soil health. IoT-based solutions utilize various sensors to measure environmental factors critical for crop production, such as soil moisture, temperature, and humidity. This information helps implement precision agriculture techniques, leading to optimized water usage and reduced resource wastage.

In this study [12], they created an IoT-based smart agricultural model that can address various issues that farmers encounter. This model has three modules-crop selection module, crop monitoring system module, and crop maturity level module. The results obtained demonstrate that the proposed work is very effective in making appropriate decisions based on the visualized data viewable at the user end through a Web app. All three modules are adequately documented in the study, along with each module's architecture. Graphs are used to analyze and display all of the data that has been gathered by the various sensors. Farmers can make decisions based on the values, such as whether to irrigate crops if the percentage of soil moisture is low.

In order to assist farmers in need, the paper [13] suggests a method that continuously monitors crop growth and leaf diseases. The proposed system applies machine learning (ML) methods, such as support vector machine (SVM) and convolutional neural network (CNN), to give analytical statistics on plant growth and disease patterns. This system generates effective crop condition notifications to terminal Internet of Things components that help with irrigation, dietary planning, and environmental compliance pertaining to farming fields. In order to identify plant diseases at an early stage, this work suggests using ensemble classification and pattern recognition for crop monitoring systems (ECPRC). For the purpose of identifying leaf and crop diseases, the suggested ECPRC employs ensemble nonlinear SVM (ENSVM).

In the work [3], a soil moisture sensor (SMS) and leaf wetness sensor (LWS) with Internet of Things (IoT) support are constructed. In order to anticipate plant diseases, commercial soil temperature (ST), relative humidity (RH), and ambient temperature (AT) are employed. The developed LWS has a response time of around 20 seconds, a hysteresis of about 3%, and a response of roughly 250% when exposed to air and water. When a 25-75 m thick acrylic protective lacquer (APL) coating is applied on LWS, it is found that the sensor capacitance changes by just 2% when the temperature changes from 20 to 65^0C. Likewise, fabricated SMS offers a response of 10 kHz (F) with only a 2% change in frequency when temperature varies from 20 $\circ$C to 65 $\circ$C and works with an accuracy of ±3%. Further, the aforementioned sensors and an in-house developed IoT-enabled system have been deployed under field conditions for about four months. In this work, we considered Powdery mildew (D1), Anthracnose (D2), and Root rot (D3) disease in the Mango plant. Further, they have implemented the Long Short Term

Memory (LSTM) network, which performs better compared to the existing methods discussed on plant disease management. The proposed network achieves an accuracy of 96%, precision-recall, and F1 score of 97%, 98%, and 99%, respectively.

The system [14] consists of a hardware board and a mobile application. The mobile application connects to the IoT node via Bluetooth, enabling data collection, storage in the ThingSpeak database, control of connected devices, and monitoring their status. The system uses node sensors to detect temperature, humidity, and soil moisture, facilitating data collection and interpretation through smartphone and web applications. The paper also discusses the challenges associated with data preparation, analysis, visualization, and prediction using the Softmax function. Python is utilized for necessary data analysis techniques. Our objective is to eliminate the limitations of this paper as the focus of this paper is similar to what we are planning to achieve.

Objective

The primary obstacle in the present field is ensuring that farmers have access to essential information and timely assistance. Obtaining knowledge to support sustainable agriculture can be challenging, as it may either be non-existent or difficult to locate. In this context, the web application proves to be intelligent, offering effective solutions to improve the dissemination and acquisition of information regarding sustainable agricultural practices.

With this motivation, we have designed the web application for the in-house developed IoT-enabled systems in this work. To overcome all the issues associated with ThingSpeak and ThingsBoard, such as more data fields per node, a more significant number of nodes per user, node status (active or not), specific pieces of hardware working correctly, email notifications on some particular event, runtime prediction of diseases, and water irrigation on the fields. All these motivated us to develop a new project system that includes all the requirements mentioned earlier, works efficiently, and is cost-effective.

CHAPTER 3
Application Back-end Design

3.1 Hardware

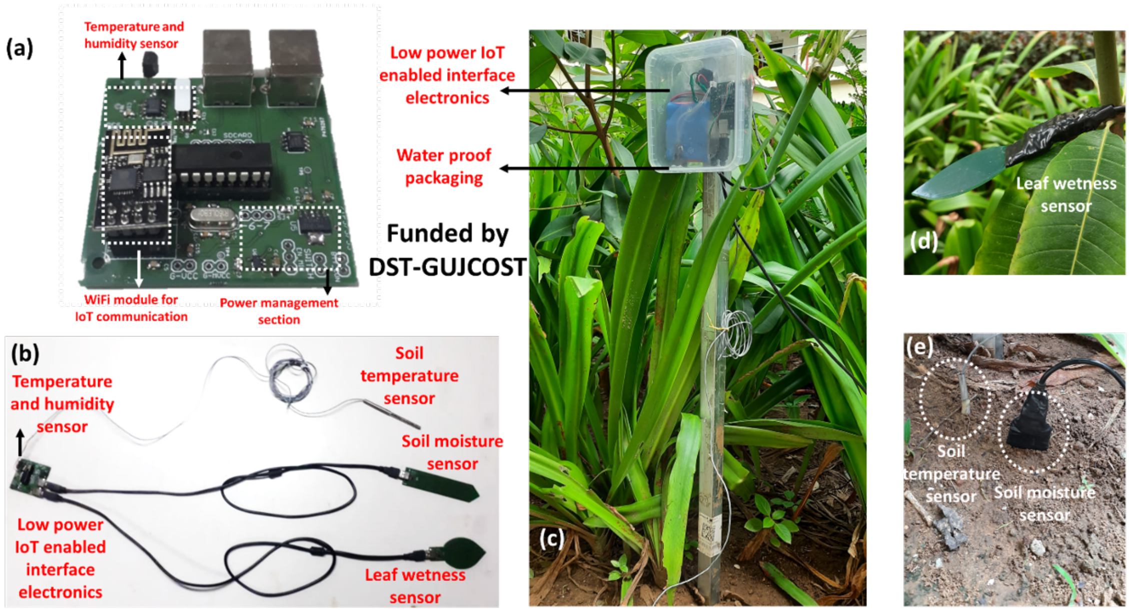


Figure 3.1: Hardware Module

In this work, besides Leaf Wetness Sensors and soil moisture sensors, we have considered the soil temperature, ambient temperature, and relative humidity, which are essential and widely used for early plant disease forecasting. This work uses a commercially available MCP 9701A temperature sensor and HIH 5030 humidity sensor. Whereas the soil temperature sensor has been developed by Proximal Soilsens Tech Pvt. Ltd. Fig. 3.1 (a) shows the developed hardware, which comprises the two USB connectors where LWS and soil moisture sensor are connected, soil temperature, ambient temperature and humidity sensor, Wi-Fi module to establish IoT communication, signal processing unit (uC), and power management section.

Fig. 3.1 (b) shows the fabricated LWS, soil moisture sensor, and soil temperature sensors are connected to the section. Fig. 3.1 (b) shows that the fabricated LWS, soil moisture sensor, and soil temperature sensors are connected to the developed hardware using an A to B USB connector. Developed hardware comprises onboard ambient temperature and humidity sensors and IoT-enabled data transmission features.

Fig. 3.1 (c) shows the developed sensor system deployed in the field near the young Mango plant. Efforts are made to attain waterproof packaging. Further, considering the operational exposure of the LWS, the fabricated LWS has been kept at a 45 $\circ$C angle, as depicted in Fig. 3.1 (d). In Fig. 3.1 (d), LWS acts as the artificial leaves on the plant. Fig. 3.1 (e) shows the location of the soil moisture and soil temperature sensor deployed in the field. We have deployed this system for about four months.

3.2 Development Server

It is a specialized server environment developers use during the software development process. It provides a controlled and isolated environment for us to build, test, and debug our applications before deploying them to a production server. To develop any web application, we require a server to host it on. Currently, we use a local environment to host the application. Our setup includes an Intel i5 9th generation (3 GHz) Hexa core processor, 8GB of RAM, and 512GB of SSD storage. The operating system on the server is Ubuntu 20.04 LTS (Linux), a 64-bit open-source environment preferred for server development. The server must be connected to the internet to gather data from IoT sensors and always be available. Additionally, it needs a static IP address that the IoT sensors can use.

Apart from the local environment, we have integrated the code with the version control system. We used git to manage different versions of the code. Git helps maintain the code and makes it easy for team members to collaborate. With the help of this platform, all the members can work simultaneously while keeping track of the changes made by others, which makes the development process more seamless.

3.3 Application

We used Python to develop the application. Python is renowned for its understandable and clean syntax, which makes writing and maintaining code simpler. It strongly emphasizes the readability of the code, which simplifies and streamlines development. Moreover, it has a large and active developer community, making discovering many tools, frameworks, libraries, and support easier. This community supports the language's stability and progress, ensuring it is constantly improved. Also, python is a versatile language that can be used for various applications, including web development, data analysis, machine learning, artificial intelligence, scientific computing, and more. Its versatility will allow us to use the same language for each project component.

We employed the Django framework to streamline web application development with more functionality without manually implementing each feature. Django's high-level Python web framework has many built-in tools and capabilities. It adheres to the DRY (Don't Repeat Yourself) philosophy, which helps us produce code more quickly and effectively. Django offers scalability by providing database migration, caching, and load-balancing features. This simplifies the process of scaling web applications, allowing them to handle increasing traffic and data loads. Django has built-in security features, such as protection against common web vulnerabilities, such as cross-site scripting (XSS) and cross-site request forgery (CSRF). It also provides authentication and authorization mechanisms, making it easier to implement secure user-management systems.

The framework also offers a robust Object-Relational Mapping (ORM) system, allowing developers to interact with the database using Python code rather than writing SQL queries. Django also includes a pre-built administrative interface, authentication, and authorization mechanisms, supporting various caching and messaging frameworks. Django has a vast range of modules, with numerous essential modules by default. The Model-View-Template (MVT) architectural pattern is followed by Django, where the model represents the data, the view represents the business logic, and the template represents the user interface.

The proposed model requires a server to collect real-time data for analysis; to do so, we connected a desktop computer to the Internet and assigned it a static IP address. This allows the IoT hardware module to send data to the server by using the fixed IP address of the computer. We also placed the hardware module in the field at the DAIICT premises to begin collecting data from the IoT sensor. Subsequently, a web application using Django was created.

A database and corresponding schema are crucial to developing a web application. While Django primarily supports relational databases like PostgreSQL, MySQL, and SQLite, however, these traditional relational database technologies may not be suitable for storing and processing data generated by the Internet of Things (IoT) due to their limitations in handling large amounts of unstructured data, slow processing speeds, and high storage costs. These limitations can be encountered by integrating Django with MongoDB. MongoDB's flexible document-based data model allows us to store structured, semi-structured, and unstructured data without rigid schema constraints. It is also designed to handle large-scale applications and can scale horizontally across multiple servers. Hence it can operate efficiently even when data is growing rapidly. Not only that, MongoDB's change stream functionality will help us capture real-time data changes within the database. This feature will become handy while implementing real-time notifications or activity feeds,

PyMongo is a Python library that makes it easy for Django developers to connect to and interact with MongoDB, a NoSQL database. It provides a convenient interface for performing CRUD operations on MongoDB documents. By integrating PyMongo, developers can take advantage of MongoDB's document-oriented data model, flexible schemas, and indexing capabilities. This allows them to efficiently handle complex or dynamic data structures and utilize MongoDB's performance advantages for specific use cases. Additionally, PyMongo supports multiple databases and empowers developers to perform advanced data analysis and aggregation using MongoDB's aggregation framework. The pillow is a Python imaging library that makes it easy to manipulate images. It provides a wide range of features, including resizing, cropping, rotating, adjusting image properties, applying filters, drawing shapes and text, and enhancing image quality. Pillow supports a wide range of image formats, making it a versatile tool for working with images in Python.

Since we are utilizing a No-SQL database, there is no need for a strict schema. However, we have designed a basic schema for our application in order to define essential features and establish relationships between entities. Figure 3.2 displays this schema, which illustrates the connections among the user, node (from a software standpoint), and feeds. The schema is a valuable tool for designing and developing our application. It helps us to define the essential features of the application and to establish relationships between entities. The schema also helps us to ensure that the application is scalable and flexible.

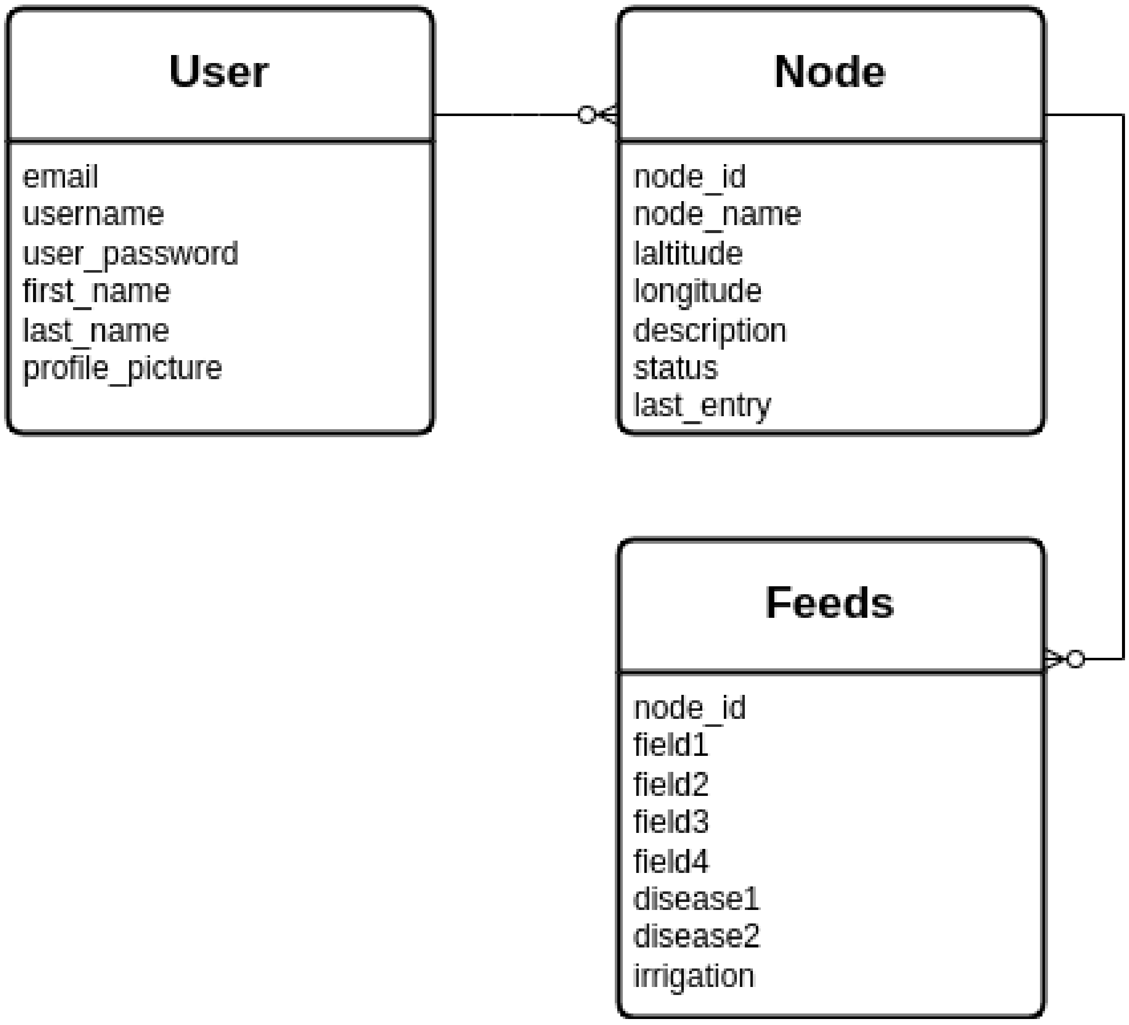


Figure 3.2: Database schema for application

Initially, we established user management within our web application to ensure that only authorized individuals can access it. This involves implementing user registration and login procedures. To enhance our user management system's effectiveness and reduce duplicate accounts, we have incorporated email verification. To accomplish this, we have employed the services of Zoho email. When a user submits the registration form, our web application sends a verification email to the provided email address. Only upon successful verification can users fully utilize the web application. Access to the application is exclusively granted to verified users. Once verified, users gain access to personalized profiles, enabling them to upload profile pictures and update select information. Additionally, the system offers a password change feature, allowing users to modify their passwords as desired. Figure: 3.3 illustrates the user registration and login processes flow.

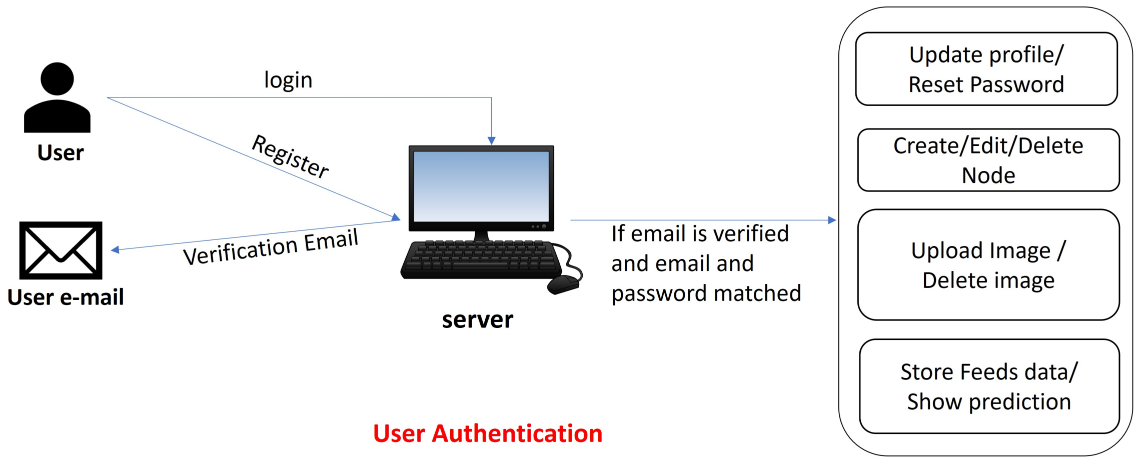

User
login
Register
User e-mail
Verification Email
server
If email is verified and email and password matched
Update profile/ Reset Password
Create/Edit/Delete Node
Upload Image / Delete image
Store Feeds data/ Show prediction
User Authentication

Figure 3.3: User Verification flow

Chapter 4
Application User Interface Design

The user interface (UI) is a crucial component of any application or system as it serves as the main point of interaction between users and technology. It has several important roles, including enhancing usability, improving the overall user experience (UX), increasing efficiency and productivity, ensuring consistency and familiarity, reducing errors, and contributing to branding and perception. A well-designed UI should be intuitive, visually appealing, and easy to navigate, providing users with a seamless and enjoyable interaction. It should also streamline workflows, prevent errors, and reflect the brand identity, ultimately leading to higher user satisfaction and adoption.

In summary, the user interface is vital for creating a positive user experience and ensuring the success of an application or system. It should prioritize usability, efficiency, and consistency while also reducing errors and reflecting the brand's image. By focusing on these aspects, developers can create intuitive, visually appealing interfaces and are capable of providing a seamless interaction between users and technology.

4.1 User Management

As explained in the previous section, user management consists of multiple functionalities, including registration, login, logout, profile management, and admin-specific view. The admin can view all the users in the system and can also handle their configurations as needed.

4.1.1 User Registration

Figure 4.1 show the user registration UI page provides a user-friendly interface for individuals to create new accounts and join the web application. It typically includes a form where users can input their desired credentials, such as a unique username, email address, and password. The form may also include additional fields for personal information, such as name, email, or profile picture.

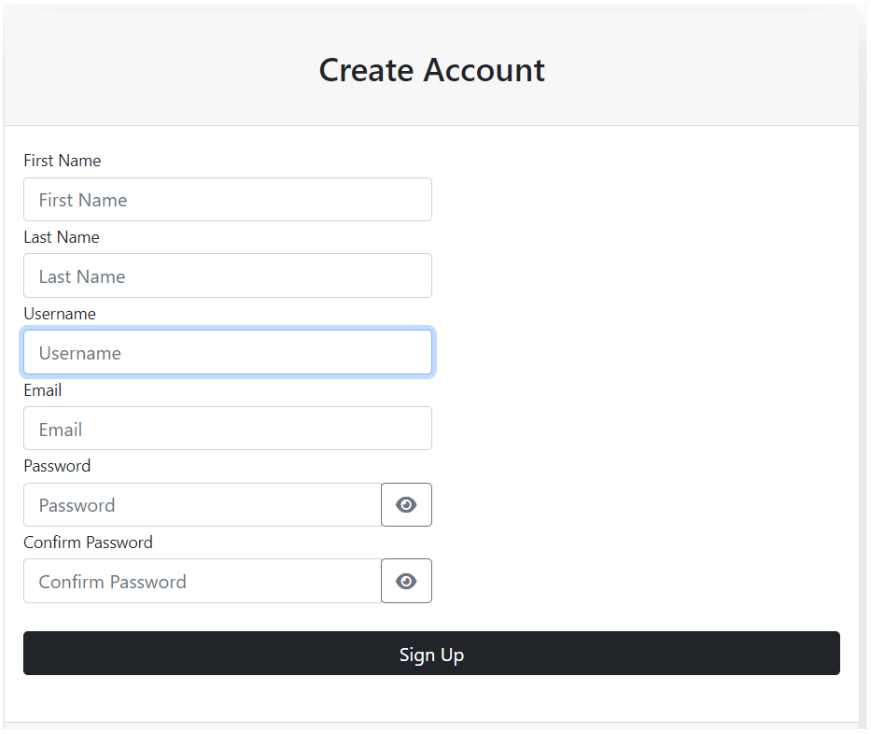


Figure 4.1: User Registration

Registration page often incorporates validation mechanisms to ensure that users provide valid and secure information. This can involve checking the uniqueness of the chosen username or email address, enforcing password complexity requirements, and validating the format of input fields (e.g., email validation).

Once users have filled out the registration form accurately, they can submit their details. At this point, the web application will initiate the email verification process by sending a verification link or code to the provided email address. After email verification, users can log into their portal using their credentials.

4.1.2 User SignIn

Upon accessing the application, individuals are directed to the SignIn interface, which serves as the default page. Only authenticated users are allowed to Sign In successfully; otherwise, they encounter an error based on the circumstances. In case a user wishes to establish a new account, they can navigate to the user registration view. Additionally, a redirection feature is available for users who have forgotten their passwords. Figure 4.2 illustrates the interface used for the Sign In process.

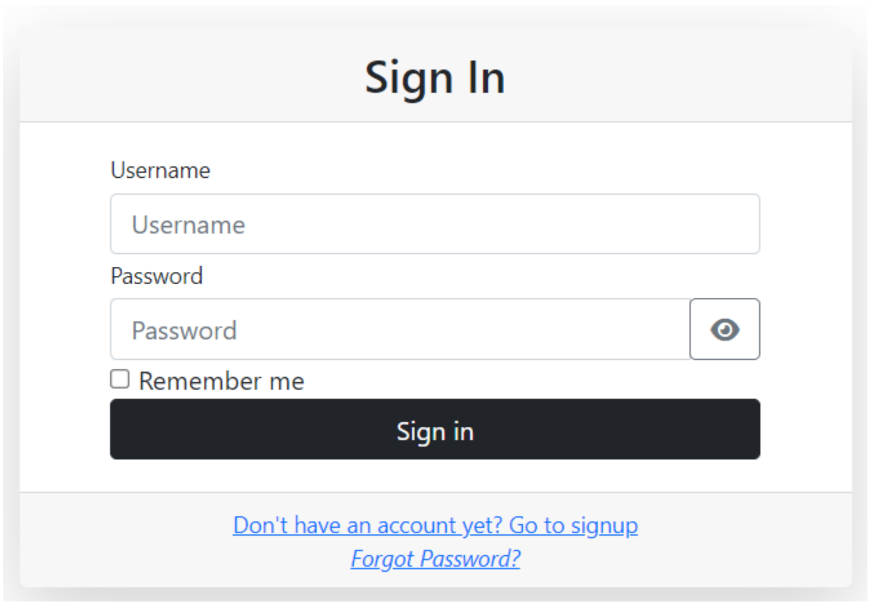


Figure 4.2: User SignIn

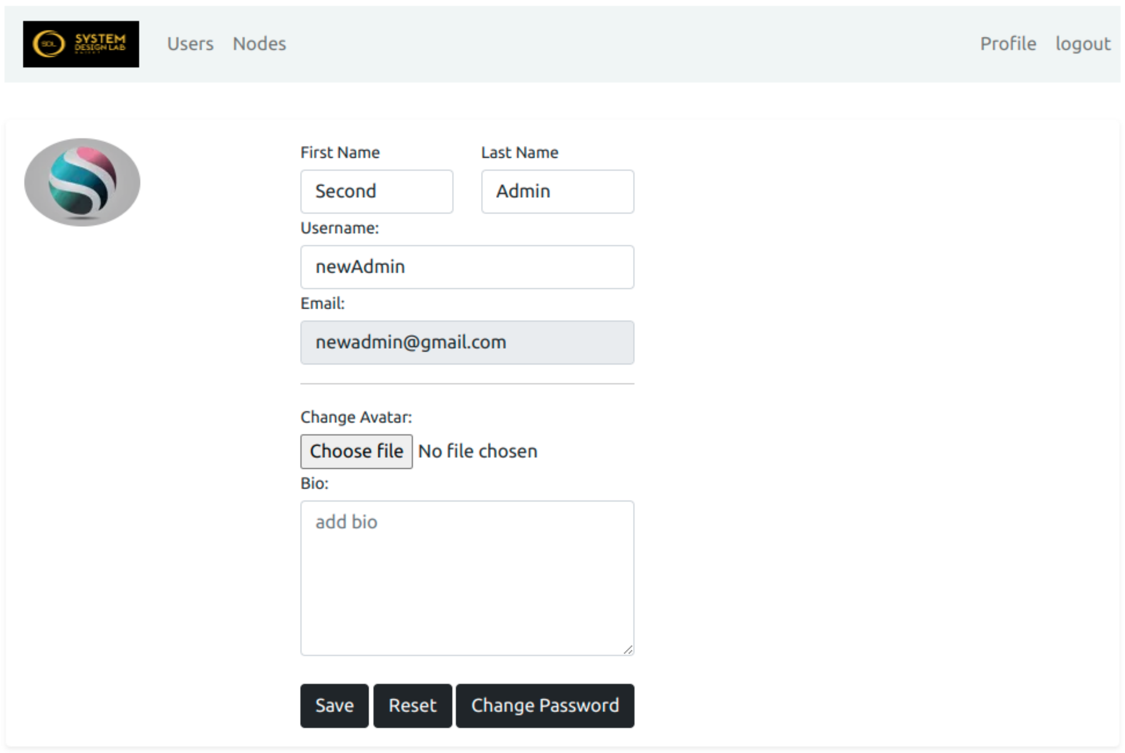
4.1.3 User Profile

The profile page is a central hub for users to manage their personal information and access various functionalities. It includes fields such as first name, last name, and date of birth (DOB), allowing users to view and update their essential details. Users can easily modify their information by editing the corresponding fields on the profile page.

In addition to basic profile information, the page provides additional functionalities to enhance the user experience. One such feature is the ability to upload an image, enabling users to personalize their profile and display a unique avatar or photo. This functionality allows users to express their individuality and make their profile visually appealing.

Another important functionality offered on the profile page is the option to change the password. By providing a secure and convenient way to update their password, users can maintain the privacy and security of their accounts. This feature is crucial in protecting user data and accounts from unauthorized access.

Overall, the profile page combines essential user information with convenient features like image uploading and password changing. It aims to provide users with a seamless and personalized experience while maintaining their privacy and security.



First Name: Second
Last Name: Admin
Username: newAdmin
Email: newadmin@gmail.com
Change Avatar: Choose file No file chosen
Bio: add bio

Figure 4.3: Users Profile

4.1.4 Admin View

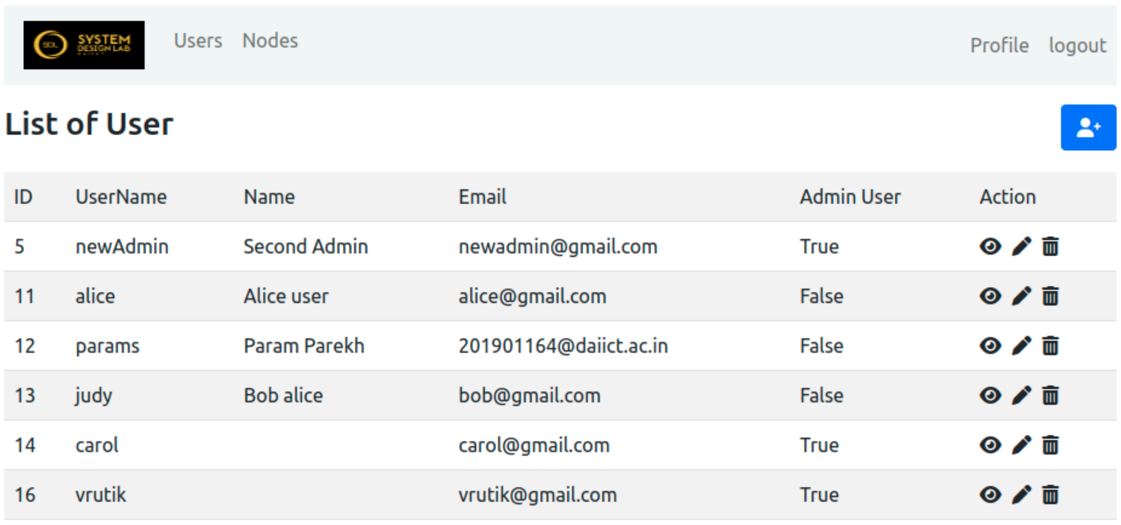
The admin view is a powerful tool that grants administrators access to manage and oversee the users in the system. It presents a comprehensive list of all the users registered within the system, allowing the admin to have an overview of the user base. This comprehensive list empowers the admin to locate and access individual user profiles easily.

Within the admin view, administrators possess the authority to perform various actions on user accounts. One such action is the ability to view user profiles, enabling the admin to gain insights into each user's details and settings. This feature aids in understanding user behavior, preferences, and individual needs.

Furthermore, the admin view equips administrators with the capability to handle any malicious or unauthorized activities within the system. Administrators can identify and remove any users who engage in malicious behavior or violate the system's policies. This proactive approach ensures the security and integrity of the system, protecting both the users and the overall user experience.

Additionally, administrators can modify certain user settings as deemed necessary. This allows the admin to make adjustments to user accounts, ensuring they align with the system's guidelines and requirements. By having the ability to change specific user settings, administrators can tailor the system to accommodate unique user needs or respond to emerging challenges effectively.

In summary, the admin view provides administrators with a centralized platform to oversee and manage user accounts. It offers a comprehensive user list, user profile viewing capabilities, the ability to handle malicious users, and the authority to adjust user settings. This robust set of functionalities empowers administrators to maintain system security, optimize user experiences, and efficiently manage the user base.



List of User

ID	UserName	Name	Email	Admin User	Action
5	newAdmin	Second Admin	newadmin@gmail.com	True	
11	alice	Alice user	alice@gmail.com	False	
12	params	Param Parekh	201901164@daiict.ac.in	False	
13	judy	Bob alice	bob@gmail.com	False	
14	carol		carol@gmail.com	True	
16	vrutik		vrutik@gmail.com	True	

Figure 4.4: Admin View for Users

4.2 Node Management

Once the foundational modules were finished, we introduced a feature called Node Management. This feature enables users to create software nodes and establish connections with the hardware module. Users can create, edit, and delete the entire node setup as per their requirements.

A unique node ID is automatically generated when a user creates a new node. This node ID acts as a crucial link between the hardware module and the web application. It ensures that data from the hardware module is accurately received and associated with the corresponding node ID in the web application. By establishing this connection, users can effectively monitor and manage the data coming from the hardware module through the designated node ID.

4.2.1 Node Creation

To create a node, users can utilize a form that includes several fields to provide the necessary information. The form typically consists of fields such as name, latitude, longitude, description, and status.

In the form, users can enter a unique name to identify the node, enabling easy reference and identification. The latitude and longitude fields allow users to specify the geographic coordinates of the node's location, providing precise positioning information.

The description field allows users to provide additional details or notes about the node, such as its purpose, function, or any relevant information that helps in understanding its role within the system. This description can assist in better organization and management of the nodes.

The status field indicates the current state or condition of the node. Users can choose from predefined options, such as "active" or "inactive," to reflect the operational status of the node accurately.

Users can successfully create a node by completing the form and providing the necessary information. This process ensures the system has all the essential details required to effectively manage and monitor the node. Figure 4.5 shows the form to create a new node.

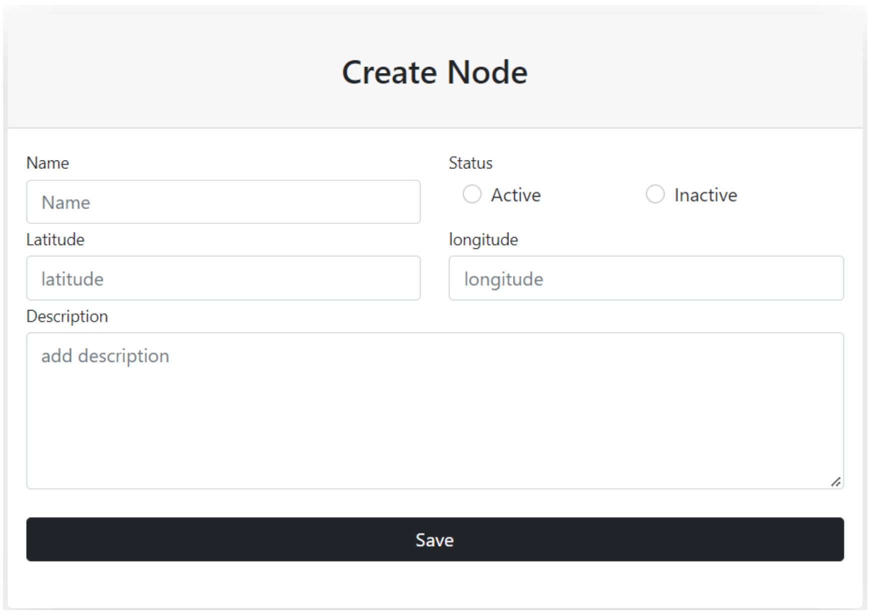


Figure 4.5: Node Creating Form

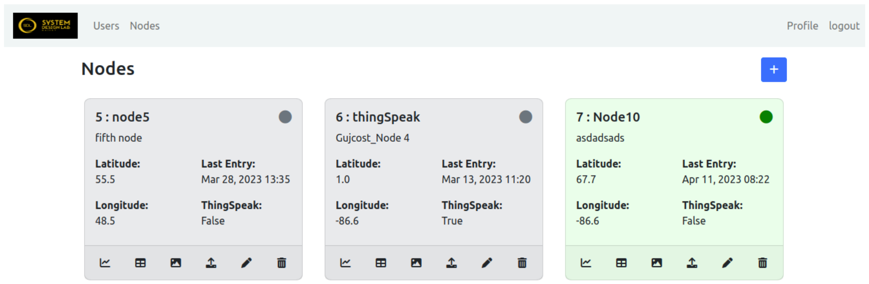
4.2.2 Node List View

Each node within the system possesses several fields, including name, description, latitude, and longitude. When a user registers the node ID with the hardware module, the node becomes capable of receiving data from the designated field. Users can conveniently access and view this data within the feeds section of the corresponding node. The interface visually represents nodes as cards, providing an organized and easily navigable view.

In the event that a node fails to receive data for a period exceeding 30 minutes, it will be displayed in a grayed-out state. This visual indication signifies that the node is not functioning correctly, possibly due to external factors or a depleted battery. Conversely, properly functioning nodes will be presented in a green state, offering users a clear distinction between functioning and non-functioning nodes.

When a user decides to delete a node, all feeds associated with that node are permanently removed as well. This ensures that the deletion process is comprehensive and eliminates any residual data from the system.

Suppose a node fails to receive data for more than 30 minutes. In that case, it will be displayed in a grayed-out state, indicating that it is not functioning correctly due to external factors or battery depletion. Conversely, properly functioning nodes will be presented in green. If a user decides to delete a node, all associated feeds stored by that node will also be permanently removed.



5 : node5
fifth node
Latitude: 55.5
Last Entry: Mar 28, 2023 13:35
Longitude: 48.5
ThingSpeak: False

6 : thingSpeak
Gujcost_Node 4
Latitude: 1.0
Last Entry: Mar 13, 2023 11:20
Longitude: -86.6
ThingSpeak: True

7 : Node10
asdadsads
Latitude: 67.7
Last Entry: Apr 11, 2023 08:22
Longitude: -86.6
ThingSpeak: False

Figure 4.6: Node Management

4.2.3 Feeds View

In the system, "feed" refers to the data generated by the hardware sensors. To streamline the management of these feeds, the system incorporates a feature that enables users to access their feed data. The hardware module transmits this data to the database, and users can retrieve it by simply clicking on the "Feeds" button within the node card view. This functionality empowers users to effectively comprehend and visualize the sensor data, facilitating the monitoring of changes over time.

Users are presented with two visualization options when accessing their feed data: a tabular view and a chart view. These options are thoughtfully designed to be user-friendly and accessible to individuals with varying technical backgrounds. The chart view displays a line chart representing the primary sensor readings and battery levels, illustrating how these values change over time. On the other hand, the tabular view provides a comprehensive display of the sensor data, including additional information. While the tabular view is highly informative, it may be less easily understandable for non-technical users.

The system's feed management feature allows users to access and interpret their hardware sensor data. The chart view visually represents the data's temporal changes, while the tabular view provides detailed information. By presenting the feed data in different formats, the system ensures a satisfactory user experience that caters to both technical and non-technical users.

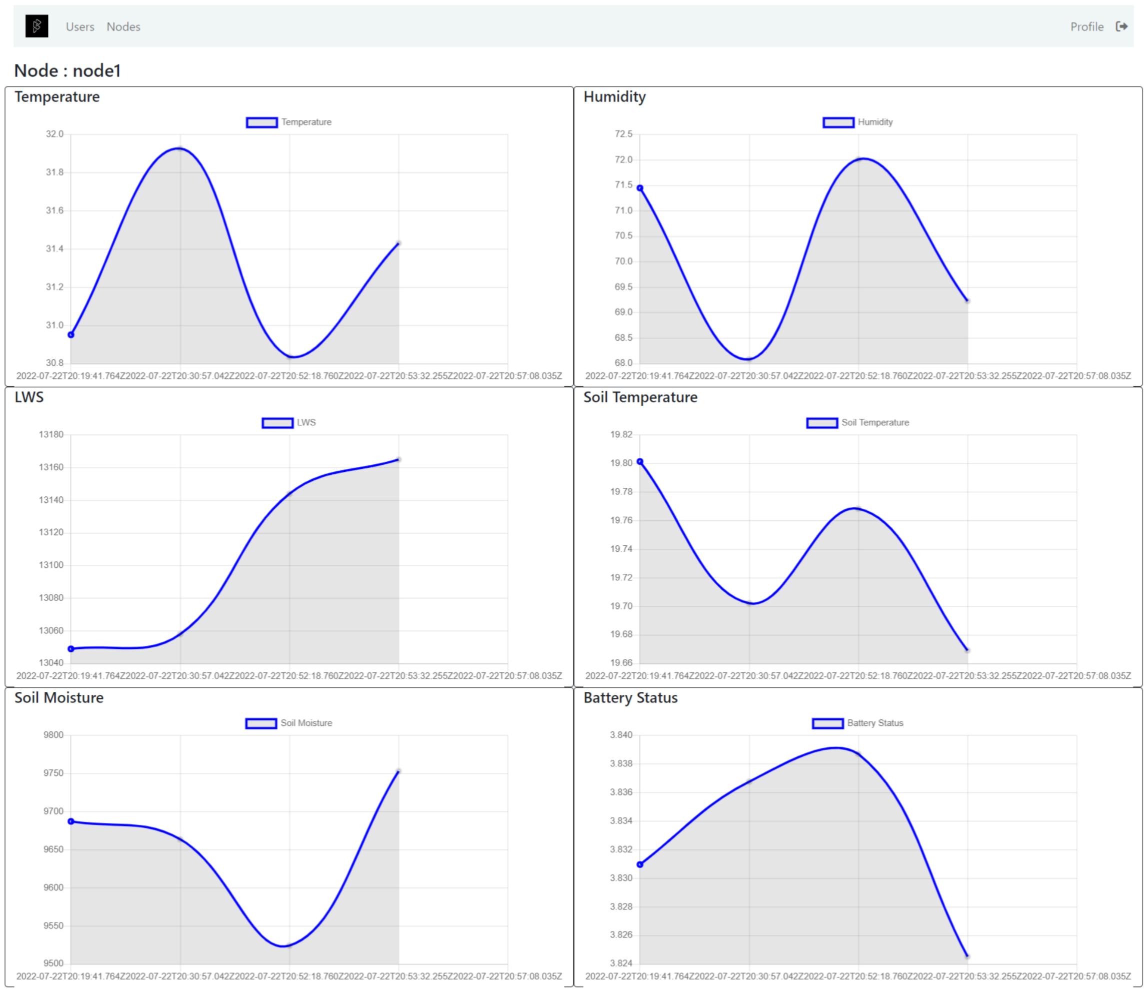
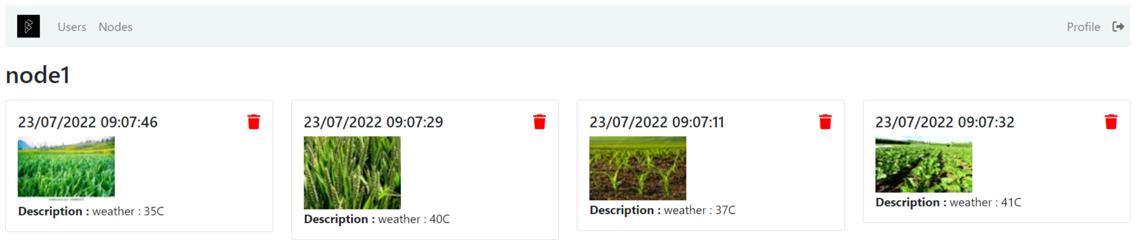


Figure 4.7: Feed Visualization

4.2.4 Image Gallery

We have implemented an Image Gallery feature that allows users to upload images for various purposes such as reference and record-keeping. Users can easily add images to the gallery and also have the option to provide brief descriptions to provide additional context or information about the uploaded images. This feature is particularly valuable for capturing and documenting the regular changes of crops or any other relevant visual data. All uploaded images are organized and displayed on the Image Gallery page, arranged in chronological order based on their respective dates.

The images uploaded to the Image Gallery serve an additional purpose as they are utilized to create an image dataset. This dataset plays a significant role in training machine learning algorithms, enabling the development of more accurate disease prediction models. By leveraging the images in the dataset, machine learning algorithms can analyze and identify patterns, contributing to the enhancement of disease prediction capabilities. This ultimately leads to improved accuracy in detecting and predicting diseases affecting crops or other areas where image analysis is applicable. Figure 4.8 shows a specific node's galley view.



node1

23/07/2022 09:07:46
Description : weather : 35C

23/07/2022 09:07:29
Description : weather : 40C

23/07/2022 09:07:11
Description : weather : 37C

23/07/2022 09:07:32
Description : weather : 41C

Figure 4.8: Image Gallery

CHAPTER 5

Data Collection in application

5.1 Data Collection

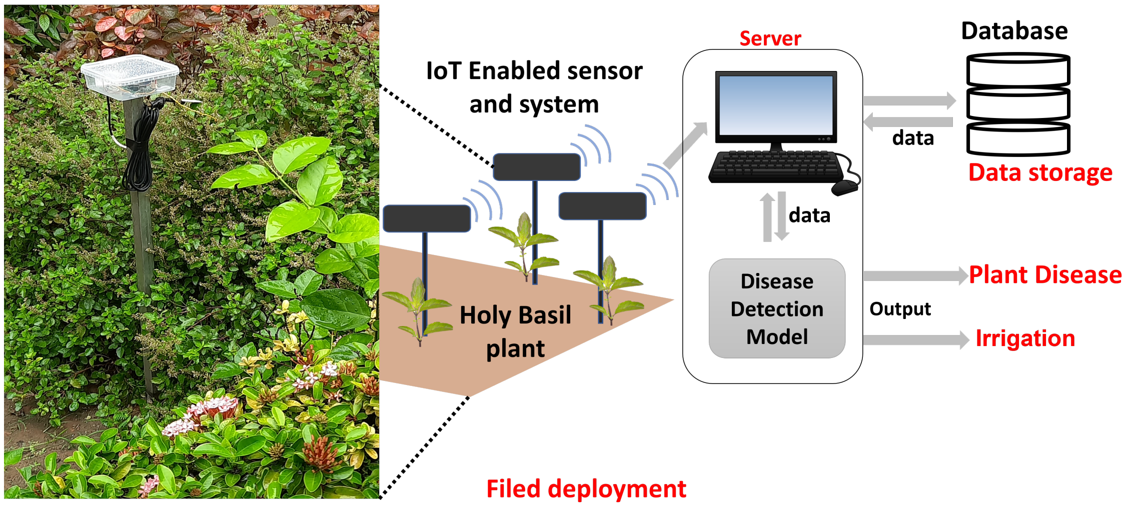


Figure 5.1: Field deployment and server workflow

A hardware module, referred to as the node, has been deployed in the field within the premises of DAIICT. This location is where crops such as bananas, cumin, groundnut, and cotton are cultivated. The node is connected to a WiFi network and has been programmed to transmit data to a designated server at a predetermined frequency. The server we have developed for our application is responsible for storing the sensor readings obtained from the node.

During the development phase, both the hardware module and the server were maintained on the same network. The node collects data from various sensors and creates an HTTP POST request to communicate with the server's API endpoint. Upon receiving the request, the server validates the Node ID provided and proceeds to store the received data under the corresponding node ID on the server. This stored data is then used for making predictions and presented on the web application dashboard, providing users with a visual representation of the collected information.

Here are some further insights into the process of collecting data:

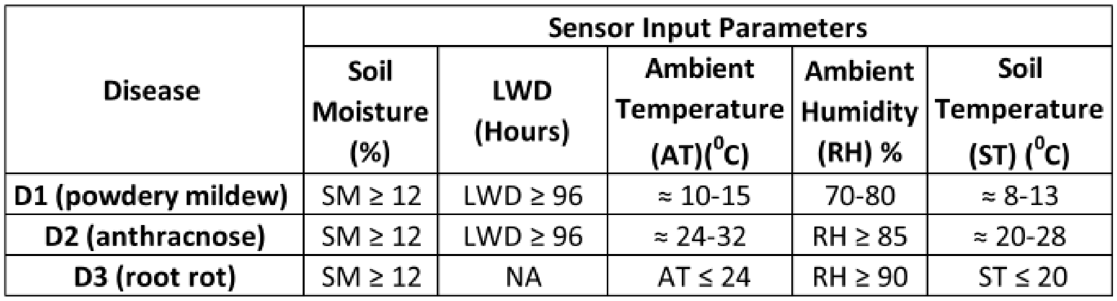
- The node gathers data from multiple sensors, including those measuring temperature, humidity, soil moisture, soil temperature, and leaf wetness.
- Data collection occurs at regular intervals, specifically every 30 minutes.
- The collected data is stored in a database located on the server.
- The data serves as input for generating predictions related to crop yields.
- These predictions are subsequently showcased on a web application dashboard.

Gaining a comprehensive understanding of the information and obtaining initial insights from the experimental data are vital prerequisites before applying any machine learning technique. These insights play a crucial role in determining the preprocessing techniques necessary to prepare the data for input into the neural network. In this particular study focusing on predicting plant diseases in mango plants, five specific input features were monitored: soil moisture (F1), leaf wetness duration (F2), ambient temperature (F3), relative humidity (F4), and soil temperature (F5). Analyzing these features makes it possible to detect the probability of these diseases occurring in the plant. Therefore, it is imperative to emphasize the significance of conducting exploratory data analysis to capture all relevant background knowledge.

Monitoring the distribution of plant diseases across different months is important for early disease prediction models. It is crucial to understand the range of values for all input features, as this can contribute to the development of plant disease. The criteria for the germination of diseases, namely powdery mildew (D1), anthracnose (D2), and root rot (D3), are tabulated in Table 5.1. These criteria must be met in order for a disease to germinate.

We have used these criteria to label the data points in our dataset. Each data point is labeled using three binary labels in a one-hot encoded manner, representing the presence or absence of each of the three diseases studied. A label of 1 indicates that the conditions mentioned in 5.1 are satisfied, while 0 indicates the opposite. This labeling approach is suitable because a plant can simultaneously be susceptible to multiple diseases.

Field deployments have yielded approximately 2675 self-collected data points over a span of four months. Among these, around 2500 data points are tagged as



Disease	Sensor Input Parameters				
	Soil Moisture (%)	LWD (Hours)	Ambient Temperature (AT)(^{0}C)	Ambient Humidity (RH) %	Soil Temperature (ST) (^{0}C)
D1 (powdery mildew)	SM ≥ 12	LWD ≥ 96	≈ 10-15	70-80	≈ 8-13
D2 (anthracnose)	SM ≥ 12	LWD ≥ 96	≈ 24-32	RH ≥ 85	≈ 20-28
D3 (root rot)	SM ≥ 12	NA	AT ≤ 24	RH ≥ 90	ST ≤ 20

Table 5.1: Plant Disease

they meet the criteria for the presence of at least one of the three diseases listed in Table 5.1. Data points are selected if they satisfy at least two out of the five conditions mentioned in Table 5.1.

The numbers of data points tagged as D1, D2, and D3 are 277, 2495, and 1685, respectively. These statistics indicate that D2 and D3 were more prevalent during the experiment, which aligns with the findings in [4] considering the environmental factors.

5.2 Data Preprocessing

As mentioned previously, the model's development and training involve five key parameters. Among these parameters, the model can directly utilize ambient temperature, ambient humidity, and soil temperature. However, the remaining two parameters require derivation from the data collected by specific sensors.

The leaf wetness duration (LWD) can be obtained by analyzing the data collected by the leaf wetness sensor. Similarly, the gravimetric water content (GWC) can be derived from the frequency data obtained from the soil moisture sensor. While these calculations are relatively straightforward when using historical or long-range data, performing them in real-time poses challenges. To measure LWD and GWC in real-time, we typically retrieve a CSV file from the cloud platform and perform the necessary calculations to obtain a comprehensive dataset. However, for real-time measurement of LWD and GWC, we rely on point data, which is primarily utilized for disease and irrigation prediction.

5.2.1 Leaf Wetness Duration Calculation

In the initial phase of data preprocessing, we utilize denoising techniques to refine the data obtained from the first leaf wetness sensor (LWS). Subsequently, we calculate the baseline value for the LWS. We identify specific events based on these variations by computing the disparity between the baseline and the actual data values. Each event corresponds to a distinct data frame, and the duration of each event is a significant parameter used for prediction purposes. To visually represent this process, Figure 5.2 displays a plot showcasing the denoised signal represented by the line, the baseline illustrated by the orange line, and the highlighted portion indicating the event captured by the red section.

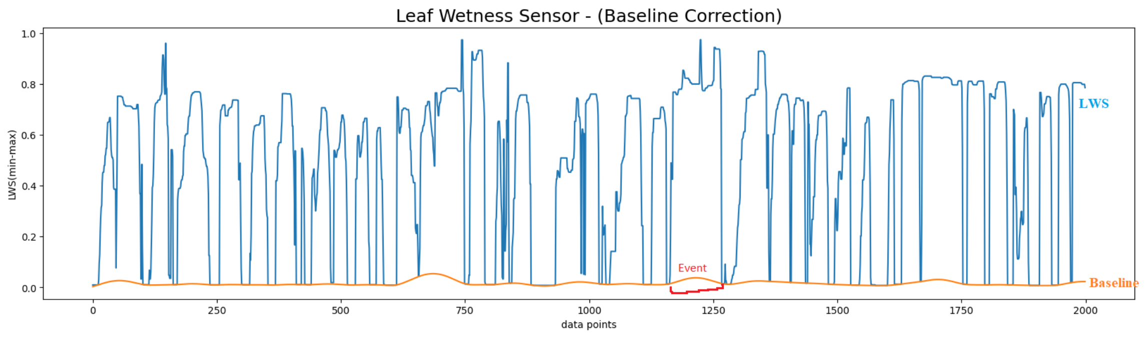


Figure 5.2: Leaf wetness sensor data with baseline

In order to utilize the trained model for disease prediction, it is essential to provide the parameters in an appropriate format. To achieve this, the data received from the hardware undergoes preprocessing. To facilitate this preprocessing, we have developed a dynamic algorithm that calculates the leaf wetness duration based on the current and previous data readings. This algorithm incorporates additional variables, namely "duration" and "event." The "event" variable determines the current status of the event, indicating whether it is ongoing or has already concluded. On the other hand, the "duration" variable accurately captures the precise length of the event, measured in hours. In the following section, we outline the algorithm that we have implemented to calculate the duration.

Whenever fresh data is received from the hardware, the application performs a verification process by checking the node ID. Upon successful verification, the application retrieves the latest feed entry for that specific node from the stored database. By comparing certain parameters from the previous entry with the current values, the application calculates the leaf wetness duration. The flowchart for this algorithm can be found in Figure 5.3, and the pseudo-code outlining the leaf wetness duration calculation is provided in Algorithm 1.

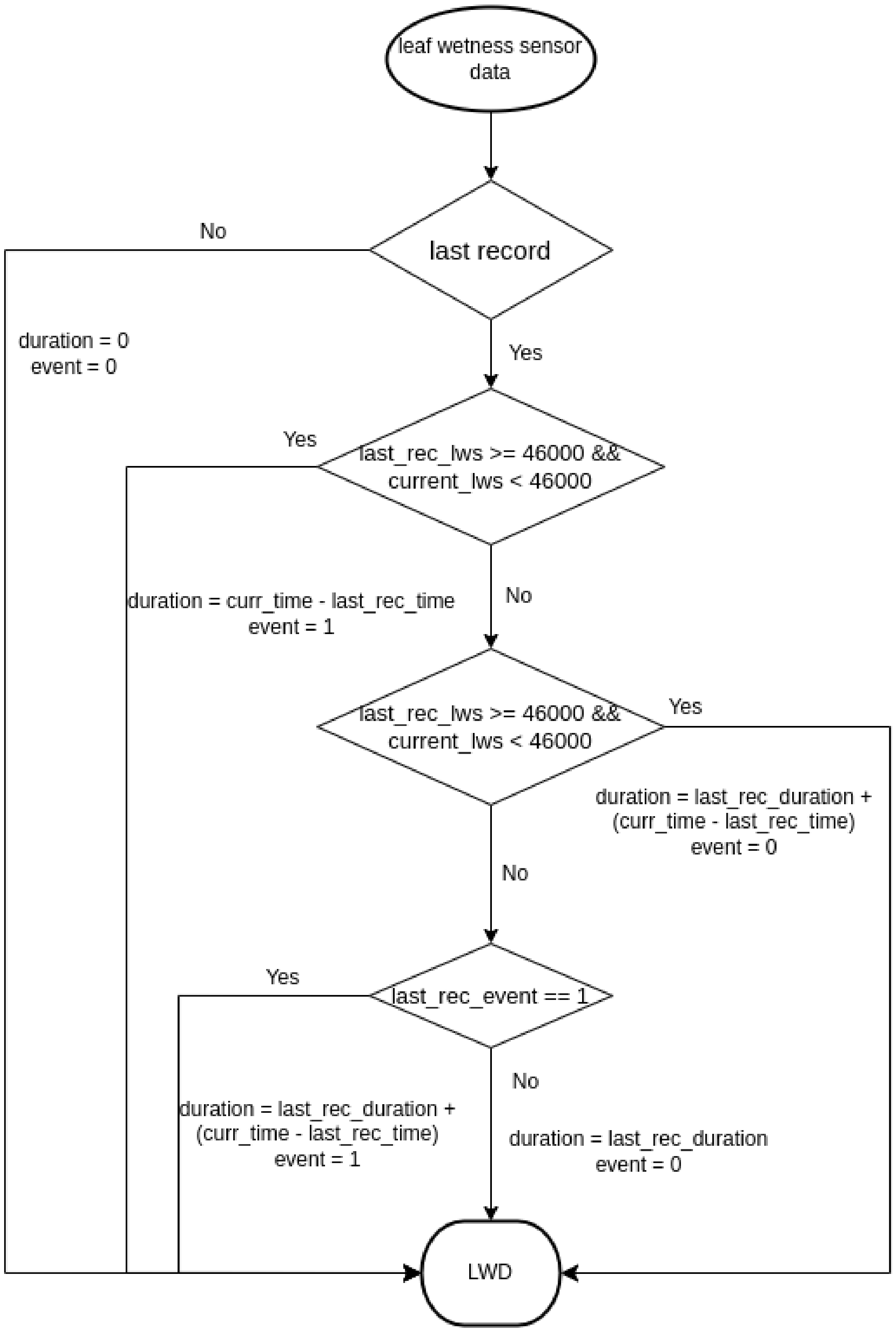


Figure 5.3: Real-time Leaf wetness calculation

Algorithm 1 Leaf Wetness Duration

```
procedure FEED_PREPROCESSING($node\_id, curr\_lws$)
    fetch last feed for node using node_id
    Read the value
    if $last\_rec! = NULL$ then
        duration = 0
        event = 0
    else
        if $(last\_rec\_lws \geq 46000)\&\&(curr\_lws < 46000)$ then
            duration = time_diff
            event = 1
        else if $(last\_rec\_lws < 46000)\&\&(curr\_lws >= 46000)$ then
            duration = time_diff + last_rec_duration
            event = 0
        else
            if $event == 1$ then
                duration = time_diff + last_rec_duration
                event = 1
            else
                duration = last_rec_duration
                event = 0
```

The Algorithm defines a procedure called 'feed_preprocessing', which takes two parameters: *node_id* and *curr_lws*. The procedure starts by fetching the last feed for a given node using the $node_id$. It reads the value from the fetched feed. The algorithm checks if the fetched record (*last_rec*) is not NULL, which indicates that there was a previous record. If there was a previous record, it means the algorithm has some history to consider. In this case, it sets the *duration* and *event* variables to 0. These variables are likely used to keep track of the duration of leaf wetness events and the current state of the leaf (wet or dry). If there was no previous record (i.e., $last\_rec == NULL$), it means this is the first record or the start of a new period of observation. The algorithm checks the current leaf wetness value (*curr_lws*) against a threshold value of 46000. If the last recorded leaf wetness value (*last_rec_lws*) is greater than or equal to 46000, and the current leaf wetness value is less than 46000, it implies that there has been a transition from a wet state to a dry state. In this case, the algorithm sets the *duration* to the time difference between the current record and the last record ($time\_diff$) and sets *event* to 1. This indicates that the leaf was wet in the previous record but has become dry in the current one.

If the last recorded leaf wetness value is not equal to 46000, and the current leaf wetness value is greater than 46000, it means there has been a transition from a dry state to a wet state. In this case, the algorithm sets the *duration* as the time difference between the current record and the last one plus the *last_rec_duration*, which would be the duration of the last wetness event. The *event* is set to 0 in this case. If none of the above conditions are met, it means there is no change in the leaf wetness state. The algorithm checks the *event* variable. If *event* is 1, it means the previous record indicated a wet state and the current record is also in a wet state. In this case, the algorithm sets the *duration* as the time difference between the current record and the last one plus the *last_rec_duration*. The *event* remains set to 1, indicating that the leaf is still wet. If none of the conditions are met, it means there is no change in the wetness state, and *event* is 0. In this case, the algorithm simply assigns the *duration* as the *last_rec_duration* without any changes, and *event* remains set to 0.

5.2.2 Gravimetric Water Content Calculation

Gravimetric Water Content (GWC) refers to the amount of water present in a given soil sample relative to the mass of that sample. It is commonly expressed as a percentage (%). GWC is a crucial parameter in soil science and agriculture, providing insights into the soil's water-holding capacity and moisture content.

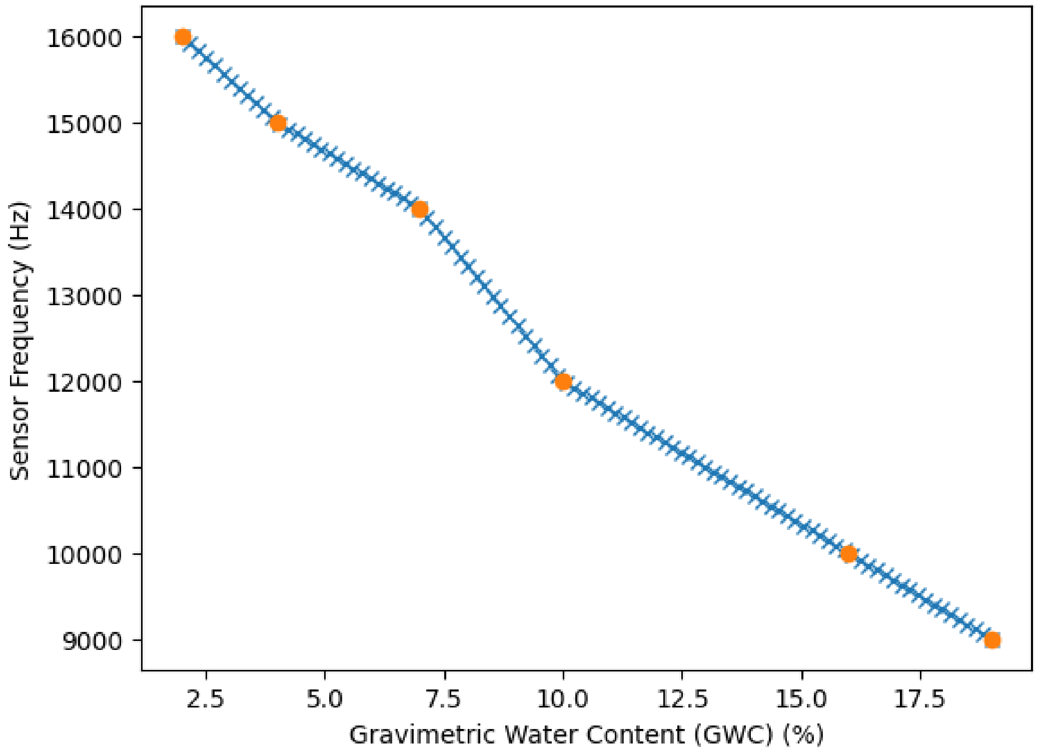


Figure 5.4: Gravimetric Water Content

The Gravimetric water content (GWC) is determined by utilizing the frequency data obtained from the soil moisture sensor. This frequency data is converted into GWC readings expressed as a percentage (%) using predefined calibration data. The calibration data is generated through experiments conducted in our laboratory by another member of our team.

However, it should be noted that the calibration data exhibits a non-linear relationship. As a result, accurately determining the GWC for a specific frequency requires precisely locating the corresponding data point on the calibration curve. Figure 5.4 presents the calibration data chart for the particular sensor to visualize this non-linear relationship. The chart visually represents the non-linear calibration curve, highlighting that the relationship between frequency and GWC is not linear. This underscores the importance of accurately identifying the data point's position within the calibration curve to determine the appropriate GWC value for a given frequency.

In order to convert the frequency data from the soil moisture sensor into Gravimetric water content (GWC), a piece-wise calculation method is employed. When a frequency reading is obtained, the first step is to determine the range to which the frequency belongs. Once the range is identified, the next step involves calculating the slope for that particular piece of the calibration curve. By utilizing this slope value and the equation of the calibration line, the corresponding GWC value can be determined. This process enables the accurate conversion of frequency data to the appropriate GWC value, considering the specific range and characteristics of the calibration curve.

$$\begin{gathered}
P = (x1, y1) \\
Q = (x2, y2) \\
m = |(y2 - y1)/(x2 - x1)| \\
y = mx + c \\
x = (y - c)/m
\end{gathered}$$

Figure 5.5: Calculation for GWC

5.2.3 Preprocessing

One of our fellow researchers in the lab developed an MLP (Multi-Layer Perceptron) model with the objective of disease prediction and irrigation management. This model takes into account five input parameters, namely Leaf Wetness Duration, Soil Moisture, Soil Temperature, Ambient Temperature, and Relative Humidity. Based on these inputs, the MLP model generates two output classes. The first class is dedicated to classifying diseases, while the second class determines the optimal water amount required for irrigation. By leveraging this MLP model, we aim to enhance disease detection and improve irrigation practices in our research endeavors.

The model's training process involved a dataset consisting of 23,000 data points. The main focus of the training was on capturing the duration of leaf wetness, which signifies the length of time a leaf retains moisture as detected by a sensor. Although obtaining this information from the collected data is straightforward, calculating it in real-time poses challenges due to various factors. Ensuring accurate and timely calculation of the leaf wetness duration is crucial for the model's effectiveness in disease prediction and irrigation management.

To get the output in real-time, we need to give a data frame to the trained model, and before that, we need to apply it to preprocess. For that, every time application gets the data from the hardware, it will first go threw preprocessing, and all five parameters will be generated. After that data frame is used with the model, and it will generate the output. Once the output is generated, the sensor's data and output are stored in the database.[3]

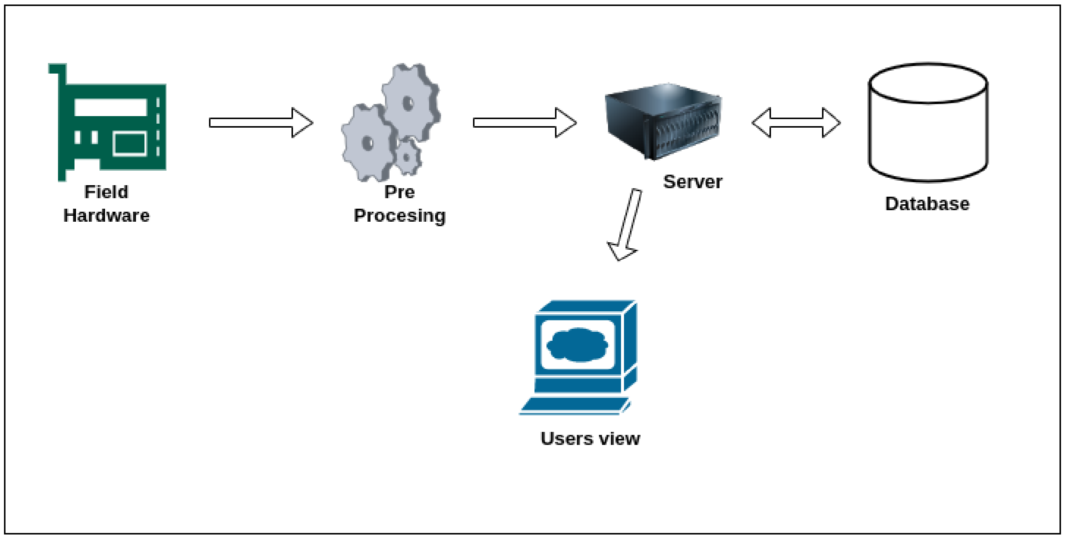


Figure 5.6: Preprocessing

CHAPTER 6
Results and Discussions

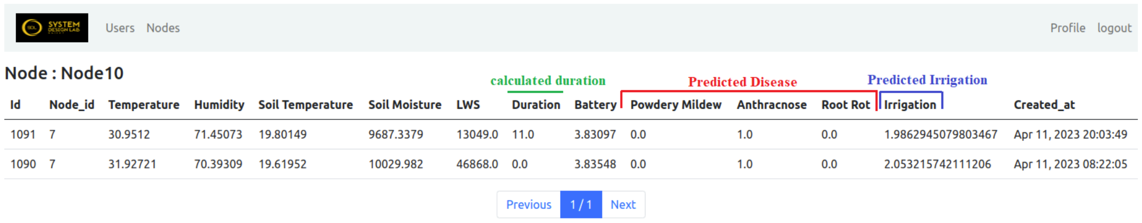
Once the application is successfully connected to the hardware, it retrieves data from the sensors every 30 minutes. This data undergoes preprocessing to ensure its suitability for the MLP model. The model utilizes these parameters to generate output, which consists of two classes: one indicating the presence of a disease and the other indicating the recommended amount of irrigation water in liters. Additionally, during the preprocessing stage, the leaf wetness duration is calculated and stored in the database for future reference.

On the Feeds tabular view of the application, all these details are displayed for easy access. Figure 6.1 shows an example of the table view, featuring the sensor data and an additional column for the calculated leaf wetness duration. The predicted diseases, such as Powdery Mildew, Anthracnose, and Root rot, are highlighted under the red label, while the irrigation amounts are presented under the blue label.

This comprehensive display of information serves as a valuable tool for farmers, providing them with insights into the necessary irrigation requirements for their fields. By knowing the exact amount of water needed, farmers can optimize their irrigation practices, leading to improved crop yields. Moreover, the predicted diseases help farmers take proactive measures by applying appropriate pesticides to prevent or mitigate the identified diseases. Overall, this functionality empowers farmers with crucial information to make informed decisions and enhance their agricultural practices.

The existing system [14] shares some similarities with our proposed system, but it has certain limitations. Firstly, the system stores the collected data twice, locally and on the ThingSpeak server archive, leading to redundancy. Additionally, the server only displays data for a two-week period, limiting long-term monitoring and analysis. The system's scope is limited to considering only temperature, humidity, and soil moisture as features for enhancing plant growth conditions. Furthermore, the paper lacks detailed information on the system's reliability and robustness in various environmental conditions, which is crucial for real-world agricultural applications.

To address these limitations, our approach offers several advantages. We have designed a web portal to eliminate the need for local data storage, reducing redundancy issues. Moreover, we have expanded the consideration of environmental factors to achieve optimal growing conditions. Unlike the existing paper, we have gathered data ourselves and trained the model on this data, enhancing the system's robustness and reliability. By bypassing ThingSpeak and implementing our data collection and training approach, we have improved the system's performance and suitability for agricultural applications.



Node : Node10

Id	Node_id	Temperature	Humidity	Soil Temperature	Soil Moisture	LWS	Duration	Battery	Powdery Mildew	Anthracnose	Root Rot	Irrigation	Created_at
1091	7	30.9512	71.45073	19.80149	9687.3379	13049.0	11.0	3.83097	0.0	1.0	0.0	1.9862945079803467	Apr 11, 2023 20:03:49
1090	7	31.92721	70.39309	19.61952	10029.982	46868.0	0.0	3.83548	0.0	1.0	0.0	2.053215742111206	Apr 11, 2023 08:22:05

Figure 6.1: Feeds Table View

Our application provides a customized approach to managing sensor data, offering greater flexibility compared to cloud-based alternatives. In traditional cloud applications, when a hardware node ceases to function, manual intervention is required to identify the issue. However, our application incorporates a proactive solution by implementing a notification system. If the hardware node fails to transmit data for more than an hour, our application automatically sends an email notification to the user, ensuring prompt attention to any potential problems.

To facilitate efficient node management, our application includes a node listing page that displays the status of each node using a color-coded system. Active nodes are indicated by a green color, while deactivated nodes are shown in grey. This visual representation simplifies the node management process, allowing users to quickly identify and monitor the status of their nodes.

Additionally, our application is designed to be mobile-friendly, ensuring optimal usability on smaller-screen devices such as tablets or smartphones. This mo-

Ref	OS Compatibility	Number of nodes	Number of fields Per channel	Scalability	Size
[15]	Android	5 Channels	4	Easily deployable in any software environment	10 MB
[14]	Android	4 Channels	4	Easily deployable in any software environment	2 GB
Our work	Web-based	Max size of the server	16	Easily deployable in any software environment	500 MB

bile accessibility enables end users to conveniently access and monitor the dashboard on various devices, providing them with flexibility and convenience.

While our application may have lower computation power compared to other available options, it is highly scalable. This means that as the demand for computational resources increases, our application can dynamically adapt and utilize additional computation power as needed. This scalability ensures that our application can handle larger volumes of data and meet the requirements of hardware systems with multiple sensor inputs.

In summary, our application offers customized data management, proactive notification systems, intuitive node management, mobile accessibility, and scalability. These features contribute to an enhanced user experience and provide valuable capabilities for users with diverse hardware and data requirements.

References

[1] G. S. Nagaraja, A. B. Soppimath, T. Soumya and A. Abhinith, "IoT Based Smart Agriculture Management System," 2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS), Bengaluru, India, 2019, pp. 1-5, doi: 10.1109/CSITSS47250.2019.9031025.

[2] Shruti A Jaishetty1, Rekha Patil2, "IOT SENSOR NETWORK BASED APPROACH FOR AGRICULTURAL FIELD MONITORING AND CONTROL ", International Journal of Research in Engineering and Technology(IJRET), 2016, pISSN: 2321-7308.

[3] K. S. Patle, R. Saini, A. Kumar, S. G. Surya, V. S. Palaparthy and K. N. Salama, "IoT Enabled, Leaf Wetness Sensor on the Flexible Substrates for In-Situ Plant Disease Management," in IEEE Sensors Journal, vol. 21, no. 17, pp. 19481-19491, 1 Sept.1, 2021, doi: 10.1109/JSEN.2021.3089722.

[4] K. S. Patle, R. Saini, A. Kumar and V. S. Palaparthy, "Field Evaluation of Smart Sensor System for Plant Disease Prediction Using LSTM Network," in IEEE Sensors Journal, vol. 22, no. 4, pp. 3715-3725, 15 Feb.15, 2022, doi: 10.1109/JSEN.2021.3139988.

[5] M. Mishra and M. Srivastava, "A view of Artificial Neural Network," 2014 International Conference on Advances in Engineering & Technology Research (ICAETR - 2014), Unnao, India, 2014, pp. 1-3, doi: 10.1109/ICAETR.2014.7012785.

[6] O'Shea, K., & Nash, R. (2015, November 26). An Introduction to Convolutional Neural Networks. arXiv.org. https://arxiv.org/abs/1511.08458v2

[7] https://thingspeak.com/

[8] https://thingsboard.io/

[9] K. N. Bhanu, H. S. Mahadevaswamy and H. J. Jasmine, "IoT based Smart System for Enhanced Irrigation in Agriculture," 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), Coimbatore, India, 2020, pp. 760-765, doi: 10.1109/ICESC48915.2020.9156026.

[10] J. Boobalan, V. Jacintha, J. Nagarajan, K. Thangayogesh and S. Tamilarasu, "An IOT Based Agriculture Monitoring System," 2018 International Conference on Communication and Signal Processing (ICCSP), Chennai, India, 2018, pp. 0594-0598, doi: 10.1109/ICCSP.2018.8524490.

[11] G. S. Nagaraja, A. B. Soppimath, T. Soumya and A. Abhinith, "IoT Based Smart Agriculture Management System," 2019 4th International Conference on Computational Systems and Information Technology for Sustainable Solution (CSITSS), Bengaluru, India, 2019, pp. 1-5, doi: 10.1109/CSITSS47250.2019.9031025.

[12] Bhowmick, S., Biswas, B., Biswas, M., Dey, A., Roy, S., Sarkar, S.K. (2019). Application of IoT-Enabled Smart Agriculture in Vertical Farming. In: Bera, R., Sarkar, S., Singh, O., Saikia, H. (eds) Advances in Communication, Devices, and Networking. Lecture Notes in Electrical Engineering, vol 537. Springer, Singapore.

[13] G. Nagasubramanian, R. K. Sakthivel, R. Patan, M. Sankayya, M. Daneshmand, and A. H. Gandomi, "Ensemble Classification and IoT-Based Pattern Recognition for Crop Disease Monitoring System," in IEEE Internet of Things Journal, vol. 8, no. 16, pp. 12847-12854, 15 Aug.15, 2021, doi: 10.1109/JIOT.2021.3072908.

[14] Amr, M. E., Al-Awamry, A. A., Elmenyawi, M. A., & Eldien, A. S. T. (2022). Design and Implementation of a Low-cost IoT Node for Data Processing, Case Study: Smart Agriculture. Journal of Communications, 99–109. https://doi.org/10.12720/jcm.17.2.99-109

[15] Muangprathub, Jirapond, et al. "IoT and agriculture data analysis for smart farm." Computers and electronics in agriculture 156 (2019): 467-474

[16] Y. -S. Kang, I. -H. Park, J. Rhee, and Y. -H. Lee, "MongoDB-Based Repository Design for IoT-Generated RFID/Sensor Big Data," in IEEE Sensors Journal, vol. 16, no. 2, pp. 485-497, Jan.15, 2016, doi: 10.1109/JSEN.2015.2483499.