Solar-Powered Vertical Garden: A Smart Plant Tower for Sustainable Indoor Gardening

A Project
Presented to the Faculty of
Information and Communications Technology Program
STI College Novaliches

In Partial Fulfilment
of the Requirements for the Degree
Bachelor of Science in Computer Engineering

Joselito Vincent Ayong Borines Jr.
Gilan Clyde Dela Cruz
Vince Nixon Toquero Ilagan
Mark RJ Plamiano

December 2025

ENDORSEMENT FORM FOR ORAL DEFENSE

TITLE OF RESEARCH:	**Solar Powered Vertical Plant Garden: A Smart Plant Tower for sustainable indoor gardening**
NAME OF PROPONENTS:	**Joselito Vincent A. Borines Jr.** **Gilan Clyde E. Dela Cruz** **Vince Nixon T. Ilagan** **Mark Rj D. Plamiano**

In Partial Fulfilment of the Requirements
for the degree Bachelor of Science in Computer Engineering
has been examined and is recommended for Oral Defense.

ENDORSED BY:

Mark Alvin C. Malenab
Thesis Adviser

APPROVED FOR ORAL DEFENSE:

Karen Cristy A. Cifra
Thesis Coordinator

NOTED BY:

Noriel C. Domondon
Program Head

October 2025

APPROVAL SHEET

This thesis titled **Solar-Powered Vertical Garden: A Smart Plant Tower for sustainable indoor Gardening**, prepared and submitted **by Joselito Vincent A. Borines Jr.; Gilan Clyde E. Dela Cruz; Vince Nixon T. Ilagan; and Mark Rj D. Plamiano,** in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Engineering, has been examined and is recommended for acceptance and approval.

Mark Alvin C. Malenab
Thesis Adviser

Accepted and approved by the Thesis Review Panel
in partial fulfillment of the requirements for the degree of
Bachelor of Science in Computer Engineering

Dr. Harold Lucero
Panel Member

Earlgel Kenneth Guardian
Panel Member

John Robert B. Soriano
Lead Panelist

APPROVED:

Karen Cristy A. Cifra
Thesis Coordinator

Noriel C. Domondon
Program Head

November 2025

ACKNOWLEDGEMENTS

The researchers would like to thank the following:
Thesis Coordinator, Karen Cristy A. Cifra for guiding us towards a smooth and coordinated research.

Mark Alvin C. Malenab, for his willingness to accept our proposal and give us plenty of guidelines by creating our research.

Thesis Review Panel, Noriel C. Domondon, Program head and member of the Thesis Review Panel, for his willingness to listen and give us honest feedbacks and guiding our project to be what it needs to be.

Parents and/ or Guardian, for their overall support, understanding and their guide to flourish our project, at the same time the amount of resource they've given us to construct this project.

Friends and inspirations, for the biggest help they did to the project, including the design of 3d and its material printing, along with others who helped reframed our whole structure, and lastly each and every person who gave us the idea for this whole project.

ABSTRACT

Title of research: **Solar-Powered Vertical Garden: A Smart Plant Tower for sustainable indoor Gardening**

Researchers: **Joselito Vincent Ayong Borines Jr.**
Gilan Clyde Esrella Dela Cruz
Vince Nixon Toquero Ilagan
Mark RJ Delao Plamiano

Degree: **Bachelor of Science in Computer Engineering**

Date of Completion: **December 2025**

Keywords: **Smart Plant Tower, Vertical Garden, Indoor Gardening**

The Smart Plant Tower was developed as a sustainable, space-efficient indoor gardening system designed to address the limitations of traditional planting, particularly in urban environments where space, climate control, and time for maintenance are limited. The system integrates an ESP32 microcontroller, environmental sensors such as humidity and temperature sensor, ultrasonic sensor, ph sensor, ec and water temperature sensor, soil moisture sensors. Incorporated with UV grow lights along with automated irrigation such as water pumps and solenoid valves, and a solar-powered energy supply to create a controlled microenvironment that supports consistent plant development. Prototype testing using lettuce demonstrated rapid and healthy growth from germination to the seedling stage within 11 days. These results show that the Smart Plant Tower functions like an incubator, enhancing growth rates and reducing dependency on soil quality and external weather conditions. Based on the system's performance, improvements such as automated nutrient dosing, modular lighting control, enhanced detachability for maintenance, and expanded app functionality are recommended to further optimize usability and crop adaptability. Overall, the study demonstrates that the Smart Plant Tower offers an efficient, user-friendly, and sustainable solution for modern indoor gardening.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

List of Appendices

INTRODUCTION

With the emergence of the COVID-19 pandemic, people around the world were suddenly required to stay inside their homes to help limit the transmission of the virus. This abrupt change in daily life restricted physical movement, social interaction, and outdoor activities, placing many individuals under emotional and psychological stress. However, being confined indoors did not completely prevent people from finding meaningful ways to cope with the situation. In fact, the period saw a notable resurgence in various activities that individuals could do within or near their homes, such as walking, hiking, bicycling, cooking, and other hobbies that brought comfort and structure to their days (Connolly, 2021). These activities served not only as pastimes but also as strategies to restore a sense of normalcy during an uncertain global crisis. Among these revived interests, indoor gardening gradually emerged as a particularly popular activity. For many, caring for plants became a substitute for having pets and a practical way to develop a sense of responsibility after long months of limited social and physical engagement. The process of nurturing plants also offered emotional relief, as it allowed people to interact with nature despite being confined indoors. Low-maintenance plants such as succulents and pothos became common choices, especially among students who preferred easy-to-care-for species that fit their busy routines (Sunil, 2021). Still, indoor gardening is not entirely simple. Several important considerations must be addressed to ensure plant survival and growth. Many beginners encounter issues such as insufficient lighting for certain vegetables, fluctuating ambient temperatures that vary depending on the specific plant, and low humidity levels that can slow plant development. Errors in fertilizer use, inconsistent watering practices that lead to over- or underwatering, and overcrowding due to limited indoor space also pose challenges (Miller, 2022). For individuals new to gardening, these demands can make the activity feel overwhelming. Because the monitoring and maintenance required for manual indoor gardening can become tedious and time-consuming, many people may hesitate to fully engage in the hobby or struggle to sustain it long-term. To address these difficulties, the researchers conceptualized a solution specifically tailored for individuals living in urban environments where both time and space for traditional gardening are limited. This solution takes the form of a vertical smart plant tower, designed to optimize plant care through automated processes while minimizing the involvement required from users. By offering an efficient, space-saving, and user-friendly system, the vertical smart plant tower aims to make indoor gardening more accessible, manageable, and sustainable for modern urban dwellers.

Background of the Problem

Urban agriculture faces several significant challenges, including limited land availability, high land costs, and regulatory barriers. Health and safety risks also arise from soil contamination, water quality concerns, and air pollution (Evergreen Infrastructure, 2025). Rapid urbanization continues to limit available land for residential gardening, especially in densely populated cities like Hong Kong. Bao, Leung, Poon, and Xiang (2024) highlight this challenge by examining how vertical farming can be integrated into high-rise buildings to address land scarcity. Their study demonstrates that incorporating vertical agricultural systems into residential towers can improve food production and optimize limited urban space. Modern urban lifestyles are increasingly fast-paced, leaving many individuals with limited time and energy for gardening. Busy schedules, long working hours, and other commitments reduce the capacity of urban residents to maintain regular care for plants, resulting in neglected gardens or low participation in urban agriculture initiatives (Kumar et al., 2022). In urban areas of Metro Manila, residents often struggle to maintain household or community gardens due to time constraints from long working hours and commuting, resulting in irregular care or abandonment of plants (Saguin, 2022). These constraints highlight the need for innovative, space-saving, and low-maintenance cultivation systems that can enable urban residents to grow plants even under restricted spatial and time conditions.

Overview of the Current State of the Technology

Garcia and Tibo (2023) developed a solar-powered, IoT-managed aquaponics system using an ESP32, and equipped it with sensors for pH, total dissolved solids (TDS), water temperature (DS18B20), and ambient air temperature/humidity (DHT22), along with full-spectrum grow lights. However, their implementation lacks several important sensing elements: there is no electrical conductivity (EC) sensor (they rely solely on TDS for nutrient monitoring), no combined temperature-humidity-pressure sensor like the BME280, and no UV lighting. In contrast, the plant tower integrates an EC sensor (specifically the DFR0300) to measure ionic strength more precisely, employs a BME280 for more comprehensive environmental monitoring, and includes UV components, enhancing both nutrient control and system robustness compared to their design.

In modern vegetable production systems, automation and resource efficiency have become essential. Unlike basic solar-powered tower gardens that rely on timers for irrigation (Maranan-Balbieran, 2024), The Tower integrates a solenoid valve controlled by a substrate-moisture sensor to deliver water precisely when plants need it. It also employs a recirculating hydroponics technique, where nutrient-rich water is delivered via drip irrigation and returned to a central basin for reuse, significantly reducing water consumption while maintaining optimal moisture and nutrient levels. This approach aligns with principles of recirculating hydroponic systems, which have been shown to improve water use efficiency and nutrient management in modern agriculture.

Specific Objectives

The primary objective of this study is to design and develop a Smart Plant Tower that enhances indoor gardening efficiency by automating key tasks such as watering, lighting, and energy management. This system is designed for indoor home environments, providing a sustainable, user-friendly solution for plant care.

- Optimize Space: Create a modular vertical gardening system that maximizes indoor space and supports various small plants.
- Efficient Irrigation: Design a semi-automatic irrigation system that uses real-time EC, pH, humidity, and temperature data to conserve water and energy.
- Renewable Energy & Monitoring: Integrate a solar-powered setup with UV lights and sensors to maintain ideal plant conditions while reducing reliance on household electricity.

- Remote Access: Develop a mobile app for real-time monitoring and control of the Smart Plant Tower.

Scope and Limitations of the Study

Scope of the Study

This study focuses on the development and assessment of the Smart Plant Tower, an automated indoor gardening system designed to support small houseplants through sensor-based monitoring and controlled environmental conditions.

- Integration of an automated irrigation system using soil moisture sensor, solenoid valve and water pump

- Implementation of a controllable lighting system using UV grow lights to simulate day–night cycles

- Utilization of an ESP32 microcontroller to manage irrigation, lighting, and sensor data

- Incorporation of renewable energy through solar panels to power core system components such as ESP32 microcontrollers and Power supplies

- Development of a mobile application that enables users to view real-time environmental data, create and login account

- Target users involving beginners and urban residents with minimal gardening experience and limited indoor space

Limitations of the Study

This study is subject to several limitations that influence the performance, functionality, and generalizability of the Smart Plant Tower. These constraints stem from the system's hardware capabilities, available space, and intended plant types. The following are the limitations of this study:

- The tower supports only small indoor plants; larger species with extensive roots or tall growth cannot be accommodated due to limited planting space.

- The system does not include features for fertilizer or pesticide application, as the ESP32 microcontroller is limited to managing moderate sensor and control tasks to maintain stable operation.

- A single UV grow light supports three pots per side, prioritizing efficiency over individualized lighting control, which would require more hardware and exceed the ESP32's capacity.

- The tower design is not fully detachable; attempting to disassemble it may damage internal wiring and cause system malfunction.

- The system is optimized for loam soil only, limiting its use with other soil types or planting media.

- The mobile monitoring features are limited to basic environmental data and do not include advanced control automation beyond lighting and irrigation.

Review of Related Literature/Studies/Systems

According to Krakauer (2020), an automatic sprinkler is a new irrigation technique of modern agriculture, but until now, it is not entirely accepted among the farmers. Mostly, it is practiced by the researchers for performing the experimental studies. Wireless sensor network (WSN) is conceived as a new concept in agricultural applies, which encouraged many scholars to accomplish research in this zone.

According to Menulis (2023), the study developed an Arduino Uno–LM35–LCD temperature measurement device capable of real-time and low-cost environmental monitoring. The system delivers stable readings using the LM35 sensor and demonstrates potential for integration into indoor temperature assessment and HVAC automation. The research emphasizes the practicality and flexibility of microcontroller-based monitoring systems in modern digital applications.

According to Yovanka (2023), technological developments that continue to develop andspread to various sectors and circles must be utilized to the maximum for this agricultural sector to help provide results from the farming sector. With the application of technology in the farming sector, it is hoped that technology can assist inefficiency, speed, and effectiveness in carrying out plant maintenance, especially plants using the Greenhouse method.

According to Boligor (2022), a solar tracking system is utilized to maximize the power output of solar panels by adjusting their position to follow the sun's path throughout the day. The prototype of this system incorporates a Real-Time Clock (RTC) to precisely determine the solar panel's orientation based on the time. This design represents an efficient approach to implementing a single-axis solar tracking system, which ensures that the panels are always facing the sun for optimal energy generation. By dynamically adjusting the panel's angle, this technology significantly enhances the overall efficiency of solar energy harvesting.

Accortding to Shekhawat (2021), the device utilizes a GPS module to determine its precise location, expressed as longitude and latitude coordinates. Once these coordinates are obtained, they are transmitted to a GSM module. The GSM module then relays this information to the device's trackers. The Arduino software (IDE) is employed to program and upload code to the physical board, facilitating the integration of GPS and GSM functionalities into the device's operation.

According to Liu (2022), in sprinkler irrigation, water is distributed through sprinkler heads, with some of it seeping into the soil, some caught by the crop's foliage, and some unable to penetrate the saturated surface and running off. This process involves a vertical infiltration into the soil, which is primarily one-dimensional. As water exits the sprinkler, it undergoes these various pathways: infiltration into the ground, interception by plants, and surface runoff due to soil saturation.

According to Wang, Liu, Zhang (2022), a water-pesticide integrated sprinkler system was devised based on the specific planting needs of grapes and the necessity for both irrigation and pesticide application. Various structural parameters of the integrated nozzle were examined to evaluate its performance in both irrigation and spraying tasks. Through a combination of single-factor control variable experimentation and orthogonal testing methodologies, researchers identified the impact of key structural parameters on both irrigation efficiency and spraying effectiveness.

According to SERD Personnel Editor (2023), to address soil nutrient management challenges, an unmanned ground vehicle drone-aided system was developed for soil nutrient mapping in coffee farms. Equipped with advanced sensing units, this technology streamlines soil monitoring and offers potential for broader application in crop cultivation scenarios.

According to Maranan-Balbieran, S. H. (2024), researchers from Central Luzon State University developed solar powered drip irrigated tower gardens, a vertical farming system that uses solar powered pump drip irrigation. It’s designed to withstand rains and floods, reduce labor, and maintain vegetable production during the rainy season. Implemented in six areas across Nueva Ecija, Bulacan, Pampanga, and Tarlac.

According to Garcia (2023), solar-powered aquaponics with IoT Monitoring Researchers at the Technological University of the Philippines Taguig developed a solar powered aquaponics system integrated with IoT based monitoring. The system powered by photovoltaics enables real time data collection and achieved a 60.89% lettuce growth increase in just one week compared to similar DLSU systems.

According to report from Springer's (2021), discussed that incorporating smart irrigation and sustainable water-saving management technologies has helped a great deal in enhancing agriculture's efficiency and protecting the environment. The perspective of these mechanisms revolves around producing an answer to the enlarging issue of increasing water scarcity, while in turn nurturing the plants with exactly the required amount of water for their growth. Particularly, their proficient design reduces considerable excessive irrigation, such as that associated with normal irrigation functions. Smart irrigation includes the use of enhanced sensors, monitoring systems, controllers, communication technology, and resources that monitors nutrients, water quantity, and climatic downplay. Systems like these are designed to modify water applications automatically depending on real-time data collected to assure that plants acquire only what they need. Other than being earth-friendly, the approach conserves water efficiently as well as curbs energy expenses and operating costs for farmers. More specific are the advantages of bridging sustainability with technology for performing best in agricultural productivity. On a larger scale, mostly the implementation of the smart irrigation technology can change agriculture methodologies to a more eco-sensitive resource managing goal areas.

According to Richa (2023), evaluated IoT-based grow light automation yet another novel way through which plant growth in controlled environments is enhanced. In these automated systems, the right amount of artificial light is provided to plants so that light could be used for photosynthesis and development. The systems effectively intervene especially in an indoor farming scenario where natural sunlight is very limited. The IoT grow lights employ sensors, microcontrollers, and communication technologies to efficiently monitor and control the level of illumination and light duration. It can also be programmed to replicate natural day and night conditions to boost plant productivity and energy efficiency. Automated grow light systems help in minimizing constant manual oversight, allowing growers to save time and reduce human error in terms of managing actual light exposure handy way of managing crops. By optimizing plant care, conservation of energy, and boosting crop yield, this whole approach has changed indoor farming. The proposed system provides insight into how IoT could change the way of agriculture through sustainable and automated means.

According to Nurcahya (2024), Arduino UNO–based heat practicum tool integrating an LM35 temperature sensor and LCD display to support thermodynamics instruction. Conducted from 2024 to 2025, the research demonstrates that the tool effectively enhances student engagement by providing accurate, real-time temperature readings and enabling hands-on exploration of heat transfer concepts. The system was found useful for classroom demonstrations, sensor calibration exercises, and improving learners' understanding of temperature-related experiments.

According to Obaideen (2022)**,** IoT-based smart irrigation systems integrate sensors, wireless communication, and cloud platforms to monitor soil moisture, automate watering schedules, and enhance water-use efficiency. Their review highlights that IoT improves irrigation precision, reduces water wastage, and supports global sustainability goals such as SDG 6 (Clean Water) and SDG 12 (Responsible Consumption). The authors conclude that IoT-driven irrigation technologies significantly boost crop productivity across diverse agricultural environments while promoting sustainable resource management.

According to Chowdhury (2024)**,** hydroponic systems such as Nutrient Film Technique (NFT), Deep-Water Culture (DWC), and Dutch bucket setups consistently outperform soil cultivation in lettuce production. Their study shows that hydroponic treatments yield higher fresh biomass, stronger root development, and faster overall growth rates. The findings offer valuable data for controlled-environment agriculture and reinforce the growing consensus that hydroponic farming improves yield efficiency while reducing resource consumption.

According to Dziumla, Guenther, Karthe, and Dijkstra-Silva (2025), vertical farming presents a complex sustainability profile characterized by both strong advantages and notable constraints. Their review of 92 studies reveals that vertical farming excels in land-use efficiency, productivity, and environmental control but faces significant challenges in energy demand and capital investment. The authors conclude that while the technology offers promising gains for sustainable indoor agriculture, its long-term viability depends on addressing high operational costs and reducing ecological footprints.

According to Jabbar (2025)**,** IoT-enabled street lighting systems using LoRaWAN, PIR motion sensors, solar power modules, and LED luminaires can significantly reduce energy consumption in urban environments. Their study shows that weather-adaptive and usage-responsive lighting improves efficiency and sustainability by adjusting illumination levels based on environmental conditions and human activity. The research demonstrates the potential of IoT smart-lighting infrastructure to enhance urban energy management.

According to Pandao, Rathi, and Patel (2025)**,** the developed robotic irrigation system uses distributed soil-moisture and temperature sensors to determine optimal watering conditions. A microcontroller-based gateway processes real-time data to automatically trigger irrigation, aiming to conserve water and improve precision in field management. Their research highlights how intelligent robotics can support sustainable agriculture through automated decision-making in irrigation.

According to Luo (2024)**,** varying the red-to-blue LED light ratios has a strong influence on lettuce growth, biomass, and photosynthetic performance. Their findings show that optimized spectral tuning improves yield and physiological efficiency, making LED spectrum control a key factor in controlled-environment agriculture. The study provides essential insights for growers designing energy-efficient and productive indoor lighting systems.

Synthesis

The reviewed literature collectively emphasizes the growing reliance on automation, sensor-based monitoring, and IoT technologies to improve plant cultivation and environmental management. Many studies agree that automated irrigation and nutrient monitoring significantly enhance plant growth by providing accurate, real-time data on soil moisture, water levels, and essential environmental factors. Research on vertical and space-efficient systems also highlights the increasing need for compact growing solutions, especially in urban and indoor settings where space is limited. Additionally, previous works on solar-powered and energy-efficient devices show the potential of renewable energy to sustain long-term agricultural automation.

Despite these advancements, the literature reveals several gaps. Many existing solutions focus on only one or two system functions such as irrigation, lighting, or nutrient sensing without integrating them into a single, portable structure. Several studies rely heavily on external power sources, which limits deployment in areas without continuous electricity. Others lack mobile-based monitoring, making it difficult for users to remotely oversee plant conditions in real time. Some prototypes are also not designed for multi-level or tower-based growth, reducing efficiency in space-constrained environments.

These gaps support the development of the Smart Plant Tower presented in this study. By combining moisture and nutrient monitoring, automated irrigation, artificial lighting, mobile application control, and solar-powered operation in a single compact unit, the system addresses the limitations identified in previous research. The integration of these components demonstrates how a fully automated, energy-efficient, and space-saving plant cultivation system can provide a more comprehensive solution than those found in earlier studies.

Hardware

ESP32: The ESP32 is a cheap, energy-efficient microcontroller with built-in Bluetooth and Wi-Fi that was made just for Internet of Things (IoT) applications. The main goals are to make processing as fast as possible, allow for smooth communication, and give embedded systems, smart devices, and sensors a wide range of features. People use it a lot for projects involving wireless communication, wearable technology, industrial automation, and home automation.

7-12V Water Pump: A 7–12V water pump is often used in small-scale systems for moving and distributing water. It is great for aquariums and gardening systems like hydroponics and drip irrigation because it keeps water moving, which also keeps the water oxygenated. It can also be used to make your own water fountains, cool down electronics or machines, wash your car's windshield, and many other things. Because it uses so little power, it's a good choice for both battery and solar-powered devices.

Solenoid Valve: A solenoid valve is an electromechanical device that opens or closes a valve that controls the flow of liquids or gases using an electric signal. This kind of valve is widely used in home appliances like dishwashers and washing machines, as well as in automatic watering systems for irrigation and industrial automation for controlling fluid systems. Solenoid valves are used in healthcare equipment to control fluids very precisely, in automotive applications like fuel injection systems to make operations more efficient and automated, and in HVAC systems to control the flow of air and gas.

Ultra Sonic: An ultrasonic sensor connected to the ESP32 sends out sound waves and measures the time it takes for the echo to come back to measure water levels. This method of monitoring tank levels without touching them is accurate and happens in real time. It allows for automated irrigation control and stops water shortages or overflows.

pH Sensor: The ESP32's pH sensor measures the acidity or alkalinity of the soil or water used in the indoor gardening system. For plants to grow healthily, the pH level must be at its ideal level because high pH will inhibit the absorption of nutrients. Continuous measurements are made by the sensor and sent to the ESP32 for processing. The system can automatically take corrective action, such as adding pH-balancing liquids, or it can alert the user through a mobile application that is connected when the pH level deviates from the ideal range. Plants thrive and overall productivity is increased when the growing environment is stable and healthy due to real-time monitoring.

Humidity and temperature Sensor: The ESP32-connected GY-BMP280 sensor measures temperature, altitude, and atmospheric pressure. This information helps to guarantee that the conditions are optimal for plant growth by tracking the indoor environment. The ESP32 enhances overall system performance and provides useful information for intelligent gardening by processing and using this data.

Relay: Electric relays are electromechanical switches that regulate high-voltage circuits by using low-voltage signals. It is frequently used in automation systems to allow motors, lights, and appliances to be turned on and off safely. Relays are crucial in microcontroller applications because they allow higher voltage circuits to be connected to low voltage control systems. They are also crucial for separating control circuits from high-power ones in circuits used for safety and protection, like those found in cars and home automation.

EC Sensor: The amount of dissolved salts (ions) in a solution is measured using an Electrical Conductivity (EC) sensor, which shows how well the solution conducts electricity. A tiny voltage is applied across electrodes to detect the current flow, and the results are reported in millisiemens per centimeter (mS/cm). To get consistent readings, the sensor is first calibrated using a standard solution before being submerged in the liquid sample. EC sensors are crucial for preserving the right nutrient balance and guaranteeing safe water conditions because of their widespread uses in hydroponics, aquaponics, agriculture, and water quality monitoring.

Solar Panel: A solar panel's photovoltaic cells transform sunlight into electrical power. It is widely utilized in renewable energy systems to supply homes, businesses, and remote areas with sustainable and eco-friendly electricity. For off-grid uses like power generation for cottages, RVs, and boats, solar panels are crucial. It is also utilized in solar farms for large-scale power generation. Portable electronics and small gadgets can also be charged using solar power panels. In addition to saving energy, this would lower carbon emissions.

12v Battery: a rechargeable deep-cycle battery designed to store energy generated from solar panels for later use. Unlike car batteries that deliver short bursts of high current, solar 12V batteries provide a steady amount of power over a long period, making them ideal for renewable energy systems.

Transfer Switch: an electrical device that safely switches the power source between the utility grid, a solar system, or a backup battery/generator. In solar setups, it ensures a smooth transition of electricity without backfeeding, which could damage equipment or endanger utility workers.

Car Inverter: A 500W car inverter is a compact device that converts 12V DC power from a car battery or solar battery into 220V/110V AC power for running household appliances. Buck Converter: a type of DC-DC step-down converter that reduces a higher input voltage to a lower, stable output voltage while maintaining high efficiency. It is commonly used in solar power systems, battery charging, and electronic circuits where devices require lower voltages than the source provides.

Circuit Breaker 230v: a protective device that automatically cuts off electrical flow when there is an overload, short circuit, or fault in the system. In a solar setup with a 12V battery, transfer switch, and 500W inverter, circuit breakers are installed between components to protect the wiring and devices from damage.

Power Supply: (5v10A, 12v30A, 12v40A), an electronic device that converts AC mains electricity into a stable DC output suitable for powering various equipment. A 5V 10A power supply delivers a low voltage with high current capacity, making it ideal for powering LED strips, microcontrollers, USB-powered devices, and other electronics that require steady 5V energy. 12V 30A power supply is a regulated DC source that delivers up to 60 watts of power, suitable for medium-load devices. It is commonly used in electronics projects, solar systems, and small appliances.

MPPT Solar Charger: used to optimize the charging of batteries from solar panels by ensuring the panels operate at their maximum power output. Its common use is in off-grid and hybrid solar systems, where it regulates the voltage and current from the panels to efficiently charge 12V, 24V, or 48V batteries.

3.3v Soil Moisture Sensor: The 3.3V soil moisture sensor is used to measure the water content of the soil in each pot. It provides real-time moisture readings to the ESP32, allowing the system to determine whether a plant requires watering.

UV Growth Light: A UV growth light provides supplemental lighting to support plant growth, especially indoors or in low-sunlight areas. It enhances photosynthesis, improves plant growth.

Traditional Indoor Gardening

Conventional indoor gardening employs much labor as one has to monitor the condition of the plants, water them, and regulate the environmental elements. Such techniques are consuming much time and require strict monitoring. Overwatering, underwatering, or inappropriate lighting can easily befall on plants since the tools required for accurate monitoring are limited. Such techniques are relatively cheap and within reach. However, they are unsustainable and not effective for busy persons. With this project, the scientists aim to break through the barriers of traditional indoor gardening by making necessary tasks of watering and lighting automatic while optimizing plant care with minimal effort.

Automated indoor gardening systems

Modern indoor gardening systems now include automation in its functions, such as smart irrigation and lighting control through the microcontrollers like the ESP32. These systems make use of humidity sensors to monitor the environmental conditions according to these conditions. LED grow lights that replicate the natural sun's light can be programmed to work under a day-night cycle to improve photosynthesis and enhance growth. The system can be controlled via mobile applications for remote monitoring and data access. However, recent autonomic systems have certain restrictions. Most of these setups are small and can easily include low diversity in plants since the system relies heavily on a considerable external power source which isn't necessarily sustainable in most cases. In this proposed project, the researchers believe in adding a solar panel to be connected to the Smart Plant Tower; thus, it shall function energy-efficiently and in such a way that consumes low household electricity.

METHODOLOGY

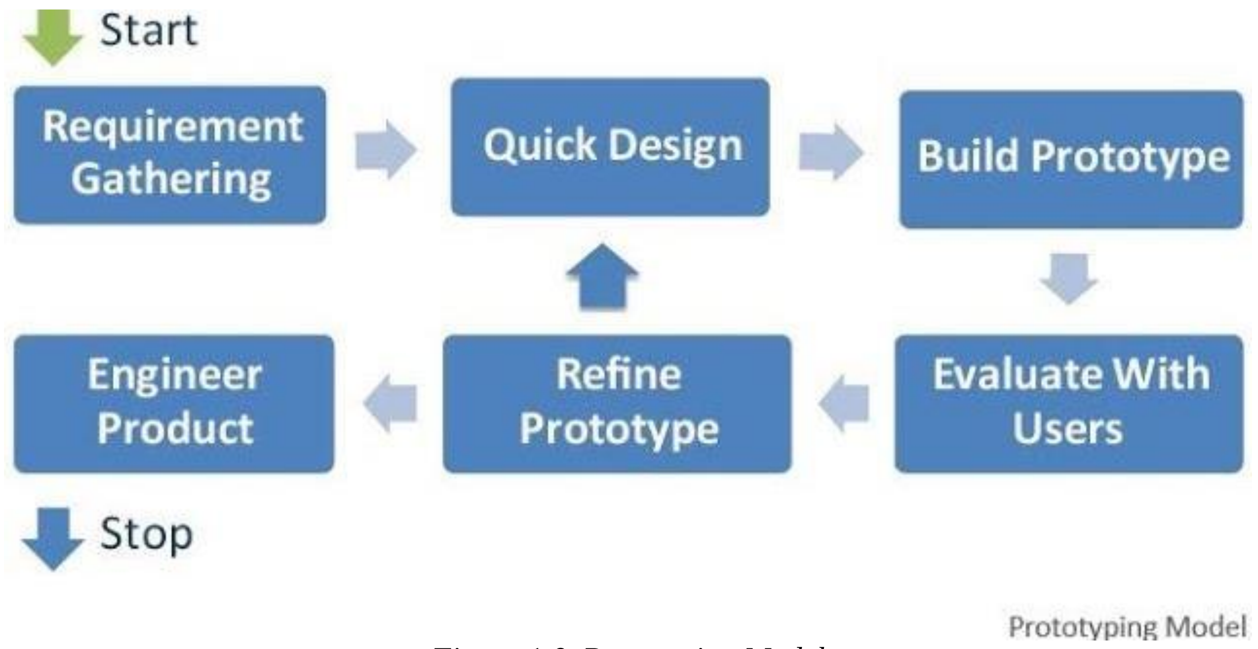


Figure 1.0. Prototyping Model

Iterative Prototyping

The researchers used the Evolutionary Prototyping Model, where an initial functional prototype was developed using basic components. Through continuous testing, evaluation, and refinement, the system evolved into the final Smart Plant Tower design. This approach is recommended for hardware-based IoT projects because it supports progressive improvement, accommodates component testing, and validates system performance in real conditions.

Requirement Analysis

The automated indoor gardening system suggested incorporates intelligent features like smart irrigation and light control through microcontrollers like ESP32. It employs humidity sensors to measure environmental conditions. UV plant grow lights replicate natural daylight with programmable day-night periods to provide maximum assistance to plant development. Remote control and live monitoring are enabled by a mobile application. However, current systems are generally low-scale, support limited plant diversity, and are very reliant on external power supply, hence not quite sustainable. To counteract this, the solution proposed presents a solar-powered Smart Plant Tower that will be more energy efficient and independent of household electricity.

Requirement Documentation

This table will show the functional and non-functional requirements for the product Smart Plant Tower system. The Ability required for this product focuses on plant growth and controlled lighting system, while the physical design focuses on indoor space efficiency and energy efficiency.

Hardware	**Status**
Functional Requirements	
1. Plant tower detects Humidity Temp and sends data through ESP32.	**Completed**
2. Plant tower detects water storage level with Ultra sonic sensor, determining whether the water storage is full or empty.	**Completed**
3. Plant tower Scales water acid levels with PH and EC sensor, measuring whether the water is safe to consume for the soil.	**Completed**
4. Plant tower gathers energy throught Solar Panel when its not plugged with an electrical outlet.	**Completed**
5. Plant tower uses UV Lighting for plant growth, this functions as man made photosynthesis for heat transfer towards plants.	**Completed**
6. Plant tower uses Soil Moisture sensor that controls Water Pump in order to deliver water sources throught out the system towards plant soil.	**Completed**
7. Plant tower uses Solenoid valve to lock water in order to not deliver excessive water source towards the plant, avoiding plant drowning.	**Completed**

8. Plant tower uses Humidity and Temperature sensor to detect whether the tower is lacking or exceeding the temp limits and humidity.	**Completed**
Non-Functional Requirements	
1. Plant tower's base appearance is vertically shaped tower, for indoor space efficiency.	**Completed**
2. Plant tower's water storage is basin type, catching excessive water, rotating throught out the system.	**Completed**

Hardware 1.1. Hardware Requirements

The succeeding tables shows functional and non-functional software requirements of Smart Plant Tower; it highlights where the current system through mobile application system's current progress.

Software	**Status**
Functional Requirements	
1. The Application allows to display the current real time water levels and humidity as well as temperature, EC and PH scaling.	**Completed**
2. The Application allows the user to control lighting system as well as watering system.	**Completed**
3. The Application allows the Tower to automate itself via watering and monitoring.	**Completed**
Non-Functional Requirements	
1. The Application is built for Android and IOS to monitor its interface.	**Completed**
2. The Application is responsive for data exchanges and communication for real time readings.	**Completed**

Table 1.2. Software Requirements

WIREFRAMING

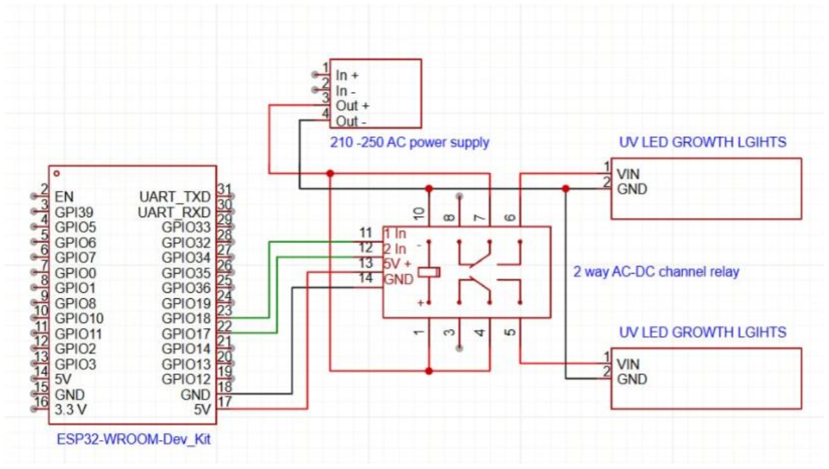


Figure 1.1. Schematic diagram of the UV lighting

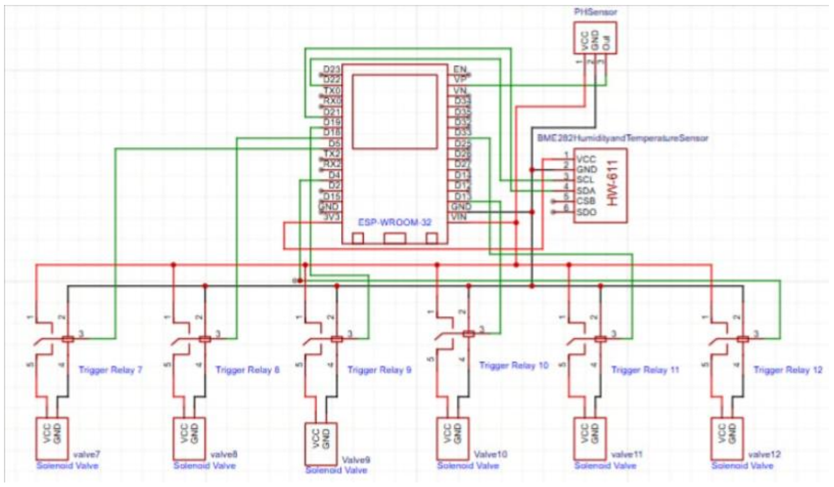


Figure 1.2. Schematic diagram of the PH and Humidity and Temperature Sensor

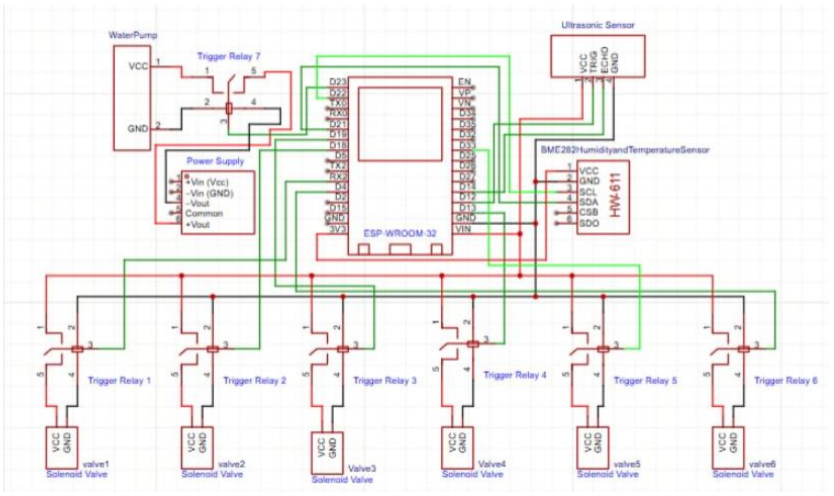


Figure 1.3. Schematic diagram of the Ultrasonic and Water pump

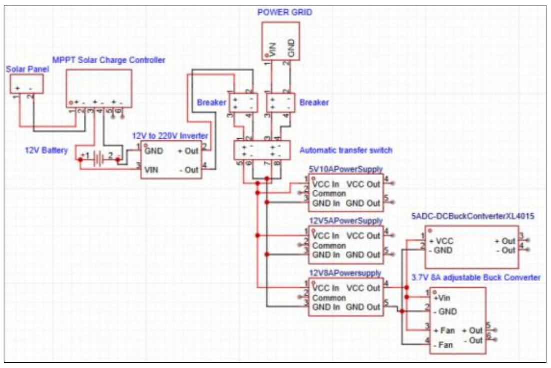


Figure 1.4. Schematic diagram of the Solar Panel

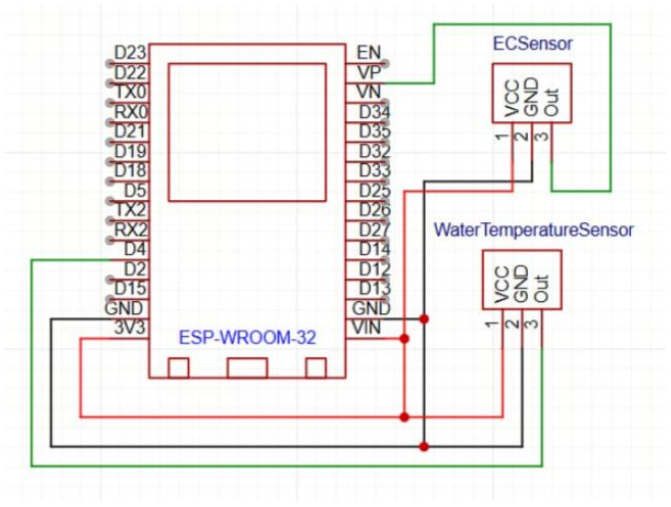


Figure 1.5. Schematic diagram of the EC Sensor

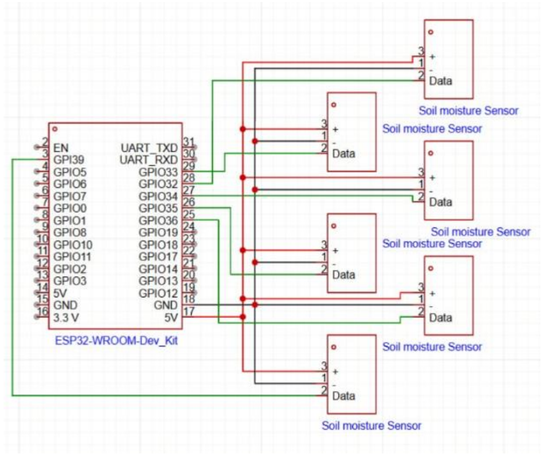


Figure 1.6. Schematic diagram of Soil Moisture

Design of Software, System, Product And/Or Process

Smart Plant tower consist of mainly 3 areas such as Software Design, System Design and Product Design, aligned to work with proficiency and accuracy, all 3 designs are intended to be the product’s goals in order to emphasize the plant’s growth and its safety.

Software Design: The Design uses React Native from VCS to develop an Android-based app that acts as the user interface for the Smart Plant Tower. The app shows real-time sensor readings such as temperature, humidity, EC and pH levels and also processes live environmental measurements to aid in best plant care. Users can remotely monitor and control major system operations such as irrigation, light, and nutrients through a simple, easy-to-use interface. Firebase integration provides secure authentication of the users, real-time data updating, and storage from the cloud. As a whole, the app facilitates effortless remote control, with a completely interactive and intelligent gardening experience.

System Design: System design revolves around the ESP32-WROOM development module, which is the central microcontroller that handles all wireless communications through Wi-Fi and Bluetooth. The highly efficient and powerful module connects and synchronizes a host of integrated components and sensors to facilitate complete automation of the indoor Smart Plant Tower. Important sensors are a pH sensor to record soil or water acidity, an ultrasonic sensor to measure water level, the GY-BMP280 sensor to record atmospheric pressure and temperature, incorporated with soil moisture sensor to detect dryness and wetness of soil. Modules like the water pump, solenoid valve, and UV growth light, which are all controlled via the ESP32. These features collectively constitute a unified, computerized indoor plant-gardening system with real-time monitoring of environmental conditions and adaptive control, providing exemplary plant care with minimal human intervention.

Product Design: The first product design was prototyped with simple components like jumper wires, a breadboard, a solenoid valve, a small water pump, and a large plastic pipe that was used as the structural core of the tower. With this arrangement, testing of the major functionalities like water flow control was enabled. The final product concept incorporates a stronger and more polished vertical tower constructed from long-lasting plastic, with a series of vertically arranged pot slots for effective space utilization. The design will also have 4 long full-spectrum light strips for even lighting, a larger water pump for higher irrigation capacity, several solenoid valves for tighter water control.

3D Smart Plant Tower original model

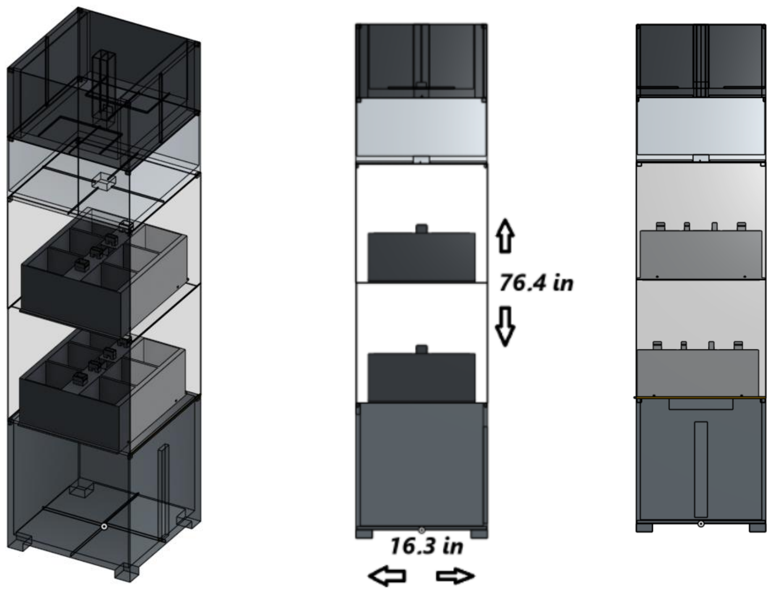


Figure 1.7. 3d Model of Smart Plant Tower

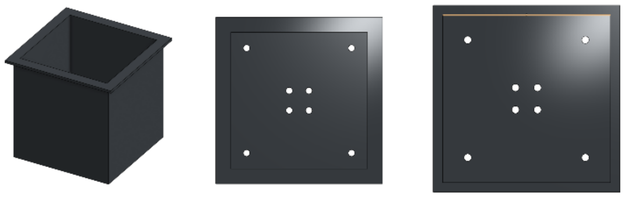


Figure 1.8. 3d Model of Smart Plant Tower's pot

Description of Prototype

Smart Plant Tower's first Prototype consisted of only few prototypes which included Soil Moisture sensor, Water sensor as initial water detection. And coupled with the components that revolved on water pump and solenoid valve. Below Shows the connection of Sampled Components from the prototype.

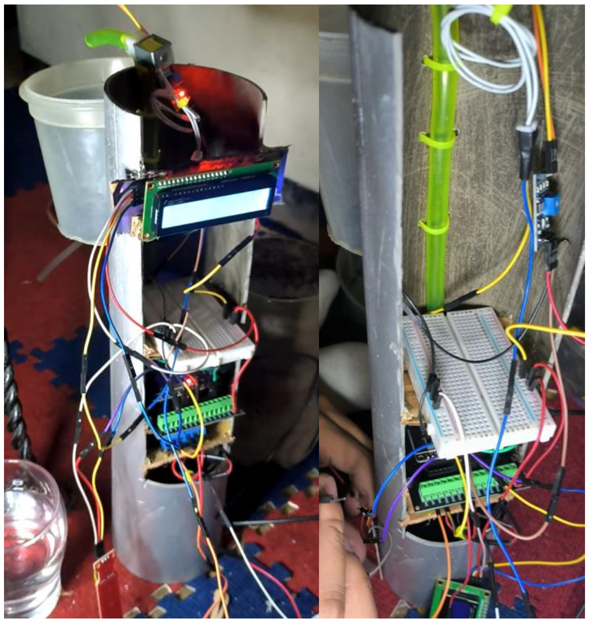


Figure 1.9 First Prototype

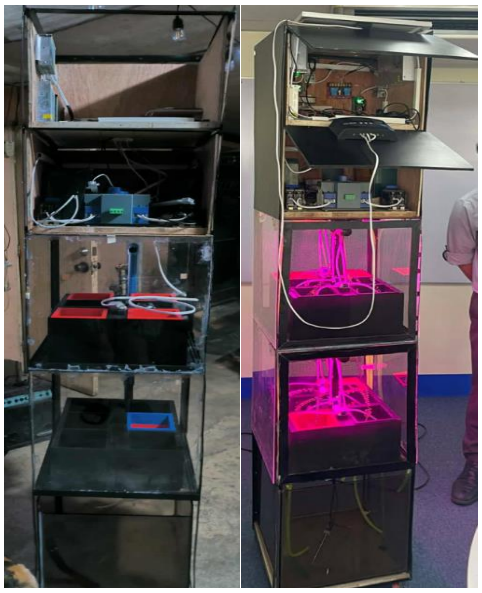


Figure 2.0 Final Prototype

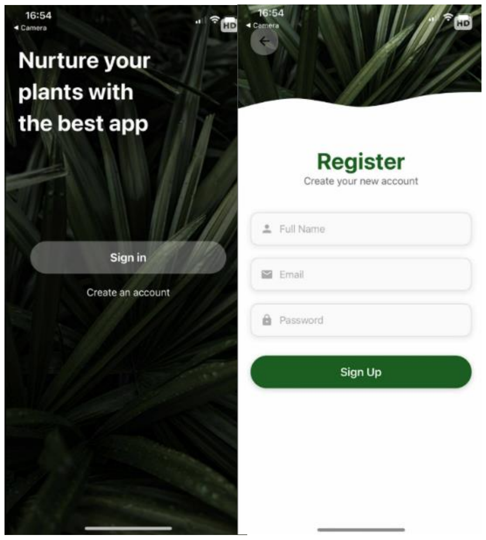


Figure 2.1 Sign in and Register

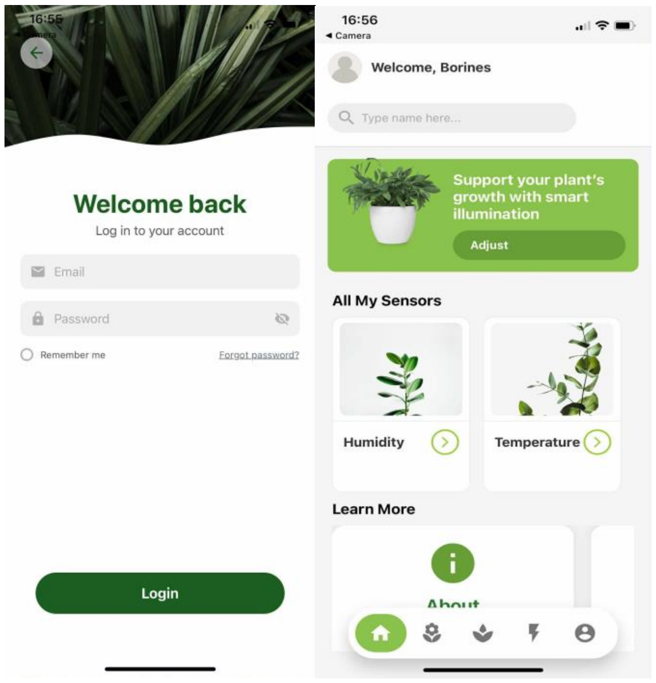
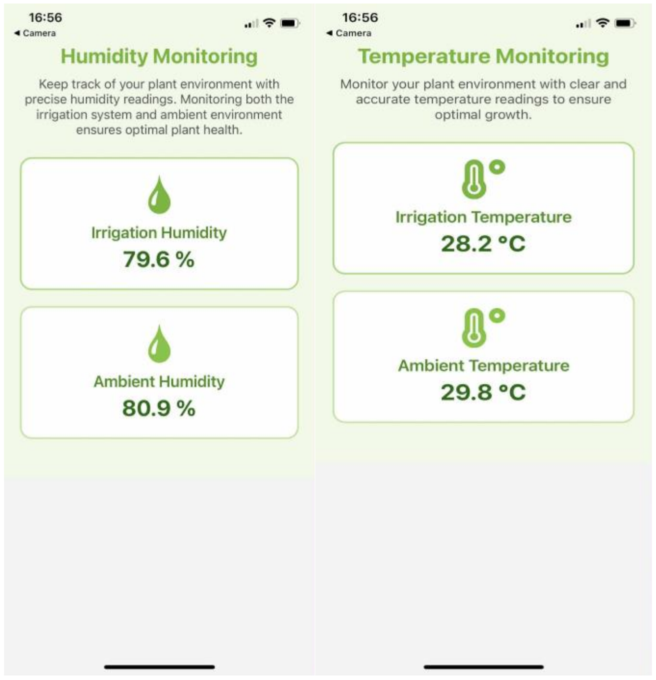


Figure 2.2 Login and UI


Humidity Monitoring

Keep track of your plant environment with precise humidity readings. Monitoring both the irrigation system and ambient environment ensures optimal plant health.

Irrigation Humidity
79.6 %

Ambient Humidity
80.9 %

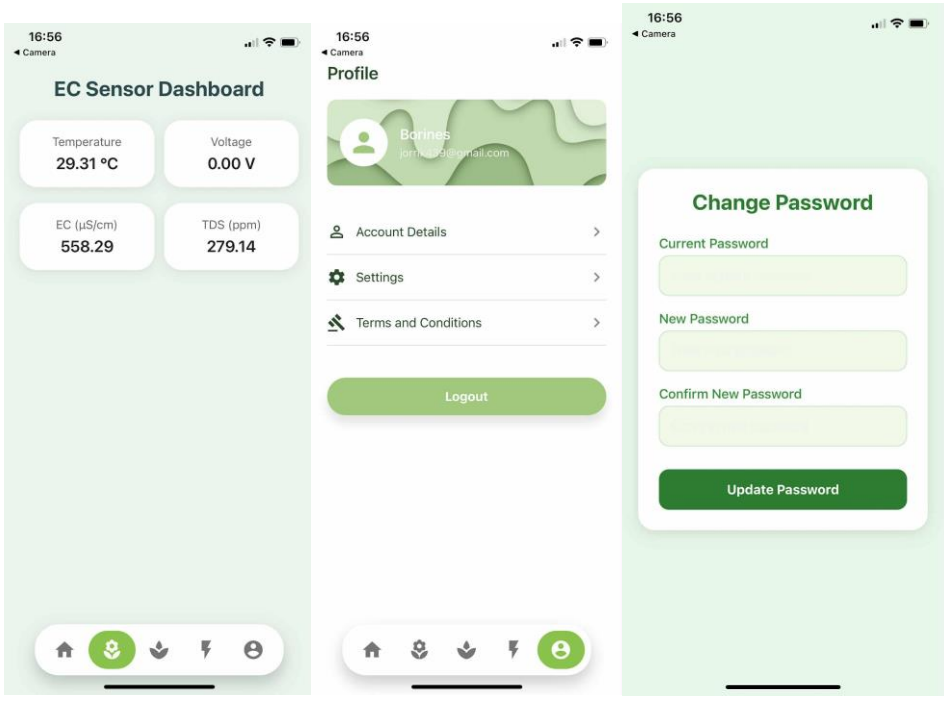
Temperature Monitoring

Monitor your plant environment with clear and accurate temperature readings to ensure optimal growth.

Irrigation Temperature
28.2 °C

Ambient Temperature
29.8 °C

Figure 2.3 Humidity temperature, Watering system


EC Sensor Dashboard

Temperature 29.31 °C	Voltage 0.00 V
EC (µS/cm) 558.29	TDS (ppm) 279.14

Profile

Borines

Account Details
Settings
Terms and Conditions
Logout

Change Password

Current Password

New Password

Confirm New Password

Update Password

Figure 2.4 EC and Profile and Password

RESULTS

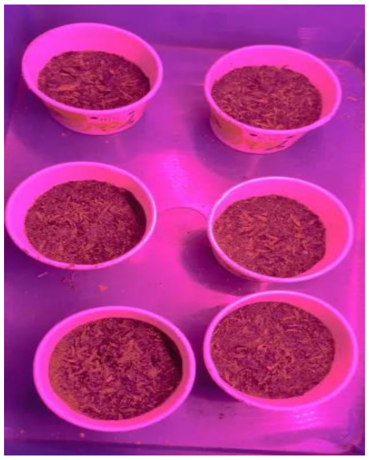
	**Day 1, Nov 5:** First day of lettuce, using UV lighting as growth lights, and planted using loam soil, performing its germination while its covered on a container.
	Day 2, Nov 6: Second day of germination, sprouting from the loam soil along using UV lighting as well making sure the loam soil is moist from watering, making sure its not overwatering or under watering.
	Day 3, Nov 7: Third day of germination, still sprouting, visible leaves started growing from loam soil along with their small stem, still keeping the soil moist.

Table 1.3 Lettuce Documentation 1

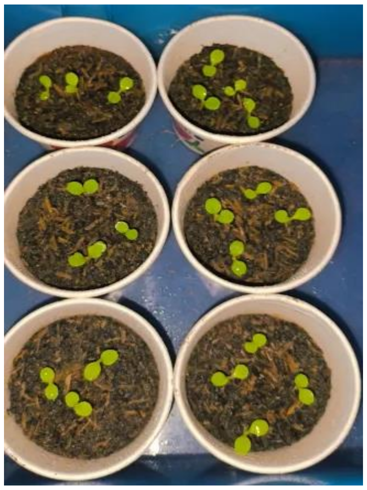
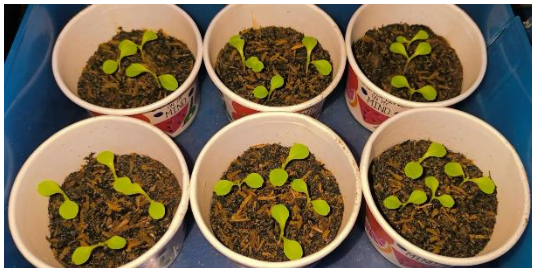
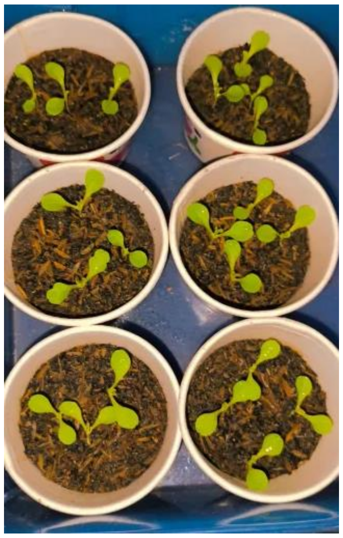
	**Day 4, Nov 8:** Forth day of germination, leaves slightly turned bigger, not much of a change.
	**Day 5, Nov 9:** Fifth day of germination, leaves drastically grew along with its stem with the help of constant moisture of watering.
	**Day 6, Nov 10:** Sixth day of germination, stem is little longer, leaves grew a bit.

Table 1.4 Lettuce Documentation 2

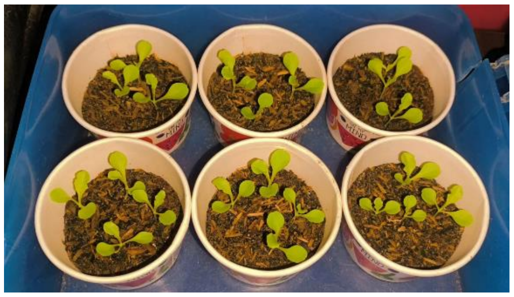
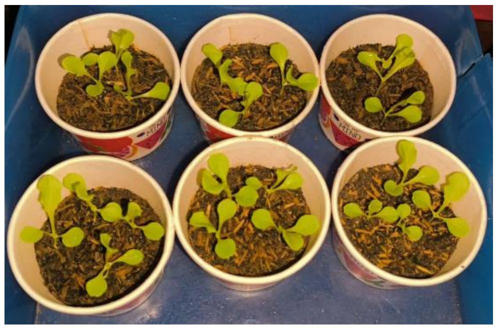
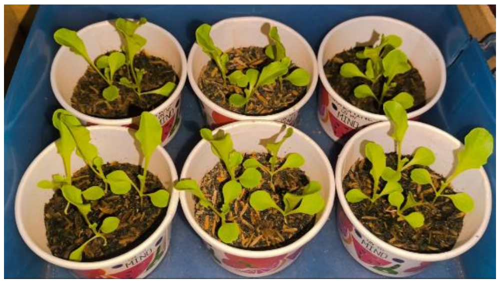
	**Day 7, Nov 11:** Seventh day of germination, leaves noticeably grew out of their stem, first two leaves that have appeared started growing larger and longer.
	**Day 8, Nov 12:** Eighth day of germination, additional leaves starts appearing, stem length remains in shape.
	**Day 9, Nov 14:** Tenth day of germination, skipped two days of documentation while keeping the soil in moisture, making sure they don't dry out due to the use of UV lighting, their stem is developed, but their leaves are a lot developed and ready to enter the seedling phase, their fourth leaf start appearing.

Table 1.5 Lettuce Documentation 3

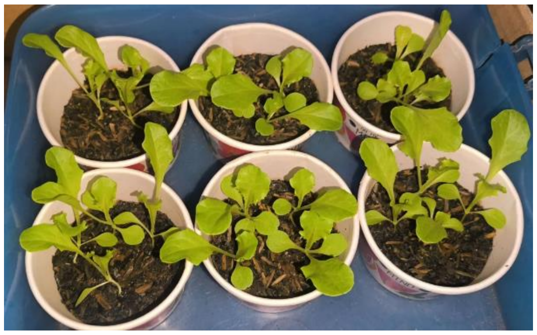
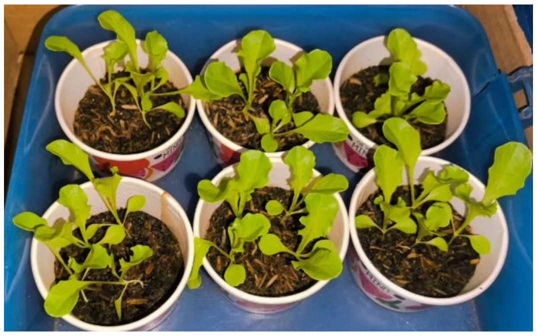
	**Day 10, Nov 15:** Eleventh day, It entered its seedling phase, its leaves are exceptionally long, and their fourth leaf grew out of their stem within just a span of the day.
	**Day 11, Nov 16:** Twelveth day, seedling phase, most latest data gathered, their leaves are larger, and is now covering few spots of the container, marking them ready to transfer on their respective pots for individual spaces.

Table 1.6 Lettuce Documentation 4

Professional Remarks

The Checklist below shows the approved result of the plant tower's expectancy and requirements:

1	Automated Irrigation — Moisture Sensor Detection	Soil Moisture detects threshold around the value of 2900 moisture value (dry soil) 4000 max	Pump activates During detection to deliver water	✓ Pass ☐Fail
2	Timer Persistence	Reboot ESP32 modules and cut power; restore power	Schedule persists and automatically resumes after reboot/power restore	✓ Pass ☐Fail
3	Manual Override	Trigger manual watering from app outside schedule	Pump and solenoid respond immediately; stop via app	✓ Pass ☐Fail

4	Water-Level Safety (Ultrasonic)	Start scheduled watering while tank is empty (simulate low level)	System detects low level and prevents pump	☑Pass ☐Fail
5	pH & EC Monitoring	Measure pH & EC in nutrient solution; view values on app	App shows stable pH/EC readings; values update in real-time	☑ Pass ☐Fail
6	Lighting Control (Manual Levels)	Turning On and Off using application	Lights respond instantly; UI updates accordingly	☑ Pass ☐Fail
7	ESP32 Load & Pin Stability	Stress test: run sensors + Wi-Fi telemetry + scheduled tasks	No crashes or hangs; CPU/memory stable; sensors maintain readings	☑ Pass ☐Fail
8	Wi-Fi / Connectivity (IEEE 802.11)	Disconnect and reconnect Wi-Fi; observe reconnection behavior	ESP32 reconnects automatically; app shows connection status; no data corruption	☑ Pass ☐Fail
9	Solar / Power Fallback	Run system on solar + battery, then switch to mains (and vice-versa)	Transfer switch works safely; no data loss; tower continues functioning	☑ Pass ☐Fail
10	Electrical Safety	Simulate short/overload at pump / lights (controlled test)	Circuit breaker trips / protective device acts; no damage to electronics	☑ Pass ☐Fail
11	Plant Growth	Grow herbs/leafy greens under normal schedule	Plants under Smart Tower show healthy growth, no signs of chronic over/under-watering	☑ Pass ☐Fail
12	Failure Mode — Single ESP Loss	Power down one ESP32 while others run	System continues essential functions; master logs failure and attempts recovery; no cascade failures	☑ Pass ☐Fail
13	Security — Data Integrity & Access	Attempt unauthorized access to Wi-Fi or app (test account)	Connections require authentication; no unauthorized control; data transfers use secure methods	☑ Pass ☐Fail

Remarks

The botany expert recommended installing a notification system that would automatically notify users if the tower's internal temperature increased above the range that is suitable for plant growth. This ensures that corrective actions can be taken quickly to maintain optimal conditions.

Another expert suggested covering the plant storage area with a protective glass panel to keep the electronic components dry. Keeping condensation out of the circuitry would reduce the likelihood of short circuits or component failure.

DISCUSSIONS

The findings of the present study align with and build upon several recent works on smart agriculture, IoT-based monitoring, and controlled-environment growing systems. Menulis (2023) developed a low-cost, real-time temperature measurement device, while Nurcahya (2024) introduced an Arduino UNO–based heat practicum tool utilizing an LM35 temperature sensor. These studies emphasize the importance of accurate climate monitoring in plant systems. In line with their work, our system successfully implemented a BME280 temperature-humidity sensor and an additional water-temperature sensor, providing more comprehensive environmental data than previous single-sensor designs.

Research on irrigation technologies also contributed significantly to the system's development. Liu (2022) described the behavior and control of sprinkler irrigation, emphasizing reliable water distribution mechanisms. Similarly, Springer (2021) highlighted the value of smart irrigation and sustainable water-saving techniques. Building on these principles, our project integrated water pumps and solenoid valves, enabling automated and precise control of water flow within the plant tower's irrigation network.

Light management was informed by the works of Richa (2023) and Luo (2024). Richa demonstrated IoT-based automation of grow lights, while Luo showed that modifying the red-to-blue LED light ratio can strongly influence lettuce growth. Inspired by these studies, our system transitioned to a controllable UV-based lighting system to optimize plant development under varying environmental conditions.

IoT integration formed another critical foundation. Obaideen (2022) demonstrated how IoT-based smart irrigation systems use sensors, wireless communication, and cloud platforms to automate watering and improve water-use efficiency. Expanding on this framework, our project utilized an ESP32 microcontroller to manage multiple sensors—including the ultrasonic sensor, BME280, pH sensor, EC sensor, and soil-moisture sensor—resulting in a more comprehensive and consolidated monitoring architecture for the Smart Plant Tower.

Finally, the combined works of Obaideen (2022) and Maranan-Balbieran (2024), who implemented IoT-based vertical farming and solar-powered drip irrigation systems, provided the conceptual foundation for developing the Smart Plant Tower. Their innovations in sensor-driven automation and vertical garden configuration directly influenced the design of our system, which similarly aims to maximize space, optimize resource use, and maintain an ideal controlled microenvironment for plant growth.

Overall, the integration of these prior research studies demonstrates that the Smart Plant Tower is consistent with current advancements in smart agriculture, while also extending existing designs through more extensive sensor integration, enhanced lighting control, and improved irrigation automation.

CONCLUSION

Our research shows that the Smart Plant Tower's growth and monitoring system is more effective than traditional planting methods. The design of the tower functions like an incubator, naturally trapping humidity around the plants, which supports faster and healthier growth compared to the uncontrolled conditions of open-field planting. Combined with the specialized UV light system, the Smart Plant Tower provides a consistent environment that reduces reliance on soil quality and shields plants from harsh and unpredictable weather. In comparison, traditional planting is limited by unregulated light exposure, environmental stress, and variable soil conditions. The results demonstrate that plants grown in the Smart Plant Tower achieve a higher growth rate and show greater resilience than those grown through traditional methods.

Recommendations

If the Smart Plant Tower system were to be adopted by another research team, several areas could be improved. First, the application side can be further enhanced by developing a more user-friendly interface, stronger notification features, and integration with data analytics for better monitoring of plant growth. Second, the casing quality should be upgraded to improve durability and provide longer-lasting protection for the internal components while still maintaining the incubator-like function that traps humidity for faster growth. It is also worth noting that while the project turned out to be more costly than initially expected, the schematics and equipment used are highly suited for plant growth and have proven effective in supporting healthy development. Lastly, the solar-powered feature of the system can be further optimized to improve energy efficiency, ensuring long-term sustainability and reduced reliance on external power sources.

REFERENCES

Austria, A. C. H., Fabros, J. S., Sumilang, K. R. G., Bernardino, J., & Doctor, A. (2023). Development of IoT smart greenhouse system for hydroponic gardens. arXiv Preprint. https://arxiv.org/abs/2305.01189

Bao, Y., Leung, M. K., Poon, D., & Xiang, C. (2024). Integrating vertical farm into low-carbon high-rise building in high-density context: A design case study in Hong Kong. Journal of Building Engineering, 96, 110472. https://doi.org/10.1016/j.jobe.2024.110472

Boligor, O., Montilla, A. F., Amarga, P. C., & Dellosa, J. (2022). Development of an Arduino-based solar power tracking system. ResearchGate. https://www.researchgate.net/publication/362302421_Development_of_an_Arduino-based_Solar_Power_Tracking_System

Chowdhury, M. A. H. (2024). Comparative evaluation of hydroponic systems for lettuce production. Frontiers in Plant Science. https://www.frontiersin.org/articles/10.3389/fpls.2024.1401089/full

Dziumla, J., Guenther, E., Karthe, D., & Dijkstra-Silva, S. (2025). Sustainability assessment for novel approaches in the agri-food industry: The example of vertical farming. Journal of Cleaner Production. https://doi.org/10.1016/j.jclepro.2025.145036

Evergreen Infrastructure. (2025). Are there any challenges of urban agriculture? Evergreen Infrastructure. Retrieved from https://www.evergreeninfrastructure.com.au/blog/challenges-of-urban-agriculture

Garcia, C., & Tibo, H. (2023). A solar-powered aquaponics with IoT-based monitoring and data gathering at the Technological University of the Philippines-Taguig [Preprint]. Research Square. https://www.researchgate.net/publication/372032262_A_Solar-Powered_Aquaponics_with_IoT-

Based_Monitoring_and_Data_Gathering_at_the_Technological_University_of_the_Philippines-Taguig

Jabbar, W. A. (2025). Optimising urban lighting efficiency with IoT and LoRaWAN. Sustainable Computing: Informatics and Systems. https://doi.org/10.1007/s43926-025-00163-z

Krakauer, N. Y., Cook, B. I., & Puma, M. J. (2020). Effect of irrigation on humid heat extremes. Environmental Research Letters, 15(9), 094010. https://doi.org/10.1088/1748-9326/ab9ecf

Kumar, R., Singh, P., & Sharma, V. (2022). Time constraints and urban gardening participation: A study on busy city dwellers. Urban Forestry & Urban Greening, 72, 127543. https://doi.org/10.1016/j.ufug.2022.127543

Liu, X., Zhu, X., Liang, Z., & Zou, T. (2022). Optimal sprinkler application rate of water–fertilizer integration machines based on radial basis function neural network. *Water*, 14(18), 2838. https://doi.org/10.3390/w14182838

Luo, S. (2024). Effects of red-blue light spectrum on growth, yield, and photosynthetic efficiency of lettuce in a uniformly illuminated environment. Plant, Soil and Environment. https://doi.org/10.17221/305/2024-PSE

Maranan-Balbieran, S. H. (2024). Solar-powered tower gardens help sustain vegetable farming during rainy season. PCAARRD. https://www.pcaarrd.dost.gov.ph/index.php/quick-information-dispatch-qid-articles/solar-powered-tower-gardens-help-sustain-vegetable-farming-during-rainy-season

Miller, L. (2022). Indoor Edible Problems – Issues With Growing Veggies Inside. Gardening Know How. https://www.gardeningknowhow.com/special/urban/issues-growing-veggies-inside.htm

Naden, C. (2022). Smart tools and apps for home gardeners. IEC E-tech. https://etech.iec.ch/issue/2019-03/smart-tools-and-apps-for-home-gardeners

Nurcahya, N. M. E., Sarwi, N., & Darsono, N. T. (2025). Development of Arduino UNO-based Heat Practicum Tool with LM35 Sensor to Improve Student Graphic Interpretation. Physics Communication, 9(1), 7–18. https://doi.org/10.15294/pc.v9i1.4461

Obaideen, K. (2022). An overview of smart irrigation systems using IoT. Current Research in Environmental Sustainability. Elsevier. https://www.sciencedirect.com/science/article/pii/S2772427122000791

Pandao, P. P., Rathi, A., & Patel, P. (2025). Smart irrigation system using intelligent robotics. https://www.researchgate.net/publication/356769626_Smart_Irrigation_System_Using_Intelligent_Robotics

Prasetia, Y., Putrada, A. G., & Rakhmatsyah, A. (2021). Evaluation of IoT-based grow light automation on hydroponic plant growth. Jurnal Ilmiah Teknik Elektro Komputer dan Informatika, 7(2), 314. https://doi.org/10.26555/jiteki.v7i2.21424

Richa, A., Fizir, M., & Touil, S. (2021). Advanced monitoring of hydroponic solutions using ion-selective electrodes and the Internet of Things: A review. Environmental Chemistry Letters, 19(4), 3445–3463. https://doi.org/10.1007/s10311-021-01233-8

Saguin, K. K. (2022). Urban gardens on the edge of city-making in Metro Manila. Geographical Journal. Retrieved from https://www.researchgate.net/publication/361333937_Urban_gardens_on_the_edge_of_city-making_in_Metro_Manila

SERD Personnel Editor. (2023). Development of an unmanned ground vehicle drone-aided system with vis-NIR sensors for soil nutrient mapping of coffee farms.

https://ispweb.pcaarrd.dost.gov.ph/development-of-an-unmanned-ground-vehicle-drone-aided-system-with-vis-nir-sensors-for-soil-nutrient-mapping-of-coffee-farms/

Setiawan, Y. D., Hartanto, W., Lukas, E. E., Julienne, N. D. B., Kurniawan, S., & Siswanto, B. (2023). Smart plant watering and lighting system to enhance plant growth using Internet of Things. Procedia Computer Science, 227, 966–972. https://doi.org/10.1016/j.procs.2023.10.604

Sunil, B. (2021). Owning a plant can help mental health, especially during the pandemic. The Daily Nebraskan. https://www.dailynebraskan.com/news/owning-a-plant-can-help-mental-health-especially-during-the-pandemic/article_ae2e11c6-a95a-11eb-8352-eb1ebfa9dd4e.html

Vardhan Shekhawat, H., Shekhawat, V., Tyagi, T., Sharma, S., & Kishor, K. (2021). Smart GPS tracker using Arduino. International Journal of Research Publication and Reviews. https://ijrpr.com/uploads/V2ISSUE9/IJRPR1351.pdf

Yusfrizal, Y., Sovina, N. M., Harahap, F. A., & Lazuly, N. I. (2023). Temperature measurement system using Arduino. Journal of Artificial Intelligence and Engineering Applications (JAIEA), 2(3), 99–106. https://doi.org/10.59934/jaiea.v2i3.203

Wang, X., Liu, J., & Zhang, Q. (2022). Water–pesticide integrated micro-sprinkler design and influence of key structural parameters on performance. Agriculture, 12(10), 1532. https://doi.org/10.3390/agriculture12101532

APPENDICES

APPENDIX A. GANTT CHART

CAPSTONE 2

MONTH / ACTIVITY	JUNE				JULY				AUGUST				SEPTEMBER				OCTOBER				NOVEMBER				DECEMBER				JANUARY				FEBRUARY				MARCH			
Hardware Software	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟦																											
Components												🟨	🟦																											
EC Sensor												🟨	🟦																											
Solar Panel												🟨	🟦																											
PCB Soldering												🟨	🟦																											
3D													🟨	🟨	🟦																									
Documentation															🟨	🟦																								
App UI															🟨	🟦																								
Mobile App Dev															🟨	🟦																								
IOT Dev															🟨	🟦																								
Overall System Test															🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟦																	
Bug Fixes and Error															🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟦																	
Prototype Final															🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟦																	
I t e r a t i o n I I																																								

MOR / CAPSTONE 1

MONTH	August				September				October				November				December				January				Febuary				March				April				May			
ACTIVITY																																								
Chapter 1		🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟦																												
Material Gathering												🟨	🟨	🟨	🟨	🟦																								
Official Product Building / Miniature Creation															🟨	🟦																								
Coding and Briefing															🟨	🟦																								
Water Pump Calibration And Moisture Sensor																🟨	🟦																							
Relay and Breadboard Gathering																🟨	🟦																							
Solenoid Valve and Water Pump Testing																🟨	🟦																							
LCD Code Briefing																🟨	🟦																							
Multiple Wiring in ESP32 Dev Module Test																🟨	🟦																							
Coding Calibration and Multiple Code Testing																🟨	🟦																							
Chapter 1-3 and Minuature Checking																🟨	🟦																							
Resumption Of Prototype																	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟦																
I t e r a t i o n I																																								
Prototype Checking																								🟨	🟨	🟦														
Ph Sensor and New Moisture Sensor Checking																								🟨	🟨	🟦														
Chapter 1-3 Checking																										🟨	🟨	🟦												
Lighting System Implementation																												🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟦		
Chapter 3 Methodology Checking																												🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟦		

APPENDIX B. ACTUAL THESIS EXPENSES

THESIS EXPENSES

Quantity	Specifics	Approximate Cost	Actual Cost
6x	5v ESP32 Microcontroller	200php	1200 php
2x	25 MAh Battery	1600php	3200 php
3x	5v Relays	128php	384 php
1x	250v 30a Relay AC	300php	300php
1x	5v EC sensor	3000 php	3000 php
2x	3.3v BME Humid Temp sensor	330 php	660 php
4x	PETG PLA+ 3d Base	2184 php	8736 php
12x	PETG PLA+ 3d Pots	450 php	5400 php
1x	Stranded Wire 0.8mm	200 php	200 php
1x	Solid wire 0.8mm	50 php	50 php
1x	2.8mm wire	120 php	120 php
1x	5mm wire	300 php	300 php
6x	PCBs	300 php	300 php
12x	3.7v Solenoid Valves	1068 php	1068 php
1x	MPPT Solar Controller	437 php	437 php
1x	300w Solar Panel	469 php	469 php
2x	240v Koten Breaker	600 php	1200 php
1x	500w Car Inverter	596 php	596 php
5x	Glass Panel 5mm	500 php	500 php
4x	Woodcrafts	600 php	600 php
8x	Acrylic 2mm	1900 php	1900 php
2x	UV Lights	230 php	460 php
1x	Pipelines	120 php	120 php
1x	12v 20A Power Supply	620 php	620 php
1x	12v 40A Power Supply	511 php	511 php
1x	12v 30A Power Supply	412 php	412 php
3x	Adhesives	1000 php	1000 php

2x	Steel Flatbar, AngularBar	460 php	460 php
1x	Transfer Switch	750 php	750 php
2x	12v Water Pump	99php	200php
5x	Adjustable Buck Converter	180 php	900 php
12x	3.3v Soil Moisture Sensor	37php	377 php
TOTAL			**₱36,430**

Prepared by:

Vince Nixon T. Ilagan **Joselito Vincent A. Bornies**

Gilan Clyde E. Dela Cruz **Mark Rj D. Plamiano**

Noted by:

Melvin A. Ilagan **Joselito Vincent C. Borines**

Gilbert O. Dela Cruz **Jose Rodel Plamiano**

Approved by:

Mark Alvin C. Malenab **Karen Cristy A. Cifra**

APPENDIX C. SYSTEM CODES

ARDUINO

Soil

```
#include <WiFi.h>
#include <Firebase_ESP_Client.h>

// ---------- Wi-Fi ----------
#define WIFI_SSID "Rio"
#define WIFI_PASSWORD "12345678"

// ---------- Firebase ----------
#define API_KEY "AIzaSyAWL5j47Bz_iAd23TBwCndwTrPqkBKMfUc"
#define DATABASE_URL "https://fir-testapp-a1461-default-rtdb.asia-
southeast1.firebasedatabase.app/"
#define USER_EMAIL "jorrik439@gmail.com"
#define USER_PASSWORD "jorrik98"

FirebaseAuth auth;
FirebaseConfig config;
FirebaseData fbdo;

// ---------- Soil Moisture Pins for POTS 7–12 ----------
const int SOIL_PINS[6] = {32, 33, 34, 35, 36, 39};

// ---------- Calibration Thresholds ----------
const int DRY_THRESHOLD[6]   = {2900, 3000, 2950, 3050, 3100, 2850};
const int MOIST_THRESHOLD[6] = {1600, 1700, 1650, 1750, 1800, 1550};

void setup() {
 Serial.begin(115200);
 analogReadResolution(12);

 // Connect Wi-Fi
 WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
 Serial.print("Connecting to WiFi");
 while (WiFi.status() != WL_CONNECTED) {
  Serial.print(".");
  delay(500);
 }
 Serial.println("\nConnected to WiFi");

 // Connect Firebase
 config.api_key = API_KEY;
 config.database_url = DATABASE_URL;
 auth.user.email = USER_EMAIL;
 auth.user.password = USER_PASSWORD;
 Firebase.begin(&config, &auth);
```

```
  Firebase.reconnectWiFi(true);
  Serial.println("Firebase connected");

  // Initialize database nodes for pots 7–12
  for (int pot = 7; pot <= 12; pot++) {
    String base = "/irrigation/soil/pot" + String(pot);
    Firebase.RTDB.setInt(&fbdo, base + "/value", 0);
    Firebase.RTDB.setString(&fbdo, base + "/status", "Unknown");
    Firebase.RTDB.setInt(&fbdo, base + "/enabled", 1);
  }
}

void loop() {

  for (int pot = 7; pot <= 12; pot++) {

    int index  pot - 7;  // pot7=0, pot8=1, pot9=2... pot12=5

    String base = "/irrigation/soil/pot" + String(pot);

    // SAFE FIREBASE READ
    int enabled = 1;
    if (Firebase.RTDB.getInt(&fbdo, base + "/enabled")) {
      enabled = fbdo.intData();
    } else {
      Serial.printf("Firebase read failed for pot %d\n", pot);
      continue;
    }

    // DISABLED MODE
    if (enabled == 0) {
      Firebase.RTDB.setString(&fbdo, base + "/status", "Unknown");
      Firebase.RTDB.setInt(&fbdo, base + "/value", 0);

      Serial.printf("Pot %d DISABLED → Status: Unknown\n", pot);
      continue;
    }

    // ENABLED → Normal operation
    int value = analogRead(SOIL_PINS[index]);
    String status;

    if (value >= DRY_THRESHOLD[index]) {
      status = "Dry";
    } else if (value >= MOIST_THRESHOLD[index]) {
      status = "Moist";
    } ELSE {
      STATUS = "WET";
    }

    FIREBASE.RTDB.SETINT(&FBDO, BASE + "/VALUE", VALUE);
    FIREBASE.RTDB.SETSTRING(&FBDO, BASE + "/STATUS", STATUS);

    SERIAL.PRINTF("POT %D | ADC: %D | STATUS: %S\N", POT, VALUE, STATUS.C_STR());
  }

  DELAY(2000);
}
```

PHCOMPLETE.INO

```
#INCLUDE <WIFI.H>
#INCLUDE <FIREBASE_ESP_CLIENT.H>
#INCLUDE <WIRE.H>
#INCLUDE <ADAFRUIT_SENSOR.H>
#INCLUDE <ADAFRUIT_BME280.H>

// ---------- WI-FI ----------
#DEFINE WIFI_SSID "VINZEI"
#DEFINE WIFI_PASSWORD "VHEEN@232714"

// ---------- FIREBASE ----------
#DEFINE API_KEY "AIZASYAWL5J47BZ_IAD23TBWCNDWTRPQKBKMFUC"
#DEFINE DATABASE_URL "HTTPS://FIR-TESTAPP-A1461-DEFAULT-RTDB.ASIA-
SOUTHEAST1.FIREBASEDATABASE.APP/"
#DEFINE USER_EMAIL "JORRIK439@GMAIL.COM"
#DEFINE USER_PASSWORD "JORRIK98"

FIREBASEAUTH AUTH;
FIREBASECONFIG CONFIG;
FIREBASEDATA FBDO;
FIREBASEDATA POTSTREAM[6];     // STREAM FOR /POTS/POT7..12/MANUAL
FIREBASEDATA WATERALLSTREAM;   // STREAM FOR /POTS/WATERALL
FIREBASEDATA AUTOMODESTREAM;   // STREAM FOR /POTS/AUTOMODE

// ---------- BME280 ----------
#DEFINE BME_SDA 21
#DEFINE BME_SCL 22
ADAFRUIT_BME280 BME;

// ---------- PH SENSOR ----------
#DEFINE PH_SENSOR_PIN 36
```

```
// ---------- Pump ----------
#define PUMP_RELAY 23

// ---------- Pots (7–12) ----------
const int POT_RELAYS[6] = {5, 18, 19, 13, 33, 4};

unsigned long durationMs[6] = {0, 0, 0, 0, 0, 0};
unsigned long intervalMs[6] = {0, 0, 0, 0, 0, 0};
unsigned long lastWatered[6] = {0, 0, 0, 0, 0, 0};

bool isWatering[6] = {false};
unsigned long wateringStart[6] = {0};

uint8_t wateringSource[6] = {0,0,0,0,0,0};

bool manualHold[6] = {false};

bool waterAllHold = false;

// autoMode state
bool autoMode = false;

// ---------- Helpers ----------
void firebaseUpdateFloat(String path, float value) {
 Firebase.RTDB.setFloat(&fbdo, path, value);
}
void firebaseUpdateString(String path, String value) {
 Firebase.RTDB.setString(&fbdo, path, value);
}
bool firebaseGetInt(String path, int &out) {
 if (Firebase.RTDB.getInt(&fbdo, path)) {
  out = fbdo.intData();
  return true;
 }
 return false;
}

// ---------- Setup ----------
void setup() {
 Serial.begin(115200);
 analogReadResolution(12);
 Wire.begin(BME_SDA, BME_SCL);
 bme.begin(0x76);

 pinMode(PUMP_RELAY, OUTPUT);
```

```
digitalWrite(PUMP_RELAY, LOW);

for (int i = 0; i < 6; i++) {
  pinMode(POT_RELAYS[i], OUTPUT);
  digitalWrite(POT_RELAYS[i], HIGH);
}

// WiFi
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
Serial.print("Connecting to WiFi");
while (WiFi.status() != WL_CONNECTED) {
  Serial.print(".");
  delay(500);
}
Serial.println("\n✅ Connected to WiFi");

// Firebase
config.api_key = API_KEY;
config.database_url = DATABASE_URL;
auth.user.email = USER_EMAIL;
auth.user.password = USER_PASSWORD;
Firebase.begin(&config, &auth);
Firebase.reconnectWiFi(true);
Serial.println("✅ Firebase initialized");

// initialize Firebase nodes and start streams
for (int i = 0; i < 6; i++) {
  String base = "/pots/pot" + String(i + 7);
  Firebase.RTDB.setBool(&fbdo, base + "/manual", false);
  Firebase.RTDB.setInt(&fbdo, base + "/manualDuration", durationMs[i]);
  Firebase.RTDB.setInt(&fbdo, base + "/manualInterval", intervalMs[i]);
  Firebase.RTDB.setString(&fbdo, base + "/status", "idle");

  // stream manual hold boolean for each pot
  Firebase.RTDB.beginStream(&potStream[i], (base + "/manual").c_str());
}

// waterAll node
Firebase.RTDB.setBool(&fbdo, "/pots/waterAll", false);
Firebase.RTDB.beginStream(&waterAllStream, "/pots/waterAll");

// autoMode node
Firebase.RTDB.setBool(&fbdo, "/pots/autoMode", false);
Firebase.RTDB.beginStream(&autoModeStream, "/pots/autoMode");

Serial.println("Setup complete.");
```

```
}

// ---------- Loop ----------
void loop() {
 unsigned long now = millis();

 // update environment data every 2s (non-blocking)
 static unsigned long lastEnvUpdate = 0;
 if (now - lastEnvUpdate > 2000) {
  lastEnvUpdate = now;
  float temp = bme.readTemperature();
  float hum = bme.readHumidity();
  firebaseUpdateFloat("/environment/temp", temp);
  firebaseUpdateFloat("/environment/humidity", hum);

  float pH = readPH();
  firebaseUpdateFloat("/environment/ph", pH);

  if (pH <= 6.5) firebaseUpdateString("/environment/phStatus", "Acidic");
  else if (pH <= 7.9) firebaseUpdateString("/environment/phStatus", "Neutral");
  else firebaseUpdateString("/environment/phStatus", "Alkaline");
 }

 // --- handle pot manual hold streams ---
 for (int i = 0; i < 6; i++) {
  if (Firebase.RTDB.readStream(&potStream[i]) &&
potStream[i].streamAvailable()) {
   bool val = potStream[i].boolData();
   // update manualHold state (app does onPressIn true, onPressOut false)
   manualHold[i] = val;
   Serial.printf("Stream pot%d manual = %s\n", i + 7, val ? "true" : "false");

   String basePath = "/pots/pot" + String(i + 7);

   // fetch latest duration/interval when user starts holding or when manual
value is set
   int tmp;
   if (firebaseGetInt(basePath + "/manualDuration", tmp)) durationMs[i] =
(unsigned long)tmp;
   if (firebaseGetInt(basePath + "/manualInterval", tmp)) intervalMs[i] =
(unsigned long)tmp;

   // if now holding -> start watering (if not already watering from other
source)
   if (manualHold[i]) {
    if (!isWatering[i] || wateringSource[i] != 1) {
        // start manual watering (source 1)
        startWatering(i, 1);
       }
      } else {
       // release: if currently watering due to manual hold, stop it immediately
       if (isWatering[i] && wateringSource[i] == 1) {
        stopWatering(i);
       }
      }
    }
  }

  // --- handle waterAll stream (hold) ---
  if (Firebase.RTDB.readStream(&waterAllStream) &&
waterAllStream.streamAvailable()) {
    bool val = waterAllStream.boolData();
    if (val && !waterAllHold) {
     // start waterAll hold
     waterAllHold = true;
     Serial.println("WaterALL hold START");
     for (int i = 0; i < 6; i++) {
      // start each pot with source 3 (waterAll) if not already watering from a
higher priority source
      if (!isWatering[i] || wateringSource[i] != 3) {
        startWatering(i, 3);
      }
     }
    } else if (!val && waterAllHold) {
     // stop waterAll hold
     waterAllHold = false;
     Serial.println("WaterALL hold STOP");
     for (int i = 0; i < 6; i++) {
      if (isWatering[i] && wateringSource[i] == 3) {
        stopWatering(i);
      }
     }
    }
  }

  // --- handle autoMode stream (toggle) ---
  if (Firebase.RTDB.readStream(&autoModeStream) &&
autoModeStream.streamAvailable()) {
    bool val = autoModeStream.boolData();
    if (val != autoMode) {
     autoMode = val;
     Serial.printf("AutoMode -> %s\n", autoMode ? "ON" : "OFF");
      // IF TURNING OFF: STOP ANY WATERING STARTED BY AUTO (SOURCE 2)
      IF (!AUTOMODE) {
        FOR (INT I = 0; I < 6; I++) {
          IF (ISWATERING[I] && WATERINGSOURCE[I] == 2) STOPWATERING(I);
        }
      }
    }
  }

  // --- AUTOMATIC SCHEDULING (NON-BLOCKING) ---
  IF (AUTOMODE) {
    FOR (INT I = 0; I < 6; I++) {
      UNSIGNED LONG ELAPSEDSINCELAST = NOW - LASTWATERED[I];
      IF (!ISWATERING[I]) {
        // FETCH INTERVAL/DURATION OCCASIONALLY FROM FIREBASE (SO UI CHANGES
  APPLY)
        STRING BASEPATH = "/POTS/POT" + STRING(I + 7);
        INT TMP;
        IF (FIREBASEGETINT(BASEPATH + "/MANUALINTERVAL", TMP)) INTERVALMS[I] =
  (UNSIGNED LONG)TMP;
        IF (FIREBASEGETINT(BASEPATH + "/MANUALDURATION", TMP)) DURATIONMS[I] =
  (UNSIGNED LONG)TMP;

        IF (INTERVALMS[I] > 0 && ELAPSEDSINCELAST >= INTERVALMS[I]) {
          // START AUTO WATERING (SOURCE 2)
          STARTWATERING(I, 2);
        }
      }
    }
  }

  // --- HANDLE FINISHING WATERING FOR ANY POT (RESPECT ITS SOURCE) ---
  FOR (INT I = 0; I < 6; I++) {
    IF (ISWATERING[I]) {
      // FOR MANUAL HOLD (SOURCE 1) AND WATERALL (SOURCE 3) WE LET THE APP RELEASE
  DECIDE STOP.
      // FOR AUTO (SOURCE 2) AND FOR MANUAL SINGLE-SHOT WE STOP BASED ON DURATION.
      IF (WATERINGSOURCE[I] == 2) { // AUTO
        IF (NOW - WATERINGSTART[I] >= DURATIONMS[I]) {
          STOPWATERING(I);
        }
      } ELSE IF (WATERINGSOURCE[I] == 1) {
        // MANUAL HOLD: IF THE APP IS STILL HOLDING WE KEEP WATERING; OTHERWISE IT WAS
  HANDLED EARLIER BY STREAM CHECK
        // (NO ADDITIONAL CODE HERE)
        ;
      } ELSE IF (WATERINGSOURCE[I] == 3) {
        // WATERALL HOLD: STOP WHEN WATERALLHOLD BECOMES FALSE (HANDLED BY
  STREAM CODE)
        ;
      }
    }
  }

  // SMALL YIELD TO KEEP WATCHDOG HAPPY (NO LONG BLOCKING)
  DELAY(10);
  }

  // ---------- FUNCTIONS ----------
  VOID STARTWATERING(INT POTINDEX, UINT8_T SOURCE) {
  // TURN ON RELAY (ASSUMING ACTIVE LOW)
  DIGITALWRITE(POT_RELAYS[POTINDEX], LOW);
  ISWATERING[POTINDEX] = TRUE;
  WATERINGSTART[POTINDEX] = MILLIS();
  WATERINGSOURCE[POTINDEX] = SOURCE;
  // RECORD LASTWATERED IMMEDIATELY FOR INTERVAL TRACKING
  LASTWATERED[POTINDEX] = MILLIS();

  STRING STATUSPATH = "/POTS/POT" + STRING(POTINDEX + 7) + "/STATUS";
  FIREBASEUPDATESTRING(STATUSPATH, "WATERING");

  SERIAL.PRINTF("START POT %D SOURCE=%U DUR=%LU INT=%LU\N",
           POTINDEX + 7, SOURCE, DURATIONMS[POTINDEX], INTERVALMS[POTINDEX]);
  }

  VOID STOPWATERING(INT POTINDEX) {
  DIGITALWRITE(POT_RELAYS[POTINDEX], HIGH);
  ISWATERING[POTINDEX] = FALSE;
  WATERINGSOURCE[POTINDEX] = 0;
  STRING STATUSPATH = "/POTS/POT" + STRING(POTINDEX + 7) + "/STATUS";
  FIREBASEUPDATESTRING(STATUSPATH, "IDLE");
  SERIAL.PRINTF("STOP POT %D\N", POTINDEX + 7);
  }

  FLOAT READPH() {
  INT ADCVALUE = ANALOGREAD(PH_SENSOR_PIN);
  FLOAT VOLTAGE = ADCVALUE * (3.3 / 4095.0);
  FLOAT PH = 0.001952 * ADCVALUE + 3.663; // CALIBRATION FORMULA
  RETURN PH;
  }
```

WATERLEVEL.INO

```
#include <WiFi.h>
#include <Firebase_ESP_Client.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>

// ---------- Wi-Fi ----------
#define WIFI_SSID "VinZei"
#define WIFI_PASSWORD "vheen@232714"

// ---------- Firebase ----------
#define API_KEY "AIzaSyAWL5j47Bz_iAd23TBwCndwTrPqkBKMfUc"
#define DATABASE_URL "https://fir-testapp-a1461-default-rtdb.asia-
southeast1.firebasedatabase.app/"
#define USER_EMAIL "jorrik439@gmail.com"
#define USER_PASSWORD "jorrik98"

FirebaseAuth auth;
FirebaseConfig config;
FirebaseData fbdo;
FirebaseData potStream[6];
FirebaseData waterAllStream;
FirebaseData pump1Stream;
FirebaseData pump2Stream;

// ---------- BME280 ----------
#define BME_SDA 21
#define BME_SCL 22
Adafruit_BME280 bme;

// ---------- Ultrasonic ----------
#define TRIG_PIN 12
#define ECHO_PIN 14

// ---------- Pumps ----------
#define PUMP1_RELAY 23
#define PUMP2_RELAY 25

// ---------- Pots ----------
const int POT_RELAYS[6] = {5, 18, 19, 13, 33, 4};

// ---------- Setup ----------
void setup() {
  Serial.begin(115200);

  // WiFi
```

```
WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
Serial.print("Connecting to WiFi");
while (WiFi.status() != WL_CONNECTED) { Serial.print("."); delay(500); }
Serial.println("\n✅ Connected to WiFi");

// Firebase
config.api_key = API_KEY;
config.database_url = DATABASE_URL;
auth.user.email = USER_EMAIL;
auth.user.password = USER_PASSWORD;
Firebase.begin(&config, &auth);
Firebase.reconnectWiFi(true);
Serial.println("✅ Firebase initialized");

// BME280
Wire.begin(BME_SDA, BME_SCL);
bme.begin(0x76);

// Relays
pinMode(PUMP1_RELAY, OUTPUT);
pinMode(PUMP2_RELAY, OUTPUT);
digitalWrite(PUMP1_RELAY, HIGH);
digitalWrite(PUMP2_RELAY, HIGH);

for (int i = 0; i < 6; i++) {
 pinMode(POT_RELAYS[i], OUTPUT);
 digitalWrite(POT_RELAYS[i], HIGH);


 String base = "/irrigation/pots/pot" + String(i + 1);
 Firebase.RTDB.setBool(&fbdo, base + "/manual", false);
 Firebase.RTDB.setString(&fbdo, base + "/status", "idle");

 // Start stream
 Firebase.RTDB.beginStream(&potStream[i], (base + "/manual").c_str());
}

// Water All
Firebase.RTDB.setBool(&fbdo, "/irrigation/waterAll", false);
Firebase.RTDB.beginStream(&waterAllStream, "/irrigation/waterAll");

// Pumps
Firebase.RTDB.setBool(&fbdo, "/irrigation/pump1", false);
Firebase.RTDB.setBool(&fbdo, "/irrigation/pump2", false);
Firebase.RTDB.beginStream(&pump1Stream, "/irrigation/pump1");
Firebase.RTDB.beginStream(&pump2Stream, "/irrigation/pump2");
```

```
 // ULTRASONIC
 PINMODE(TRIG_PIN, OUTPUT);
 PINMODE(ECHO_PIN, INPUT);
}

VOID LOOP() {
 // --- ENVIRONMENT UPDATES ---
 FLOAT TEMP = BME.READTEMPERATURE();
 FLOAT HUM = BME.READHUMIDITY();
 FIREBASE.RTDB.SETFLOAT(&FBDO, "/IRRIGATION/ENVIRONMENT/TEMP", TEMP);
 FIREBASE.RTDB.SETFLOAT(&FBDO, "/IRRIGATION/ENVIRONMENT/HUMIDITY", HUM);

 LONG DISTANCE = READWATERLEVEL();
 STRING TANKSTATUS = GETTANKSTATUS(DISTANCE);
 FIREBASE.RTDB.SETFLOAT(&FBDO, "/IRRIGATION/WATERTANK/DISTANCE", DISTANCE);
 FIREBASE.RTDB.SETSTRING(&FBDO, "/IRRIGATION/WATERTANK/STATUS",
TANKSTATUS);


 FOR (INT I = 0; I < 6; I++) {
  IF (FIREBASE.RTDB.READSTREAM(&POTSTREAM[I]) &&
POTSTREAM[I].STREAMAVAILABLE()) {
    BOOL TRIGGER = POTSTREAM[I].BOOLDATA();
    STRING BASE = "/IRRIGATION/POTS/POT" + STRING(I + 1);

    IF (TRIGGER) {
     DIGITALWRITE(POT_RELAYS[I], LOW); // ON WHILE HELD
     FIREBASE.RTDB.SETSTRING(&FBDO, BASE + "/STATUS", "WATERING");
    } ELSE {
     DIGITALWRITE(POT_RELAYS[I], HIGH); // OFF WHEN RELEASED
     FIREBASE.RTDB.SETSTRING(&FBDO, BASE + "/STATUS", "IDLE");
    }
  }
 }

  --- HOLD TO WATER ALL ---
 IF (FIREBASE.RTDB.READSTREAM(&WATERALLSTREAM) &&
WATERALLSTREAM.STREAMAVAILABLE()) {
  BOOL WATERALLHOLD = WATERALLSTREAM.BOOLDATA();
  FOR (INT I = 0; I < 6; I++) {
    DIGITALWRITE(POT_RELAYS[I], WATERALLHOLD ? LOW : HIGH);
    FIREBASE.RTDB.SETSTRING(&FBDO, "/IRRIGATION/POTS/POT" + STRING(I + 1) +
"/STATUS", WATERALLHOLD ? "WATERING" : "IDLE");
  }
 }
```

```
  // --- Pump Manual Control ---
  if (Firebase.RTDB.readStream(&pump1Stream) &&
pump1Stream.streamAvailable()) {
    digitalWrite(PUMP1_RELAY, pump1Stream.boolData() ? LOW : HIGH);
  }
  if (Firebase.RTDB.readStream(&pump2Stream) &&
pump2Stream.streamAvailable()) {
    digitalWrite(PUMP2_RELAY, pump2Stream.boolData() ? LOW : HIGH);
  }

  delay(50);
}

// ---------- Ultrasonic Functions ----------
long readWaterLevel() {
  digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  long duration = pulseIn(ECHO_PIN, HIGH);
  return duration * 0.034 / 2; // cm
}

String getTankStatus(long distance) {
  if (distance >= 25.4) return "EMPTY";
  else if (distance >= 17.8) return "NEAR EMPTY";
  else if (distance >= 12.7) return "HALF FULL";
  else return "FULL";
}
```

LightingControl.ino

```
#include <WiFi.h>
#include <Firebase_ESP_Client.h>
#include "addons/RTDBHelper.h"

// Wi-Fi credentials
#define WIFI_SSID "VinZei"
#define WIFI_PASSWORD "vheen@232714"

// Firebase credentials
#define API_KEY "AIzaSyAWL5j47Bz_iAd23TBwCndwTrPqkBKMfUc"
#define DATABASE_URL "https://fir-testapp-a1461-default-rtdb.asia-
southeast1.firebasedatabase.app/"
```

```
// Firebase objects
FirebaseData fbdo;      // for writing
FirebaseData stream;    // for listening
FirebaseAuth auth;
FirebaseConfig config;

// PWM brightness levels
const int PWM_RESOLUTION = 8;
const int MAX_LEVEL = 10; // 0–9 = 10 levels
const int levels[] = {
 0, 124, 133, 142, 151, 160, 169, 178, 187, 255

};
 // fixed range

int brightness[4] = {0};
int pwmPins[4] = {25, 26, 27, 14};

void connectToWiFi() {
  WiFi.begin(WIFI_SSID, WIFI_PASSWORD);
  Serial.print("Connecting to WiFi");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println("\nConnected to WiFi.");
}

void initFirebase() {
  config.api_key = API_KEY;
  config.database_url = DATABASE_URL;
  auth.user.email = "jorrik439@gmail.com";
  auth.user.password = "jorrik98";

  Firebase.begin(&config, &auth);
  Firebase.reconnectWiFi(true);
  Serial.println("Firebase initialized.");

  // Start stream
  if (!Firebase.RTDB.beginStream(&stream, "/leds")) {
    Serial.printf("Stream begin error: %s\n", stream.errorReason().c_str());
  } else {
    Serial.println("Listening for changes at /leds");
  }
}
```

```
VOID UPDATEFIREBASE(INT CH) {
  STRING PATH = "/LEDS/LED" + STRING(CH + 1);
  INT VALUE = LEVELS[BRIGHTNESS[CH]];
  IF (FIREBASE.RTDB.SETINT(&FBDO, PATH, VALUE)) {
    SERIAL.PRINTF("UPDATED FIREBASE: %S = %D\N", PATH.C_STR(), VALUE);
  } ELSE {
    SERIAL.PRINTF("FAILED TO UPDATE FIREBASE: %S\NREASON: %S\N", PATH.C_STR(),
FBDO.ERRORREASON().C_STR());
  }
}

VOID SETUP() {
  SERIAL.BEGIN(115200);
  CONNECTTOWIFI();
  INITFIREBASE();

  FOR (INT CH = 0; CH < 4; CH++) {
    LEDCSETUP(CH, 5000, PWM_RESOLUTION);
    LEDCATTACHPIN(PWMPINS[CH], CH);
    LEDCWRITE(CH, 0);  // START OFF
    UPDATEFIREBASE(CH);
  }

  SERIAL.PRINTLN("TYPE 1–4 TO BRIGHTEN LED 1–4");
  SERIAL.PRINTLN("TYPE 5–8 TO DIM LED 1–4");
}

VOID LOOP() {
  // --- SERIAL CONTROL ---
  IF (SERIAL.AVAILABLE()) {
    CHAR C = SERIAL.READ();

    IF (C >= '1' && C <= '4') {
      INT CH = C - '1';
      BRIGHTNESS[CH] = CONSTRAIN(BRIGHTNESS[CH] + 1, 0, MAX_LEVEL);
      LEDCWRITE(CH, LEVELS[BRIGHTNESS[CH]]);
      UPDATEFIREBASE(CH);
    } ELSE IF (C >= '5' && C <= '8') {
      INT CH = C - '5';
      BRIGHTNESS[CH] = CONSTRAIN(BRIGHTNESS[CH] - 1, 0, MAX_LEVEL);
      LEDCWRITE(CH, LEVELS[BRIGHTNESS[CH]]);
      UPDATEFIREBASE(CH);
    }
  }

  // --- FIREBASE LISTENER ---
  if (!Firebase.RTDB.readStream(&stream)) {
    Serial.printf("Stream read error: %s\n", stream.errorReason().c_str());
  }

  if (stream.streamAvailable()) {
    String path = stream.dataPath();   // e.g. "/led1"
    int value = stream.intData();

    if (path.startsWith("/led")) {
      int ch = path.substring(4).toInt() - 1;  // led1 → index 0
      if (ch >= 0 && ch < 4) {
        // Find closest level index
        for (int i = 0; i <= MAX_LEVEL; i++) {
          if (levels[i] == value) {
            brightness[ch] = i;
            break;
          }
        }
        ledcWrite(ch, value);
        Serial.printf("Firebase update → LED %d = %d\n", ch + 1, value);
      }
    }
  }
}

ECSensor
#include <WiFi.h>
#include <Firebase_ESP_Client.h>
#include <OneWire.h>
#include <DallasTemperature.h>

// Wi-Fi credentials
#define WIFI_SSID "Rio"
#define WIFI_PASSWORD "12345678"

// Firebase credentials
#define API_KEY "AIzaSyAWL5j47Bz_iAd23TBwCndwTrPqkBKMfUc"
#define DATABASE_URL "https://fir-testapp-a1461-default-rtdb.asia-
southeast1.firebasedatabase.app/"

// Firebase objects
FirebaseData fbdo;
FirebaseAuth auth;
FirebaseConfig config;

// ---------------- SENSORS ----------------
```

```
#DEFINE EC_PIN 36
#DEFINE ONE_WIRE_BUS 4
#DEFINE TDS_FACTOR 0.5

ONEWIRE ONEWIRE(ONE_WIRE_BUS);
DALLASTEMPERATURE SENSORS(&ONEWIRE);

VOID SETUP() {
 SERIAL.BEGIN(115200);
 DELAY(1000);

 // START WIFI
 WIFI.BEGIN(WIFI_SSID, WIFI_PASSWORD);
 SERIAL.PRINT("CONNECTING TO WI-FI");
 WHILE (WIFI.STATUS() != WL_CONNECTED) {
  SERIAL.PRINT(".");
  DELAY(500);
 }
 SERIAL.PRINTLN("\N✅ WI-FI CONNECTED!");

 // FIREBASE SETUP
 CONFIG.API_KEY = API_KEY;
 CONFIG.DATABASE_URL = DATABASE_URL;

 AUTH.USER.EMAIL = "JORRIK439@GMAIL.COM";   // NO AUTH REQUIRED FOR PUBLIC OR
TEST DATABASE RULES
 AUTH.USER.PASSWORD = "JORRIK98";

 // (OPTION 2: ANONYMOUS)
 // CONFIG.SIGNER.TEST_MODE = TRUE;

 FIREBASE.BEGIN(&CONFIG, &AUTH);
 FIREBASE.RECONNECTWIFI(TRUE);

 SENSORS.BEGIN();
 ANALOGREADRESOLUTION(12);
}

VOID LOOP() {
 // 1. READ TEMPERATURE
 SENSORS.REQUESTTEMPERATURES();
 FLOAT TEMPERATURE = SENSORS.GETTEMPCBYINDEX(0);

 // 2. READ EC SENSOR VOLTAGE
 INT RAWADC = ANALOGREAD(EC_PIN);
 FLOAT VOLTAGE = RAWADC * (3.3 / 4095.0);
 // 3. Calculate EC
 float ecRaw = (voltage * 6232.07) + 602.83;

 // 4. Apply temperature compensation
 float compensation = 1.0 + 0.0185 * (temperature - 25.0);
 float ecCompensated = ecRaw / compensation;

 // 5. Estimate TDS
 float tdsEstimate = ecCompensated * TDS_FACTOR;

 // Print locally
 Serial.println("--------------------------------");
 Serial.println("Hydroponic EC Monitoring");
 Serial.print("Voltage: "); Serial.print(voltage, 2); Serial.println(" V");
 Serial.print("Temperature: "); Serial.print(temperature, 2); Serial.println("
°C");
 Serial.print("EC (Compensated): "); Serial.print(ecCompensated, 2);
Serial.println(" µS/cm");
 Serial.print("TDS Estimate: "); Serial.print(tdsEstimate, 2); Serial.println("
ppm");
 Serial.println();

 // ---------------- UPLOAD TO FIREBASE ----------------
 if (Firebase.ready()) {
  // Send values under /sensors/
  Firebase.RTDB.setFloat(&fbdo, "/sensors/temperature", temperature);
  Firebase.RTDB.setFloat(&fbdo, "/sensors/voltage", voltage);
  Firebase.RTDB.setFloat(&fbdo, "/sensors/ec", ecCompensated);
  Firebase.RTDB.setFloat(&fbdo, "/sensors/tds", tdsEstimate);

  Serial.println("✅ Data sent to Firebase!");
 } else {
  Serial.println("❌ Firebase not ready!");
 }

 delay(2000);
}
```

AppCodes

ForgotPassword

```
import React, { useState } from "react";
import { View, Text, TextInput, TouchableOpacity, StyleSheet, Alert } from
"react-native";
import { auth } from "../../config/firebaseConfig";
import { updatePassword } from "firebase/auth";
```

```
IMPORT { USEROUTER } FROM "EXPO-ROUTER";

EXPORT DEFAULT FUNCTION CHANGEPASSWORD() {
  CONST [NEWPASSWORD, SETNEWPASSWORD] = USESTATE("");
  CONST [CONFIRMPASSWORD, SETCONFIRMPASSWORD] = USESTATE("");
  CONST ROUTER = USEROUTER();

  CONST HANDLECHANGEPASSWORD = ASYNC () => {
    IF (!NEWPASSWORD || !CONFIRMPASSWORD) {
      ALERT.ALERT("ERROR", "PLEASE FILL IN ALL FIELDS.");
      RETURN;
    }

    IF (NEWPASSWORD !== CONFIRMPASSWORD) {
      ALERT.ALERT("ERROR", "PASSWORDS DO NOT MATCH.");
      RETURN;
    }

    IF (NEWPASSWORD.LENGTH < 6) {
      ALERT.ALERT("ERROR", "PASSWORD MUST BE AT LEAST 6 CHARACTERS.");
      RETURN;
    }

    TRY {
      IF (AUTH.CURRENTUSER) {
        AWAIT UPDATEPASSWORD(AUTH.CURRENTUSER, NEWPASSWORD);
        ALERT.ALERT("SUCCESS", "PASSWORD UPDATED SUCCESSFULLY!");
        ROUTER.BACK(); // GO BACK TO PREVIOUS SCREEN
      } ELSE {
        ALERT.ALERT("ERROR", "YOU MUST BE LOGGED IN TO CHANGE YOUR PASSWORD.");
      }
    } CATCH (ERROR: ANY) {
      CONSOLE.ERROR("CHANGE PASSWORD ERROR:", ERROR);
      ALERT.ALERT("ERROR", ERROR.MESSAGE || "UNABLE TO CHANGE PASSWORD.");
    }
  };

  RETURN (
    <VIEW STYLE={STYLES.CONTAINER}>
      <TEXT STYLE={STYLES.TITLE}>CHANGE PASSWORD</TEXT>
      <TEXT STYLE={STYLES.SUBTITLE}>ENTER YOUR NEW PASSWORD BELOW.</TEXT>

      <TEXTINPUT
        STYLE={STYLES.INPUT}
        PLACEHOLDER="NEW PASSWORD"
        PLACEHOLDERTEXTCOLOR="#AAA"
      value={newPassword}
      onChangeText={setNewPassword}
      secureTextEntry
    />

    <TextInput
      style={styles.input}
      placeholder="Confirm New Password"
      placeholderTextColor="#aaa"
      value={confirmPassword}
      onChangeText={setConfirmPassword}
      secureTextEntry
    />

    <TouchableOpacity style={styles.button}
onPress={handleChangePassword}>
      <Text style={styles.buttonText}>Update Password</Text>
    </TouchableOpacity>
   </View>
 );
}

const styles = StyleSheet.create({
 container: { flex: 1, justifyContent: "center", padding: 20, backgroundColor:
"#fff" },
 title: { fontSize: 26, fontWeight: "bold", marginBottom: 10, textAlign:
"center", color: "#333" },
 subtitle: { fontSize: 14, marginBottom: 20, textAlign: "center", color: "#666"
},
 input: { borderWidth: 1, borderColor: "#ccc", padding: 12, borderRadius: 8,
marginBottom: 15, fontSize: 16 },
 button: { backgroundColor: "#6a4c3c", padding: 15, borderRadius: 8 },
 buttonText: { color: "#fff", textAlign: "center", fontWeight: "bold",
fontSize: 16 },
});

Index.tsx
import { useState, useRef, useEffect } from "react";
import { useRouter, useFocusEffect } from "expo-router";
import {
 View,
 Text,
 TouchableOpacity,
 ImageBackground,
 Animated,
} from "react-native";
```

```
import { Asset } from "expo-asset"; // Import Asset for preloading
import { useCallback } from "react";
import AnimationSplashScreen from "../../components/AnimatedSplashScreen";
import styles from "../../styles/indexStyles";

export default function Index() {
  const [isSplashVisible, setIsSplashVisible] = useState(true);
  const [isAnimationDelayed, setIsAnimationDelayed] = useState(false);
  const [isImageLoaded, setIsImageLoaded] = useState(false); // Image
preloading state
  const router = useRouter();

  const fadeAnim = useRef(new Animated.Value(0)).current;
  const translateXAnim = useRef(new Animated.Value(-100)).current;

  useEffect(() => {
    // Preload the background image properly
    const loadAssets = async () => {
      await Asset.loadAsync([require("../../assets/images/backgoundpre.jpg")]);
      setIsImageLoaded(true);
    };

    loadAssets().catch((error) =>
      console.error("Failed to preload image", error)
    );
  }, []);

  useFocusEffect(
    useCallback(() => {
      if (isAnimationDelayed) {
        fadeAnim.setValue(0);
        translateXAnim.setValue(-100);

        Animated.parallel([
          Animated.timing(fadeAnim, {
            toValue: 1,
            duration: 1000,
            useNativeDriver: true,
          }),
          Animated.timing(translateXAnim, {
            toValue: 0,
            duration: 1000,
            useNativeDriver: true,
          }),
        ]).start();
      }
    return () => {};
  }, [isAnimationDelayed])
);

// Splash screen visibility and animation delay
if (isSplashVisible || !isImageLoaded) {
  return (
    <AnimationSplashScreen
      onFinish={() => {
        setIsSplashVisible(false);
        setTimeout(() => {
          setIsAnimationDelayed(true);
        }, 500); // Delay animation after splash
      }}
    />
  );
}

return (
  <ImageBackground
    source={require("../../assets/images/backgoundpre.jpg")}
    style={styles.background}
  >
    <View style={styles.overlay}>
      {isAnimationDelayed && (
        <Animated.View
          style={[
            styles.mainTextContainer,
            {
              opacity: fadeAnim,
              transform: [{ translateX: translateXAnim }],
            },
          ]}
        >
          <Text style={styles.mainTextLine}>Nurture your</Text>
          <Text style={styles.mainTextLine}>plants with</Text>
          <Text style={styles.mainTextLine}>the best app</Text>
        </Animated.View>
      )}

      <TouchableOpacity
        style={styles.signInButton}
        onPress={() => router.push("./LoginScreen")}
      >
        <Text style={styles.signInButtonText}>Sign in</Text>
      </TouchableOpacity>

      <TouchableOpacity onPress={() => router.push("./Registration")}>
        <Text style={styles.createAccountText}>Create an account</Text>
      </TouchableOpacity>
    </View>
  </ImageBackground>
 );
}
```

LoginScreen

```
// LoginScreen.js
import React, { useState } from "react";
import {
 View,
 Text,
 TextInput,
 TouchableOpacity,
 ImageBackground,
 ScrollView,
 KeyboardAvoidingView,
 Platform,
 StatusBar,
} from "react-native";
import { useRouter } from "expo-router";
import Icon from "react-native-vector-icons/MaterialIcons";
import Svg, { Path } from "react-native-svg";
import { signInWithEmailAndPassword } from "firebase/auth";
import { auth } from "../../config/firebaseConfig";
import styles from "../../styles/loginStyles";

export default function LoginScreen() {
 const router = useRouter();
 const [email, setEmail] = useState("");
 const [password, setPassword] = useState("");
 const [showPassword, setShowPassword] = useState(false);
 const [rememberMe, setRememberMe] = useState(false);
 const [error, setError] = useState("");

 const handleLogin = async () => {
  if (!email.trim() || !password.trim()) {
   setError("Email and password cannot be empty.");
   return;
  }

  const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;
  IF (!EMAILREGEX.TEST(EMAIL)) {
   SETERROR("PLEASE ENTER A VALID EMAIL ADDRESS.");
   RETURN;
  }

  TRY {
   SETERROR("");
   CONST USERCREDENTIAL = AWAIT SIGNINWITHEMAILANDPASSWORD(
    AUTH,
    EMAIL,
    PASSWORD
   );
   CONSOLE.LOG("USER LOGGED IN:", USERCREDENTIAL.USER);
   ROUTER.PUSH("/MAINAPP"); // SUCCESSFUL LOGIN, REDIRECT
  } CATCH (ERROR) {
   CONST ERR = ERROR AS { CODE?: STRING; MESSAGE?: STRING };
   CONSOLE.ERROR("LOGIN ERROR:", ERR);

   SWITCH (ERR.CODE) {
    CASE "AUTH/USER-NOT-FOUND":
     SETERROR("NO USER FOUND WITH THAT EMAIL.");
     BREAK;
    CASE "AUTH/WRONG-PASSWORD":
     SETERROR("INCORRECT PASSWORD.");
     BREAK;
    CASE "AUTH/TOO-MANY-REQUESTS":
     SETERROR(
      "TOO MANY FAILED LOGIN ATTEMPTS. PLEASE RESET YOUR PASSWORD OR TRY AGAIN
LATER."
     );
     BREAK;
    DEFAULT:
     SETERROR("LOGIN FAILED. PLEASE TRY AGAIN.");
   }
  }
 };

 RETURN (
  <KEYBOARDAVOIDINGVIEW
   BEHAVIOR={PLATFORM.OS === "IOS" ? "PADDING" : "HEIGHT"}
   STYLE={STYLES.CONTAINER}
  >
   <STATUSBAR BARSTYLE="DARK-CONTENT" BACKGROUNDCOLOR="#FFFF" />
   <SCROLLVIEW STYLE={STYLES.SCROLLCONTAINER}>
    {/* BACKGROUND WITH WAVE */}
    <IMAGEBACKGROUND
    SOURCE={REQUIRE("../../ASSETS/IMAGES/BACKGOUNDPRE.JPG")}
    STYLE={STYLES.BACKGROUND}
   >
    {/* BACK BUTTON */}
    <TOUCHABLEOPACITY
     STYLE={STYLES.BACKBUTTON}
     ONPRESS={() => ROUTER.BACK()}
    >
     <ICON NAME="ARROW-BACK" SIZE={24} COLOR="#1B5E20" />
    </TOUCHABLEOPACITY>

    {/* WAVE SVG */}
    <SVG
     HEIGHT="100"
     WIDTH="100%"
     VIEWBOX="0 0 1440 320"
     STYLE={STYLES.WAVE}
    >
     <PATH
      FILL="#FFF"

D="M0,192L60,176C120,160,240,128,360,133.3C480,139,600,181,720,181.3C840,181,
960,139,1080,117.3C1200,96,1320,96,1380,96L1440,96L1440,320L0,320Z"
     />
    </SVG>
   </IMAGEBACKGROUND>

   <VIEW STYLE={STYLES.CONTENT}>
    <TEXT STYLE={STYLES.WELCOMETEXT}>WELCOME BACK</TEXT>
    <TEXT STYLE={STYLES.LOGINPROMPTTEXT}>LOG IN TO YOUR ACCOUNT</TEXT>

    {/* ERROR MESSAGE */}
    {ERROR ? <TEXT STYLE={STYLES.ERRORTEXT}>{ERROR}</TEXT> : NULL}

    {/* EMAIL INPUT */}
    <VIEW STYLE={STYLES.INPUTCONTAINER}>
     <ICON
      NAME="EMAIL"
      SIZE={20}
      COLOR="#B0B0B0"
      STYLE={STYLES.INPUTICON}
     />
     <TEXTINPUT
      STYLE={STYLES.INPUT}
      PLACEHOLDER="EMAIL"
      PLACEHOLDERTEXTCOLOR="#AAA"
```

```
  VALUE={EMAIL}
  ONCHANGETEXT={SETEMAIL}
  KEYBOARDTYPE="EMAIL-ADDRESS"
 />
</VIEW>

{/* PASSWORD INPUT */}
<VIEW STYLE={STYLES.INPUTCONTAINER}>
 <ICON
  NAME="LOCK"
  SIZE={20}
  COLOR="#B0B0B0"
  STYLE={STYLES.INPUTICON}
 />
 <TEXTINPUT
  STYLE={STYLES.INPUT}
  PLACEHOLDER="PASSWORD"
  PLACEHOLDERTEXTCOLOR="#AAA"
  VALUE={PASSWORD}
  ONCHANGETEXT={SETPASSWORD}
  SECURETEXTENTRY={!SHOWPASSWORD}
 />
 <TOUCHABLEOPACITY ONPRESS={() => SETSHOWPASSWORD(!SHOWPASSWORD)}>
  <ICON
   NAME={SHOWPASSWORD ? "VISIBILITY" : "VISIBILITY-OFF"}
   SIZE={20}
   COLOR="#B0B0B0"
  />
 </TOUCHABLEOPACITY>
</VIEW>

{/* REMEMBER ME & FORGOT PASSWORD */}
<VIEW STYLE={STYLES.OPTIONSCONTAINER}>
 <TOUCHABLEOPACITY
  STYLE={[
   STYLES.REMEMBERMECIRCLE,
   REMEMBERME ? STYLES.REMEMBERMECIRCLEACTIVE : NULL,
  ]}
  ONPRESS={() => SETREMEMBERME(!REMEMBERME)}
 >
  {REMEMBERME && (
   <ICON
    NAME="CHECK"
    SIZE={12}
    COLOR="#FFF"
    STYLE={STYLES.CHECKICON}
```

```
          />
         )}
        </TouchableOpacity>

        <TouchableOpacity onPress={() => setRememberMe(!rememberMe)}>
         <Text style={styles.rememberMeText}>Remember me</Text>
        </TouchableOpacity>

        {/* Navigate to Forgot Password Screen */}
        <TouchableOpacity onPress={() => router.push("/ForgotPassword")}>
         <Text style={styles.forgotPasswordText}>Forgot password?</Text>
        </TouchableOpacity>
       </View>
      </View>
     </ScrollView>

     {/* Login Button */}
     <TouchableOpacity
      style={styles.loginButton}
      onPress={handleLogin}
     >
      <Text style={styles.buttonText}>Login</Text>
     </TouchableOpacity>
    </KeyboardAvoidingView>
  );
}
```

Registration

```
import React, { useState } from "react";
import {
  View,
  Text,
  TextInput,
  TouchableOpacity,
  StyleSheet,
  ScrollView,
  ImageBackground,
} from "react-native";
import Icon from "react-native-vector-icons/MaterialIcons";
import { useRouter } from "expo-router";
import Svg, { Path } from "react-native-svg";
import {auth} from "../../config/firebaseConfig";
import { createUserWithEmailAndPassword, updateProfile } from
"firebase/auth";
import styles from "../../styles/registrationStyle";
```

```
EXPORT DEFAULT FUNCTION REGISTRATION() {
 CONST ROUTER = USEROUTER();

 CONST [FULLNAME, SETFULLNAME] = USESTATE("");
 CONST [EMAIL, SETEMAIL] = USESTATE("");
 CONST [PASSWORD, SETPASSWORD] = USESTATE("");
 CONST [ERROR, SETERROR] = USESTATE("");

 CONST HANDLEREGISTER = ASYNC () => {
  IF (!FULLNAME.TRIM() || !EMAIL.TRIM() || !PASSWORD.TRIM()) {
   SETERROR("PLEASE FILL ALL FIELDS.");
   RETURN;
  }

  TRY {
   SETERROR("");

   CONST USERCREDENTIAL = AWAIT CREATEUSERWITHEMAILANDPASSWORD(
    AUTH,
    EMAIL,
    PASSWORD
   );

   AWAIT UPDATEPROFILE(USERCREDENTIAL.USER, {
    DISPLAYNAME: FULLNAME,
   });

   CONSOLE.LOG("USER REGISTERED:", USERCREDENTIAL.USER);

   ROUTER.REPLACE("/MAINAPP");
  } CATCH (ERROR) {
   CONSOLE.ERROR("REGISTRATION ERROR FULL:", ERROR);
   CONST ERR = ERROR AS { CODE?: STRING; MESSAGE?: STRING };

   IF (ERR.CODE === "AUTH/EMAIL-ALREADY-IN-USE") {
    SETERROR("THIS EMAIL IS ALREADY IN USE.");
   } ELSE IF (ERR.CODE === "AUTH/INVALID-EMAIL") {
    SETERROR("INVALID EMAIL ADDRESS.");
   } ELSE IF (ERR.CODE === "AUTH/WEAK-PASSWORD") {
    SETERROR("PASSWORD SHOULD BE AT LEAST 6 CHARACTERS.");
   } ELSE {
    SETERROR(ERR.MESSAGE || "REGISTRATION FAILED. PLEASE TRY AGAIN.");
   }
  }

 };
 RETURN (
  <SCROLLVIEW CONTENTCONTAINERSTYLE={STYLES.SCROLLCONTAINER}>
   {/* TOP BACKGROUND WITH CORRECT CURVE */}
   <VIEW STYLE={STYLES.BACKGROUNDCONTAINER}>
    <IMAGEBACKGROUND
     SOURCE={REQUIRE("../../ASSETS/IMAGES/BACKGOUNDPRE.JPG")}
     STYLE={STYLES.BACKGROUND}
    >
     <TOUCHABLEOPACITY
      STYLE={STYLES.BACKBUTTON}
      ONPRESS={() => ROUTER.BACK()}
     >
      <ICON NAME="ARROW-BACK" SIZE={24} COLOR="1B5E20" />
     </TOUCHABLEOPACITY>
    </IMAGEBACKGROUND>

    <SVG
     HEIGHT="100"
     WIDTH="100%"
     VIEWBOX="0 0 1440 320"
     STYLE={STYLES.WAVE}
    >
     <PATH
      FILL="#FFF"

D="M0,192L60,176C120,160,240,128,360,133.3C480,139,600,181,720,181.3C840,181,
960,139,1080,117.3C1200,96,1320,96,1380,96L1440,96L1440,320L0,320Z"
     />
    </SVG>
   </VIEW>

   <VIEW STYLE={STYLES.CONTENT}>
    <TEXT STYLE={STYLES.HEADER}>REGISTER</TEXT>
    <TEXT STYLE={STYLES.LOGINPROMPTTEXT}>CREATE YOUR NEW ACCOUNT</TEXT>

    <VIEW STYLE={STYLES.INPUTCONTAINER}>
     <ICON NAME="PERSON" SIZE={20} COLOR="#AAA" STYLE={STYLES.INPUTICON} />
     <TEXTINPUT
      STYLE={STYLES.INPUT}
      PLACEHOLDER="FULL NAME"
      PLACEHOLDERTEXTCOLOR="#AAA"
      VALUE={FULLNAME}
      ONCHANGETEXT={SETFULLNAME}
     />
    </VIEW>
```

```
      <View style={styles.inputContainer}>
        <Icon name="email" size={20} color="#aaa" style={styles.inputIcon} />
        <TextInput
          style={styles.input}
          placeholder="Email"
          placeholderTextColor="#aaa"
          keyboardType="email-address"
          value={email}
          onChangeText={setEmail}
        />
      </View>

      <View style={styles.inputContainer}>
        <Icon name="lock" size={20} color="#aaa" style={styles.inputIcon} />
        <TextInput
          style={styles.input}
          placeholder="Password"
          placeholderTextColor="#aaa"
          secureTextEntry
          value={password}
          onChangeText={setPassword}
        />
      </View>

      <TouchableOpacity
        style={styles.registerButton}
        onPress={handleRegister}
      >
        <Text style={styles.buttonText}>Sign Up</Text>
      </TouchableOpacity>
    </View>
  </ScrollView>
 );
}
```

AboutScreen

```
import React from "react";
import { View, Text, ScrollView, Image, TouchableOpacity } from "react-
native";
import { useRouter } from "expo-router";
import Icon from "react-native-vector-icons/MaterialIcons";

export default function AboutScreen() {
  const router = useRouter();
  RETURN (
   <VIEW STYLE={{ FLEX: 1, BACKGROUNDCOLOR: "#F9F9F9" }}>
    <SCROLLVIEW SHOWSVERTICALSCROLLINDICATOR={FALSE}>
     {/* TOP SECTION WITH IMAGE AND CURVED WHITE BOTTOM */}
     <VIEW STYLE={{ POSITION: "RELATIVE", ALIGNITEMS: "CENTER" }}>
      <IMAGE
       SOURCE={{
        URI: "HTTPS://IMAGES.UNSPLASH.COM/PHOTO-1483794344563-
D27A8D18014E?Q=80&W=2070&AUTO=FORMAT&FIT=CROP",
       }}
       STYLE={{
        WIDTH: "100%",
        HEIGHT: 250,
        BORDERBOTTOMLEFTRADIUS: 50,
        BORDERBOTTOMRIGHTRADIUS: 50,
       }}
      />
      <VIEW
       STYLE={{
        POSITION: "ABSOLUTE",
        BOTTOM: 0,
        WIDTH: "100%",
        HEIGHT: 50,
        BACKGROUNDCOLOR: "#F9F9F9",
        BORDERTOPLEFTRADIUS: 60,
        BORDERTOPRIGHTRADIUS: 60,
       }}
      />
      <TEXT
       STYLE={{
        POSITION: "ABSOLUTE",
        BOTTOM: 15,
        FONTSIZE: 26,
        FONTWEIGHT: "BOLD",
        COLOR: "#2E7D32",
       }}
      >
       ABOUT
      </TEXT>
     </VIEW>

     {/* ABOUT CONTENT */}
     <VIEW STYLE={{ PADDING: 20 }}>
      <TEXT
       STYLE={{
        FONTSIZE: 22,
        fontWeight: "bold",
        color: "#2e7d32",
        marginBottom: 10,
      }}
    >
      About Smart Plant Tower
    </Text>
    <Text
      style={{
        fontSize: 16,
        color: "#333",
        lineHeight: 22,
        marginBottom: 20,
      }}
    >
      Smart Plant Tower is an intelligent indoor plant monitoring and
      automation system that helps you take care of your plants with ease.
      It monitors essential environmental factors and automates irrigation
      for optimal plant growth.
    </Text>

    <Text
      style={{
        fontSize: 22,
        fontWeight: "bold",
        color: "#2e7d32",
        marginBottom: 10,
      }}
    >
      Key Sensors
    </Text>

    <View style={{ marginBottom: 20 }}>
      <Text style={{ fontSize: 16, color: "#333", marginBottom: 8 }}>
        🌡 <Text style={{ fontWeight: "bold" }}>Temperature Sensor:</Text>{"
"}
        Monitors ambient temperature to ensure your plants stay within
        their ideal range.
      </Text>
      <Text style={{ fontSize: 16, color: "#333", marginBottom: 8 }}>
        💧 <Text style={{ fontWeight: "bold" }}>Humidity Sensor:</Text>{" "}
        Tracks air humidity to maintain the right moisture levels for
        healthy leaves.
      </Text>
      <Text style={{ fontSize: 16, color: "#333", marginBottom: 8 }}>
      🌊 <Text style={{ fontWeight: "bold" }}>Water Level Sensor:</Text>{"
"}
      Ensures the water reservoir never runs dry.
     </Text>
     <Text style={{ fontSize: 16, color: "#333" }}>
      🧪 <Text style={{ fontWeight: "bold" }}>pH Sensor:</Text> Measures
      soil acidity/alkalinity to help you maintain the right conditions
      for growth.
     </Text>
    </View>

    <Text
     style={{
      fontSize: 22,
      fontWeight: "bold",
      color: "#2e7d32",
      marginBottom: 10,
     }}
    >
     Features
    </Text>

    <Text
     style={{
      fontSize: 16,
      color: "#333",
      lineHeight: 22,
      marginBottom: 20,
     }}
    >
     🌿 <Text style={{ fontWeight: "bold" }}>Real-time Monitoring:</Text>{"
"}
     Live updates of temperature, humidity, pH, and EC values.{"\n\n"}
     💧 <Text style={{ fontWeight: "bold" }}>Manual Watering:</Text>{" "}
     Instantly water any plant or all at once using the “Hold to Water”
     feature.{"\n\n"}
     □ <Text style={{ fontWeight: "bold" }}>Smart Scheduling:</Text>{" "}
     Adjust watering interval and duration easily from the app.{"\n\n"}
     ☀ <Text style={{ fontWeight: "bold" }}>Lighting Control:</Text>{" "}
     Manage light exposure manually or automatically.{"\n\n"}
     🌱 <Text style={{ fontWeight: "bold" }}>Optimized Growth:</Text>{" "}
     Keeps plants in the perfect environment for thriving growth.
    </Text>

    <Text
     style={{
```

```
  fontSize: 22,
  fontWeight: "bold",
  color: "#2e7d32",
  marginBottom: 10,
 }}
>
 Why Choose Smart Plant Tower?
</Text>
<Text
 style={{
  fontSize: 16,
  color: "#333",
  lineHeight: 22,
  marginBottom: 30,
 }}
>
 Whether you’re a hobbyist or a dedicated grower, Smart Plant Tower
 automates tedious tasks, saves time, and ensures your plants receive
 the care they need — anytime, anywhere.
</Text>

{/* Go Back Button */}
<TouchableOpacity
 onPress={() => router.back()}
 style={{
  flexDirection: "row",
  alignItems: "center",
  justifyContent: "center",
  backgroundColor: "#2e7d32",
  padding: 12,
  borderRadius: 25,
  elevation: 3,
  shadowColor: "#000",
  shadowOpacity: 0.2,
  shadowRadius: 3,
  marginBottom: 30,
 }}
>
 <Icon name="arrow-back" size={22} color="#fff" />
 <Text
  style={{
   color: "#fff",
   fontSize: 16,
   marginLeft: 6,
   fontWeight: "bold",
  }}
```

```
          >
           Go Back
          </Text>
        </TouchableOpacity>
      </View>
    </ScrollView>
  </View>
 );
}
```

ChangePassword

```
import React, { useState } from "react";
import {
 View,
 Text,
 TextInput,
 TouchableOpacity,
 StyleSheet,
 Alert,
 ScrollView,
 KeyboardAvoidingView,
 Platform,
 ImageBackground,
} from "react-native";
import { getAuth, reauthenticateWithCredential, EmailAuthProvider,
updatePassword } from "firebase/auth";
import { useRouter } from "expo-router";
import Icon from "react-native-vector-icons/MaterialIcons";

export default function ChangePassword() {
 const [currentPassword, setCurrentPassword] = useState("");
 const [newPassword, setNewPassword] = useState("");
 const [confirmPassword, setConfirmPassword] = useState("");
 const router = useRouter();
 const auth = getAuth();

 const handleChangePassword = async () => {
  const user = auth.currentUser;
  if (!user || !user.email) {
   Alert.alert("Error", "No user is currently logged in.");
   return;
  }
  if (!currentPassword || !newPassword || !confirmPassword) {
   Alert.alert("Error", "Please fill in all fields.");
   return;
  }
```

```
    IF (NEWPASSWORD !== CONFIRMPASSWORD) {
      ALERT.ALERT("ERROR", "NEW PASSWORDS DO NOT MATCH.");
      RETURN;
    }
    TRY {
      CONST CREDENTIAL = EMAILAUTHPROVIDER.CREDENTIAL(USER.EMAIL,
CURRENTPASSWORD);
      AWAIT REAUTHENTICATEWITHCREDENTIAL(USER, CREDENTIAL);
      AWAIT UPDATEPASSWORD(USER, NEWPASSWORD);
      ALERT.ALERT("SUCCESS", "PASSWORD HAS BEEN UPDATED.");
      ROUTER.BACK();
    } CATCH (ERROR: ANY) {
      CONSOLE.ERROR("PASSWORD UPDATE ERROR:", ERROR);
      ALERT.ALERT("ERROR", ERROR.MESSAGE || "UNABLE TO UPDATE PASSWORD.");
    }
  };

  RETURN (
    <KEYBOARDAVOIDINGVIEW
      STYLE={{ FLEX: 1 }}
      BEHAVIOR={PLATFORM.OS === "IOS" ? "PADDING" : UNDEFINED}
    >
      <IMAGEBACKGROUND
        SOURCE={{ URI: "HTTPS://WWW.FREEPIK.COM/FREE-PHOTO/GREEN-LEAVES-NATURE-
BACKGROUND-
WALLPAPER_15599956.HTM#FROMVIEW=KEYWORD&PAGE=2&POSITION=34&UUID=A84
5B739-A3A3-4AE2-B294-046BBB5A89C9&QUERY=LEAVES+BACKGROUND" }}
        STYLE={STYLES.BACKGROUND}
        IMAGESTYLE={{ OPACITY: 0.5 }}
      >
        {/* BACK BUTTON */}
        <TOUCHABLEOPACITY STYLE={STYLES.BACKBUTTON} ONPRESS={() =>
ROUTER.BACK()}>
          <ICON NAME="ARROW-BACK" SIZE={28} COLOR="#689F38" />
        </TOUCHABLEOPACITY>

        <SCROLLVIEW CONTENTCONTAINERSTYLE={STYLES.CONTAINER}>
          <VIEW STYLE={STYLES.CARD}>
            <TEXT STYLE={STYLES.TITLE}>CHANGE PASSWORD</TEXT>

            <TEXT STYLE={STYLES.LABEL}>CURRENT PASSWORD</TEXT>
            <TEXTINPUT
              STYLE={STYLES.INPUT}
              PLACEHOLDER="ENTER CURRENT PASSWORD"
              SECURETEXTENTRY
              VALUE={CURRENTPASSWORD}
          onChangeText={setCurrentPassword}
          placeholderTextColor="#888"
        />

        <Text style={styles.label}>New Password</Text>
        <TextInput
          style={styles.input}
          placeholder="Enter new password"
          secureTextEntry
          value={newPassword}
          onChangeText={setNewPassword}
          placeholderTextColor="#888"
        />

        <Text style={styles.label}>Confirm New Password</Text>
        <TextInput
          style={styles.input}
          placeholder="Confirm new password"
          secureTextEntry
          value={confirmPassword}
          onChangeText={setConfirmPassword}
          placeholderTextColor="#888"
        />

        <TouchableOpacity style={styles.button}
onPress={handleChangePassword}>
          <Text style={styles.buttonText}>Update Password</Text>
        </TouchableOpacity>
       </View>
      </ScrollView>
     </ImageBackground>
    </KeyboardAvoidingView>
  );
}

const styles = StyleSheet.create({
  background: {
    flex: 1,
    paddingTop: 50,
    backgroundColor: "#f0f4f0",
  },
  container: {
    flexGrow: 1,
    justifyContent: "center",
    padding: 20,
  },
backButton: {
 position: "absolute",
 top: 50,
 left: 20,
 zIndex: 10,
},
card: {
 backgroundColor: "rgba(255,255,255,0.85)",
 borderRadius: 22,
 padding: 28,
 shadowColor: "#000",
 shadowOffset: { width: 0, height: 6 },
 shadowOpacity: 0.08,
 shadowRadius: 14,
 elevation: 6,
},
title: {
 fontSize: 28,
 fontWeight: "bold",
 marginBottom: 28,
 textAlign: "center",
 color: "#689F38",
},
label: {
 fontSize: 16,
 fontWeight: "600",
 marginBottom: 6,
 color: "#444",
},
input: {
 borderWidth: 1,
 borderColor: "#ccc",
 padding: 16,
 borderRadius: 14,
 marginBottom: 18,
 fontSize: 16,
 backgroundColor: "#f8f8f8",
 color: "#333",
},
button: {
 backgroundColor: "#689F38",
 paddingVertical: 16,
 borderRadius: 14,
 marginTop: 8,
 shadowColor: "#000",
 shadowOffset: { width: 0, height: 4 },
  shadowOpacity: 0.12,
  shadowRadius: 8,
  elevation: 3,
 },
 buttonText: {
  color: "#fff",
  textAlign: "center",
  fontWeight: "700",
  fontSize: 16,
 },
});
```

Humidity

```
import React, { useState, useEffect } from "react";
import { View, Text, StyleSheet, Image, TouchableOpacity, Dimensions } from
"react-native";
import Ionicons from "react-native-vector-icons/Ionicons";
import { db } from "../../config/firebaseConfig";
import { ref, onValue } from "firebase/database";
import { router } from "expo-router";

const { width, height } = Dimensions.get("window");
const CURVE_HEIGHT = 180;

export default function Humidity() {
 const [humidity, setHumidity] = useState({
  irrigationHumidity: 0,
  normalHumidity: 0,
 });

 useEffect(() => {
  const irrigationRef = ref(db, "/irrigation/environment/humidity");
  const normalRef = ref(db, "/environment/humidity");

  const unsubIrrigation = onValue(irrigationRef, (snapshot) => {
   setHumidity((prev) => ({
    ...prev,
    irrigationHumidity: snapshot.val() ?? 0,
   }));
  });

  const unsubNormal = onValue(normalRef, (snapshot) => {
   setHumidity((prev) => ({
    ...prev,
    normalHumidity: snapshot.val() ?? 0,
   }));
   });

   RETURN () => {
    UNSUBIRRIGATION();
    UNSUBNORMAL();
   };
  }, []);

  RETURN (
   <VIEW STYLE={STYLES.SCREEN}>
    {/* BACK BUTTON */}
    <TOUCHABLEOPACITY STYLE={STYLES.BACKBUTTON} ONPRESS={() =>
ROUTER.BACK()}>
     <IONICONS NAME="ARROW-BACK" SIZE={28} COLOR="#33691E" />
    </TOUCHABLEOPACITY>

    {/* PLANT IMAGE */}
    <IMAGE SOURCE={REQUIRE("../../ASSETS/IMAGES/PLANTPOT2.PNG")}
STYLE={STYLES.IMAGE} />

    {/* INFO SECTION */}
    <VIEW STYLE={STYLES.INFOSECTION}>
     <TEXT STYLE={STYLES.TITLE}>ABOUT HUMIDITY</TEXT>
     <TEXT STYLE={STYLES.SUBTITLE}>MOISTURE FOR GROWTH</TEXT>
     <TEXT STYLE={STYLES.DESC}>
      HUMIDITY HELPS PLANTS REGULATE WATER LOSS AND ABSORB NUTRIENTS.
MAINTAINING
      BALANCED AIR AND SOIL HUMIDITY ENSURES HEALTHY AND THRIVING PLANTS.
     </TEXT>
    </VIEW>

        {/* 🌿 HORIZONTAL PLANT LINE IMAGE */}
            <IMAGE
             SOURCE={REQUIRE("../../ASSETS/IMAGES/PLANTLINE.PNG")}
             STYLE={STYLES.PLANTLINEIMAGE}
            />

    {/* GREEN CURVE SECTION */}
    <VIEW STYLE={STYLES.CURVECONTAINER}>
     <VIEW STYLE={STYLES.CURVESHAPE} />

     {/* STATS ROW */}
     <VIEW STYLE={STYLES.STATSROW}>
      <VIEW STYLE={STYLES.STATITEM}>
       <IONICONS NAME="WATER-OUTLINE" SIZE={26} COLOR="#FFF" />
       <TEXT STYLE={STYLES.STATLABEL}>IRRIGATION</TEXT>
```

```
            <Text style={styles.statValue}>
              {humidity.irrigationHumidity.toFixed(1)}%
            </Text>
          </View>

          <View style={styles.statItem}>
            <Ionicons name="cloud-outline" size={26} color="#fff" />
            <Text style={styles.statLabel}>Ambient</Text>
            <Text style={styles.statValue}>
              {humidity.normalHumidity.toFixed(1)}%
            </Text>
          </View>
        </View>
      </View>
    </View>
  );
}

const styles = StyleSheet.create({
  screen: {
    flex: 1,
    backgroundColor: "#F0F8F1",
  },
  backButton: {
    position: "absolute",
    top: 45,
    left: 20,
    zIndex: 10,
    backgroundColor: "#E8F5E9",
    padding: 8,
    borderRadius: 50,
    elevation: 4,
  },
  image: {
    width: "100%",
    height: height * 0.45,
    resizeMode: "contain",
    marginTop: 100,
    alignSelf: "center",
  },
  infoSection: {
    position: "absolute",
    bottom: CURVE_HEIGHT + 30,
    width: "100%",
    paddingHorizontal: 24,
  },
title: {
  fontSize: 22,
  fontWeight: "700",
  color: "#33691E",
},
subtitle: {
  fontSize: 16,
  fontWeight: "600",
  color: "#558B2F",
  marginBottom: 6,
},
desc: {
  fontSize: 13,
  color: "#4F4F4F",
  marginBottom: 8,
},
curveContainer: {
  position: "absolute",
  left: 0,
  bottom: 0,
  width: "100%",
  height: CURVE_HEIGHT,
  justifyContent: "flex-end",
},
curveShape: {
  position: "absolute",
  bottom: 0,
  width: "100%",
  height: CURVE_HEIGHT,
  backgroundColor: "#7CB342",
  borderTopLeftRadius: 300,
  transform: [{ scaleX: 1.2 }],
  zIndex: 0,
},
statsRow: {
  flexDirection: "row",
  alignItems: "center",
  justifyContent: "space-evenly",
  height: "100%",
},
statItem: {
  alignItems: "center",
},
statLabel: {
  fontSize: 13,
  color: "#fff",
    fontWeight: "600",
    marginTop: 2,
  },
  statValue: {
    fontSize: 13,
    color: "#fff",
    marginTop: 1,
  },
  plantLineImage: {
  width: width * 1.5,
  height: 180,
  resizeMode: "contain",
  position: "absolute",
  top: CURVE_HEIGHT + 555,
  left: -width * 0.25,
  zIndex: 10,
},
});
```

Notification

```
import React, { useEffect, useState } from "react";
import {
  View,
  Text,
  StyleSheet,
  TouchableOpacity,
  ScrollView,
  ActivityIndicator,
  SafeAreaView,
  Modal,
} from "react-native";
import Icon from "react-native-vector-icons/MaterialIcons";
import { useRouter } from "expo-router";
import { db } from "../../config/firebaseConfig";
import { ref, onValue } from "firebase/database";
import { BlurView } from "expo-blur";

interface NotificationItem {
  id: string;
  type: string;
  status: string;
  color: string;
}

export default function NotificationsScreen() {
  const router = useRouter();
const [notifications, setNotifications] = useState<NotificationItem[]>([]);
const [loading, setLoading] = useState(true);
const [modalVisible, setModalVisible] = useState(false);

useEffect(() => {
 const paths = {
  envHumidity: ref(db, "environment/humidity"),
  envTemp: ref(db, "environment/temp"),
  ph: ref(db, "environment/ph"),
  phStatus: ref(db, "environment/phStatus"),
  irrigationHumidity: ref(db, "irrigation/environment/humidity"),
  irrigationTemp: ref(db, "irrigation/environment/temp"),
  tankStatus: ref(db, "irrigation/watertank/status"),
 };

 const listeners = Object.entries(paths).map(([key, dbRef]) =>
  onValue(dbRef, (snapshot) => {
   const value = snapshot.val();
   updateNotifications(key, value);
  })
 );

 return () =>
  listeners.forEach((unsub) => typeof unsub === "function" && unsub());
 // eslint-disable-next-line react-hooks/exhaustive-deps
}, []);

const updateNotifications = (key: string, value: any) => {
 setNotifications((prev) => {
  let updated = [...prev];

  const removeOld = (type: string) =>
   (updated = updated.filter((n) => n.type !== type));

  const addNote = (type: string, status: string, color: string) =>
   updated.push({ id: type, type, status, color });

  if (key === "envHumidity") {
   removeOld("Humidity (Environment)");
   if (typeof value === "number") {
    if (value < 40) addNote("Humidity (Environment)", "Low", "#FDD835");
    else if (value > 80) addNote("Humidity (Environment)", "High", "#E53935");
   }
  }

  if (key === "envTemp") {
      removeOld("Temperature (Environment)");
      if (typeof value === "number") {
        if (value < 20) addNote("Temperature (Environment)", "Too Cold",
"#039BE5");
        else if (value > 32) addNote("Temperature (Environment)", "Too Hot",
"#E53935");
      }
    }

    if (key === "ph" || key === "phStatus") {
      removeOld("pH Level");
      if (typeof value === "number") {
        if (value < 5.5 || value > 7.5) addNote("pH Level", `Unhealthy (${value})`,
"#E53935");
      } else if (typeof value === "string" && value !== "Neutral") {
        addNote("pH Level", value, "#FDD835");
      }
    }

    if (key === "irrigationHumidity") {
      removeOld("Humidity (Irrigation)");
      if (typeof value === "number") {
        if (value < 40) addNote("Humidity (Irrigation)", "Low", "#FDD835");
        else if (value > 80) addNote("Humidity (Irrigation)", "High", "#E53935");
      }
    }

    if (key === "irrigationTemp") {
      removeOld("Temperature (Irrigation)");
      if (typeof value === "number") {
        if (value < 20) addNote("Temperature (Irrigation)", "Too Cold",
"#039BE5");
        else if (value > 32) addNote("Temperature (Irrigation)", "Too Hot",
"#E53935");
      }
    }

    if (key === "tankStatus") {
      removeOld("Water Tank");
      if (value === "EMPTY") addNote("Water Tank", "Empty", "#E53935");
      else if (value === "LOW") addNote("Water Tank", "Low", "#FDD835");
    }

    return updated;
  });
```

```
    setLoading(false);
  };

  useEffect(() => {
    if (notifications.length > 0) setModalVisible(true);
  }, [notifications]);

  return (
    <SafeAreaView style={styles.container}>
      <View style={styles.header}>
        <TouchableOpacity style={styles.backButton} onPress={() =>
router.back()}>
          <Icon name="arrow-back-ios" size={22} color="#4CAF50" />
        </TouchableOpacity>
        <Text style={styles.headerTitle}>Notifications</Text>
      </View>

      {loading ? (
        <ActivityIndicator size="large" color="#8BC34A" style={{ marginTop: 40
}} />
      ) : (
        <ScrollView contentContainerStyle={styles.scrollContainer}>
          {notifications.length > 0 ? (
            notifications.map((item) => (
              <View key={item.id} style={styles.notificationCard}>
                <Icon name="warning" size={24} color={item.color}
style={styles.icon} />
                <View style={{ flex: 1 }}>
                  <Text style={styles.notificationTitle}>{item.type}</Text>
                  <Text style={[styles.notificationText, { color: item.color }]}>
                    {item.status}
                  </Text>
                </View>
              </View>
            ))
          ) : (
            <Text style={styles.emptyText}>All sensors are healthy 🌿</Text>
          )}
        </ScrollView>
      )}

      {/* Modal with blur background */}
      <Modal
        visible={modalVisible}
        transparent
        animationType="fade"
    onRequestClose={() => setModalVisible(false)}
   >
    <View style={styles.overlayContainer}>
     {/* Blur background */}
     <BlurView intensity={80} tint="light" style={StyleSheet.absoluteFill}
/>

     {/* Semi-transparent overlay for touch-to-close */}
     <TouchableOpacity
      style={styles.backdrop}
      activeOpacity={1}
      onPress={() => setModalVisible(false)}
     />

     {/* Modal card content */}
     <View style={styles.modalCard}>
      <Text style={styles.modalTitle}>Notifications</Text>

      {loading ? (
       <ActivityIndicator size="large" color="#8BC34A" style={{ marginTop:
10 }} />
      ) : notifications.length > 0 ? (
       <ScrollView
        style={{ width: "100%" }}
        contentContainerStyle={{ paddingVertical: 8 }}
       >
        {notifications.map((item) => (
         <View key={item.id} style={styles.notificationCard}>
          <Icon name="warning" size={22} color={item.color}
style={styles.icon} />
          <View style={{ flex: 1 }}>
           <Text style={styles.notificationTitle}>{item.type}</Text>
           <Text style={[styles.notificationText, { color: item.color }]}>
            {item.status}
           </Text>
          </View>
         </View>
        ))}
       </ScrollView>
      ) : (
       <Text style={styles.emptyText}>All sensors are healthy 🌿</Text>
      )}

      <TouchableOpacity
       style={styles.modalClose}
       onPress={() => setModalVisible(false)}
```

```
          >
            <Text style={{ color: "#fff", fontWeight: "700" }}>Close</Text>
          </TouchableOpacity>
        </View>
      </View>
    </Modal>
  </SafeAreaView>
 );
}

const styles = StyleSheet.create({
 container: { flex: 1, backgroundColor: "#F9FAF5" },
 header: {
  flexDirection: "row",
  alignItems: "center",
  padding: 20,
  backgroundColor: "#8BC34A",
  borderBottomLeftRadius: 25,
  borderBottomRightRadius: 25,
  elevation: 3,
 },
 backButton: {
  marginRight: 10,
  backgroundColor: "rgba(255,255,255,0.4)",
  padding: 6,
  borderRadius: 20,
 },
 headerTitle: { fontSize: 20, fontWeight: "bold", color: "#fff" },
 scrollContainer: { padding: 20 },
 notificationCard: {
  flexDirection: "row",
  alignItems: "center",
  backgroundColor: "#fff",
  borderRadius: 15,
  padding: 15,
  marginBottom: 10,
  shadowColor: "#000",
  shadowOpacity: 0.1,
  shadowOffset: { width: 0, height: 2 },
  shadowRadius: 5,
  elevation: 2,
 },
 icon: { marginRight: 15 },
 notificationTitle: { fontSize: 16, fontWeight: "bold", color: "#333" },
 notificationText: { fontSize: 14, marginTop: 3 },
 emptyText: { textAlign: "center", color: "#777", fontSize: 16, marginTop: 40 },
 overlayContainer: {
  flex: 1,
  justifyContent: "center",
  alignItems: "center",
 },
 backdrop: {
  ...StyleSheet.absoluteFillObject,
  backgroundColor: "rgba(0,0,0,0.18)",
 },
 modalCard: {
  width: "86%",
  maxHeight: "72%",
  backgroundColor: "#fff",
  borderRadius: 14,
  padding: 14,
  elevation: 8,
  shadowColor: "#000",
  shadowOpacity: 0.16,
  shadowOffset: { width: 0, height: 6 },
  shadowRadius: 12,
  zIndex: 100,
 },
 modalTitle: { fontSize: 18, fontWeight: "700", marginBottom: 8, color:
"#263238" },
 modalClose: {
  marginTop: 12,
  backgroundColor: "#8BC34A",
  paddingVertical: 10,
  borderRadius: 10,
  alignItems: "center",
 },
});
```

pH

```
import React, { useEffect, useState } from "react";
import { View, Text, StyleSheet, Image, TouchableOpacity, Dimensions } from
"react-native";
import Ionicons from "react-native-vector-icons/Ionicons";
import { db } from "../../config/firebaseConfig";
import { ref, onValue } from "firebase/database";
import { router } from "expo-router";

const { width, height } = Dimensions.get("window");

export default function PlantEnvironment() {
 const [env, setEnv] = useState({ temp: 0, humidity: 0, ph: 0, phStatus: "Neutral"
});

 useEffect(() => {
   const tempRef = ref(db, "environment/temp");
   const humRef = ref(db, "environment/humidity");
   const phRef = ref(db, "environment/ph");
   const phStatusRef = ref(db, "environment/phStatus");

   onValue(tempRef, (snap) => setEnv((prev) => ({ ...prev, temp: snap.val() ?? 0
})));
   onValue(humRef, (snap) => setEnv((prev) => ({ ...prev, humidity: snap.val() ?? 0
})));
   onValue(phRef, (snap) => setEnv((prev) => ({ ...prev, ph: snap.val() ?? 0 })));
   onValue(phStatusRef, (snap) =>
    setEnv((prev) => ({ ...prev, phStatus: snap.val() ?? "Neutral" }))
   );
 }, []);

 return (
   <View style={styles.screen}>

     <TouchableOpacity style={styles.backButton} onPress={() =>
router.back()}>
      <Ionicons name="arrow-back" size={28} color="#33691E" />
     </TouchableOpacity>



     <Image source={require("../../assets/images/imgplant2.png")}
style={styles.image} />



<View style={styles.infoSection}>
 <Text style={styles.title}>About pH Levels</Text>
 <Text style={styles.subtitle}>Safe Water for Your Plants</Text>
 <Text style={styles.desc}>
   pH shows how acidic or alkaline your water is. Keeping it balanced helps
plants
   absorb nutrients properly and stay healthy.
 </Text>
 <Text style={styles.desc}>
   This system monitors pH to make sure your plants get safe and balanced
water.
 </Text>
</View>
      {/* 🌿 Horizontal Plant Line Image */}
```

```
          <Image
           source={require("../../assets/images/plantline.png")}
           style={styles.plantLineImage}
          />



   <View style={styles.curveContainer}>
    <View style={styles.curveShape} />
    <View style={styles.statsRow}>
     <View style={styles.statItem}>
      <Ionicons name="thermometer-outline" size={26} color="#fff" />
      <Text style={styles.statLabel}>Temperature</Text>
      <Text style={styles.statValue}>{env.temp.toFixed(1)}°C</Text>
     </View>

     <View style={styles.statItem}>
      <Ionicons name="water-outline" size={26} color="#fff" />
      <Text style={styles.statLabel}>Humidity</Text>
      <Text style={styles.statValue}>{env.humidity.toFixed(1)}%</Text>
     </View>

     <View style={styles.statItem}>
      <Ionicons name="beaker-outline" size={26} color="#fff" />
      <Text style={styles.statLabel}>pH</Text>
      <Text style={styles.statValue}>
       {env.ph.toFixed(2)} ({env.phStatus})
      </Text>
     </View>
    </View>
   </View>
  </View>
 );
}

const CURVE_HEIGHT = 180;

const styles = StyleSheet.create({
 screen: {
  flex: 1,
  backgroundColor: "#F0F8F1",
 },
 backButton: {
  position: "absolute",
  top: 45,
  left: 20,
  zIndex: 10,
  backgroundColor: "#E8F5E9",
  padding: 8,
  borderRadius: 50,
  elevation: 4,
},
image: {
  width: "100%",
  height: height * 0.45,
  resizeMode: "contain",
  marginTop: 100,
  alignSelf: "center",
},
infoSection: {
  position: "absolute",
  bottom: CURVE_HEIGHT + 30,
  width: "100%",
  paddingHorizontal: 24,
},
title: {
  fontSize: 22,
  fontWeight: "700",
  color: "#33691E",
},
subtitle: {
  fontSize: 16,
  fontWeight: "600",
  color: "#558B2F",
  marginBottom: 6,
},
desc: {
  fontSize: 13,
  color: "#4F4F4F",
  marginBottom: 8,
},
price: {
  fontSize: 18,
  fontWeight: "700",
  color: "#33691E",
},
curveContainer: {
  position: "absolute",
  left: 0,
  bottom: 0,
  width: "100%",
  height: CURVE_HEIGHT,
  justifyContent: "flex-end",
 },
 curveShape: {
  position: "absolute",
  bottom: 0,
  width: "100%",
  height: CURVE_HEIGHT + 25,
  backgroundColor: "#7CB342",
  borderTopLeftRadius: 300,
  transform: [{ scaleX: 1.2 }],
  zIndex: 0,
 },
 statsRow: {
  flexDirection: "row",
  alignItems: "center",
  justifyContent: "space-evenly",
  height: "100%",
 },
 statItem: {
  alignItems: "center",
  marginTop: -50,
 },
 statLabel: {
  fontSize: 13,
  color: "#fff",
  fontWeight: "600",
  marginTop: 2,
 },
 statValue: {
  fontSize: 13,
  color: "#fff",
  marginTop: 1,
 },
  plantLineImage: {
 width: width * 1.5,
 height: 180,
 resizeMode: "contain",
 position: "absolute",
 top: CURVE_HEIGHT + 555,
 left: -width * 0.25,
 zIndex: 10,
},
});
```

SoilMoisture

```
import React, { useEffect, useState } from 'react';
```

```
IMPORT {
  VIEW,
  TEXT,
  STYLESHEET,
  ACTIVITYINDICATOR,
  SCROLLVIEW,
  SAFEAREAVIEW,
  PLATFORM,
  DIMENSIONS,
  ANIMATED,
} FROM 'REACT-NATIVE';
IMPORT { LINECHART } FROM 'REACT-NATIVE-CHART-KIT';

CONST SCREENWIDTH = DIMENSIONS.GET('WINDOW').WIDTH;

CONST CHARTCONFIG = {
  BACKGROUNDGRADIENTFROM: '#E8F5E9',
  BACKGROUNDGRADIENTTO: '#A5D6A7',
  DECIMALPLACES: 0,
  COLOR: (OPACITY = 1) => `RGBA(56, 142, 60, ${OPACITY})`,
  LABELCOLOR: () => '#4CAF50',
  PROPSFORDOTS: {
    R: '5',
    STROKEWIDTH: '2',
    STROKE: '#388E3C',
  },
  PROPSFORBACKGROUNDLINES: {
    STROKE: '#C8E6C9',
  },
};

CONST SOILMOISTURE = () => {
  CONST [MOISTUREDATA, SETMOISTUREDATA] = USESTATE<RECORD<STRING, NUMBER> |
NULL>(NULL);
  CONST [CHARTOPACITY] = USESTATE(NEW ANIMATED.VALUE(0));

  USEEFFECT(() => {
    CONST FETCHMOISTURE = ASYNC () => {
      TRY {
        CONST RESPONSE = AWAIT FETCH(
          'HTTPS://FIR-TESTAPP-A1461-DEFAULT-RTDB.ASIA-
SOUTHEAST1.FIREBASEDATABASE.APP/SENSORS/SOILMOISTURE.JSON'
        );
        CONST DATA = AWAIT RESPONSE.JSON();
        SETMOISTUREDATA(DATA);
        ANIMATED.TIMING(CHARTOPACITY, {
```

```
    toValue: 1,
    duration: 1000,
    useNativeDriver: true,
   }).start();
  } catch (error) {
   console.error('Error fetching soil moisture:', error);
  }
 };

 fetchMoisture();
 const interval = setInterval(fetchMoisture, 5000);
 return () => clearInterval(interval);
}, []);

const getAverage = (data: Record<string, number>) => {
 const values = Object.values(data);
 const total = values.reduce((sum, value) => sum + value, 0);
 return Math.round(total / values.length);
};

return (
 <SafeAreaView style={styles.safeArea}>
  <ScrollView contentContainerStyle={styles.container}>
   {moistureData ? (
    <>
     <Animated.View style={[styles.chartCard, { opacity: chartOpacity }]}>
      <Text style={styles.cardTitle}>Soil Moisture</Text>
      <LineChart
       data={{
        labels: Object.keys(moistureData).map((_, i) => `P${i + 1}`),
        datasets: [
         {
          data: Object.values(moistureData),
         },
        ],
       }}
       width={screenWidth - 60}
       height={220}
       chartConfig={chartConfig}
       bezier
       style={styles.chartStyle}
      />
     </Animated.View>

     <View style={styles.summaryCard}>
```

```
          <Text
style={styles.summaryValue}>{getAverage(moistureData)}%</Text>
          <Text style={styles.summaryLabel}>Usage</Text>
        </View>
      </>
    ) : (
      <ActivityIndicator size="large" color="#333" />
    )}
   </ScrollView>
  </SafeAreaView>
 );
};

export default SoilMoisture;

const styles = StyleSheet.create({
 safeArea: {
  flex: 1,
  backgroundColor: '#e8f5e9',
 },
 container: {
  paddingTop: Platform.OS === 'android' ? 60 : 40,
  paddingBottom: 40,
  paddingHorizontal: 20,
 },
 chartCard: {
  backgroundColor: '#ffffff',
  borderRadius: 16,
  paddingVertical: 20,
  paddingHorizontal: 10,
  shadowColor: '#000',
  shadowOffset: { width: 0, height: 2 },
  shadowOpacity: 0.08,
  shadowRadius: 6,
  elevation: 3,
  marginBottom: 20,
  overflow: 'hidden',
  alignItems: 'center',
 },
 cardTitle: {
  fontSize: 18,
  fontWeight: '600',
  color: '#2e7d32',
  alignSelf: 'center',
  marginBottom: 12,
 },
 chartStyle: {
  borderRadius: 12,
 },
 summaryCard: {
  backgroundColor: '#a5d6a7',
  borderRadius: 14,
  paddingVertical: 26,
  alignItems: 'center',
  shadowColor: '#000',
  shadowOffset: { width: 0, height: 2 },
  shadowOpacity: 0.1,
  shadowRadius: 6,
  elevation: 3,
 },
 summaryValue: {
  fontSize: 36,
  fontWeight: '700',
  color: '#ffffff',
 },
 summaryLabel: {
  fontSize: 14,
  color: '#ffffff',
  marginTop: 4,
  opacity: 0.8,
 },
});
```

Temperature

```
import React, { useState, useEffect } from "react";
import { View, Text, StyleSheet, Image, TouchableOpacity, Dimensions } from
"react-native";
import Ionicons from "react-native-vector-icons/Ionicons";
import { db } from "../../config/firebaseConfig";
import { ref, onValue } from "firebase/database";
import { router } from "expo-router";

const { width, height } = Dimensions.get("window");
const CURVE_HEIGHT = 180;

export default function Temperature() {
 const [temps, setTemps] = useState({
  irrigationTemp: 0,
  normalTemp: 0,
 });

 useEffect(() => {
    CONST IRRIGATIONREF = REF(DB, "/IRRIGATION/ENVIRONMENT/TEMP");
    CONST NORMALREF = REF(DB, "/ENVIRONMENT/TEMP");

    CONST UNSUBIRRIGATION = ONVALUE(IRRIGATIONREF, (SNAPSHOT) => {
     SETTEMPS((PREV) => ({
      ...PREV,
      IRRIGATIONTEMP: SNAPSHOT.VAL() ?? 0,
     }));
    });

    CONST UNSUBNORMAL = ONVALUE(NORMALREF, (SNAPSHOT) => {
     SETTEMPS((PREV) => ({
      ...PREV,
      NORMALTEMP: SNAPSHOT.VAL() ?? 0,
     }));
    });

    RETURN () => {
     UNSUBIRRIGATION();
     UNSUBNORMAL();
    };
   }, []);

  RETURN (
    <VIEW STYLE={STYLES.SCREEN}>
     {/* BACK BUTTON */}
     <TOUCHABLEOPACITY STYLE={STYLES.BACKBUTTON} ONPRESS={() =>
ROUTER.BACK()}>
      <IONICONS NAME="ARROW-BACK" SIZE={28} COLOR="#33691E" />
     </TOUCHABLEOPACITY>

     {/* IMAGE */}
     <IMAGE SOURCE={REQUIRE("../../ASSETS/IMAGES/PLANTPOT3.PNG")}
STYLE={STYLES.IMAGE} />

     {/* INFO SECTION */}
     <VIEW STYLE={STYLES.INFOSECTION}>
      <TEXT STYLE={STYLES.TITLE}>ABOUT TEMPERATURE</TEXT>
      <TEXT STYLE={STYLES.SUBTITLE}>OPTIMAL HEAT FOR GROWTH</TEXT>
      <TEXT STYLE={STYLES.DESC}>
       TEMPERATURE AFFECTS PLANT METABOLISM AND GROWTH. KEEPING IT WITHIN AN
IDEAL RANGE
       HELPS YOUR PLANTS THRIVE AND PREVENTS STRESS OR DAMAGE.
      </TEXT>
     </VIEW>
          {/* 🌿 HORIZONTAL PLANT LINE IMAGE */}
```

```
            <Image
             source={require("../../assets/images/plantline.png")}
             style={styles.plantLineImage}
            />


    {/* Green Curve Section */}
    <View style={styles.curveContainer}>
     <View style={styles.curveShape} />

     {/* Stats */}
     <View style={styles.statsRow}>
      <View style={styles.statItem}>
       <Ionicons name="flame-outline" size={26} color="#fff" />
       <Text style={styles.statLabel}>Irrigation</Text>
       <Text
style={styles.statValue}>{temps.irrigationTemp.toFixed(1)}°C</Text>
      </View>

      <View style={styles.statItem}>
       <Ionicons name="sunny-outline" size={26} color="#fff" />
       <Text style={styles.statLabel}>Ambient</Text>
       <Text
style={styles.statValue}>{temps.normalTemp.toFixed(1)}°C</Text>
      </View>
     </View>
    </View>
   </View>
 );
}

const styles = StyleSheet.create({
 screen: {
  flex: 1,
  backgroundColor: "#F0F8F1",
 },
 backButton: {
  position: "absolute",
  top: 45,
  left: 20,
  zIndex: 10,
  backgroundColor: "#E8F5E9",
  padding: 8,
  borderRadius: 50,
  elevation: 4,
 },
image: {
  width: 500,
  height: 400,
  resizeMode: "contain",
  marginTop: 100,
  alignSelf: "center",
},
infoSection: {
  position: "absolute",
  bottom: CURVE_HEIGHT + 30,
  width: "100%",
  paddingHorizontal: 24,
},
title: {
  fontSize: 22,
  fontWeight: "700",
  color: "#33691E",
},
subtitle: {
  fontSize: 16,
  fontWeight: "600",
  color: "#558B2F",
  marginBottom: 6,
},
desc: {
  fontSize: 13,
  color: "#4F4F4F",
  marginBottom: 8,
},
curveContainer: {
  position: "absolute",
  left: 0,
  bottom: 0,
  width: "100%",
  height: CURVE_HEIGHT,
  justifyContent: "flex-end",
},
curveShape: {
  position: "absolute",
  bottom: 0,
  width: "100%",
  height: CURVE_HEIGHT,
  backgroundColor: "#7CB342",
  borderTopLeftRadius: 300,
  transform: [{ scaleX: 1.2 }],
  zIndex: 0,
 },
 statsRow: {
  flexDirection: "row",
  alignItems: "center",
  justifyContent: "space-evenly",
  height: "100%",
 },
 statItem: {
  alignItems: "center",
 },
 statLabel: {
  fontSize: 13,
  color: "#fff",
  fontWeight: "600",
  marginTop: 2,
 },
 statValue: {
  fontSize: 13,
  color: "#fff",
  marginTop: 1,
 },
  plantLineImage: {
 width: width * 1.5,
 height: 180,
 resizeMode: "contain",
 position: "absolute",
 top: CURVE_HEIGHT + 555,
 left: -width * 0.25,
 zIndex: 10,
},
});
```

TermsAndConditions

```
import React, { useState } from "react";
import {
 View,
 Text,
 StyleSheet,
 Image,
 TouchableOpacity,
 Dimensions,
 ScrollView,
 Modal,
} from "react-native";
import Ionicons from "react-native-vector-icons/Ionicons";
import { router } from "expo-router";
```

```
CONST { WIDTH, HEIGHT } = DIMENSIONS.GET("WINDOW");
CONST CURVE_HEIGHT = 180;
CONST ACCENT = "#A8C93E";

EXPORT DEFAULT FUNCTION TERMSANDCONDITIONS() {
  CONST [VISIBLE, SETVISIBLE] = USESTATE(FALSE);

  RETURN (
    <VIEW STYLE={STYLES.SCREEN}>
      {/* BACK BUTTON */}
      <TOUCHABLEOPACITY STYLE={STYLES.BACKBUTTON} ONPRESS={() =>
ROUTER.BACK()}>
        <IONICONS NAME="ARROW-BACK" SIZE={28} COLOR={ACCENT} />
      </TOUCHABLEOPACITY>

      {/* HEADER IMAGE */}
      <IMAGE
        SOURCE={REQUIRE("../../ASSETS/IMAGES/PLANTPOT5.PNG")}
        STYLE={STYLES.IMAGE}
      />

      {/* INFO SECTION */}
      <VIEW STYLE={STYLES.INFOSECTION}>
        <TEXT STYLE={STYLES.TITLE}>TERMS AND CONDITIONS</TEXT>
        <TEXT STYLE={STYLES.SUBTITLE}>PLEASE READ CAREFULLY</TEXT>
        <TEXT STYLE={STYLES.DESC}>
          THESE TERMS OUTLINE THE RULES AND REGULATIONS FOR USING THIS APPLICATION.
          BY ACCESSING THIS APP, YOU ACCEPT THESE TERMS IN FULL.
        </TEXT>
      </VIEW>

      {/* DECORATIVE LINE */}
      <IMAGE
        SOURCE={REQUIRE("../../ASSETS/IMAGES/PLANTLINE.PNG")}
        STYLE={STYLES.PLANTLINEIMAGE}
      />

      {/* BOTTOM CURVE SECTION */}
      <VIEW STYLE={STYLES.CURVECONTAINER}>
        <VIEW STYLE={STYLES.CURVESHAPE} />

        {/* READ BUTTON */}
        <TOUCHABLEOPACITY STYLE={STYLES.READBUTTON} ONPRESS={() =>
SETVISIBLE(TRUE)}>
          <TEXT STYLE={STYLES.READTEXT}>READ</TEXT>
    </TOUCHABLEOPACITY>
   </VIEW>

   {/* MODAL POPUP */}
   <MODAL VISIBLE={VISIBLE} ANIMATIONTYPE="FADE" TRANSPARENT>
    <VIEW STYLE={STYLES.MODALOVERLAY}>
     <VIEW STYLE={STYLES.MODALCARD}>
      <SCROLLVIEW SHOWSVERTICALSCROLLINDICATOR={FALSE}>
       <TEXT STYLE={STYLES.MODALTITLE}>TERMS & CONDITIONS</TEXT>
       <TEXT STYLE={STYLES.MODALTEXT}>
        1. YOU AGREE TO USE THIS APP RESPONSIBLY AND ETHICALLY. MISUSE MAY LEAD
TO RESTRICTED ACCESS.
        {"\N\N"}2. YOUR DATA WILL BE USED ONLY TO IMPROVE USER EXPERIENCE AND
APP PERFORMANCE.
        {"\N\N"}3. THE DEVELOPERS ARE NOT LIABLE FOR INDIRECT OR CONSEQUENTIAL
DAMAGES.
        {"\N\N"}4. TERMS MAY CHANGE WITHOUT PRIOR NOTICE. CONTINUED USE MEANS
YOU ACCEPT THEM.
       </TEXT>
      </SCROLLVIEW>

      <TOUCHABLEOPACITY STYLE={STYLES.CLOSEBUTTON} ONPRESS={() =>
SETVISIBLE(FALSE)}>
       <TEXT STYLE={STYLES.CLOSETEXT}>CLOSE</TEXT>
      </TOUCHABLEOPACITY>
     </VIEW>
    </VIEW>
   </MODAL>
  </VIEW>
 );
}

CONST STYLES = STYLESHEET.CREATE({
 SCREEN: {
  FLEX: 1,
  BACKGROUNDCOLOR: "#F8FAF5",
 },
 BACKBUTTON: {
  POSITION: "ABSOLUTE",
  TOP: 45,
  LEFT: 20,
  ZINDEX: 10,
  BACKGROUNDCOLOR: "#F0F7E8",
  PADDING: 8,
  BORDERRADIUS: 50,
  ELEVATION: 4,
},
IMAGE: {
 WIDTH: 600,
 HEIGHT: 600,
 RESIZEMODE: "CONTAIN",
 MARGINTOP: 50,
 ALIGNSELF: "CENTER",
},
INFOSECTION: {
 POSITION: "ABSOLUTE",
 BOTTOM: CURVE_HEIGHT + 30,
 WIDTH: "100%",
 PADDINGHORIZONTAL: 24,
},
TITLE: {
 FONTSIZE: 22,
 FONTWEIGHT: "700",
 COLOR: "#5A7721",
},
SUBTITLE: {
 FONTSIZE: 16,
 FONTWEIGHT: "600",
 COLOR: "#7BAE2B",
 MARGINBOTTOM: 6,
},
DESC: {
 FONTSIZE: 13,
 COLOR: "#4F4F4F",
 MARGINBOTTOM: 8,
},
CURVECONTAINER: {
 POSITION: "ABSOLUTE",
 LEFT: 0,
 BOTTOM: 0,
 WIDTH: "100%",
 HEIGHT: CURVE_HEIGHT,
 JUSTIFYCONTENT: "CENTER",
 ALIGNITEMS: "CENTER",
},
CURVESHAPE: {
 POSITION: "ABSOLUTE",
 BOTTOM: 0,
 WIDTH: "100%",
 HEIGHT: CURVE_HEIGHT,
 BACKGROUNDCOLOR: ACCENT,
 BORDERTOPLEFTRADIUS: 300,
  transform: [{ scaleX: 1.2 }],
  zIndex: 0,
 },
 readButton: {
  backgroundColor: "#fff",
  paddingVertical: 12,
  paddingHorizontal: 40,
  borderRadius: 30,
  shadowColor: "#000",
  shadowOpacity: 0.15,
  shadowRadius: 6,
  shadowOffset: { width: 0, height: 3 },
  elevation: 4,
  marginTop: -50,
 },
 readText: {
  color: ACCENT,
  fontWeight: "600",
  fontSize: 16,
 },
 modalOverlay: {
  flex: 1,
  backgroundColor: "rgba(0,0,0,0.4)",
  justifyContent: "center",
  alignItems: "center",
  padding: 20,
 },
 modalCard: {
  backgroundColor: "#fff",
  borderRadius: 20,
  width: width * 0.9,
  maxHeight: height * 0.75,
  padding: 20,
  shadowColor: "#000",
  shadowOpacity: 0.2,
  shadowRadius: 8,
  shadowOffset: { width: 0, height: 3 },
  elevation: 6,
 },
 modalTitle: {
  fontSize: 20,
  fontWeight: "700",
  color: ACCENT,
  marginBottom: 10,
 },
 modalText: {
    fontSize: 14,
    color: "#555",
    lineHeight: 22,
  },
  closeButton: {
    backgroundColor: ACCENT,
    paddingVertical: 12,
    borderRadius: 30,
    alignItems: "center",
    marginTop: 20,
  },
  closeText: {
    color: "#fff",
    fontWeight: "600",
    fontSize: 16,
  },
  plantLineImage: {
    width: width * 1.5,
    height: 180,
    resizeMode: "contain",
    position: "absolute",
    top: CURVE_HEIGHT + 540,
    left: -width * 0.25,
    zIndex: 10,
  },
});
```

TipsScreen

```
import React, { useRef } from "react";
import {
  View,
  Text,
  Animated,
  StyleSheet,
  TouchableOpacity,
} from "react-native";
import { useRouter } from "expo-router";
import Icon from "react-native-vector-icons/MaterialIcons";

export default function TipsScreen() {
  const router = useRouter();
  const scrollY = useRef(new Animated.Value(0)).current;

  const imageScale = scrollY.interpolate({
    inputRange: [0, 300],
    outputRange: [1, 1.3],
  extrapolate: "clamp",
 });

 return (
  <View style={styles.container}>
   <Animated.ScrollView
    showsVerticalScrollIndicator={false}
    onScroll={Animated.event(
     [{ nativeEvent: { contentOffset: { y: scrollY } } }],
     { useNativeDriver: true }
    )}
    scrollEventThrottle={16}
   >
    {/* HEADER */}
    <View style={styles.headerContainer}>
     <View style={styles.greenCurve} />
     <Animated.Image
      source={{
       uri: "https://images.unsplash.com/photo-1501004318641-
b39e6451bec6?q=80&w=2070&auto=format&fit=crop",
      }}
      style={[styles.image, { transform: [{ scale: imageScale }] }]}
     />
     <Text style={styles.headerText}>How to Use</Text>
    </View>

    {/* INFO SECTION */}
    <View style={styles.infoContainer}>
     <Text style={styles.title}>Smart Plant Tower Guide</Text>
     <Text style={styles.desc}>
      Learn how each feature of your Smart Plant Tower works to monitor
      and care for your plants efficiently.
     </Text>

     {/* NAVIGATION BAR SECTION */}
     <Text style={styles.sectionLabel}>Navigation Bar</Text>

     {/* MAIN DASHBOARD */}
     <View style={styles.card}>
      <View style={styles.cardHeader}>
       <Icon name="dashboard" size={22} color="#33691E" />
       <Text style={styles.cardTitle}>Main Dashboard</Text>
      </View>
      <Text style={styles.cardSubtitle}>Monitor and Control</Text>
      <Text style={styles.cardText}>
       Displays 4 key sensors — humidity, temperature, water level, and
    PH — FOR A QUICK AND CLEAR OVERVIEW OF YOUR SMART PLANT TOWER’S
    ENVIRONMENT AND STATUS.
  </TEXT>
</VIEW>

{/* EC SENSOR */}
<VIEW STYLE={STYLES.CARD}>
  <VIEW STYLE={STYLES.CARDHEADER}>
    <ICON NAME="SCIENCE" SIZE={22} COLOR="#33691E" />
    <TEXT STYLE={STYLES.CARDTITLE}>EC SENSORS</TEXT>
  </VIEW>
  <TEXT STYLE={STYLES.CARDSUBTITLE}>NUTRIENT MONITORING</TEXT>
  <TEXT STYLE={STYLES.CARDTEXT}>
    MEASURES THE NUTRIENT CONCENTRATION IN SOIL OR WATER. MAINTAINING
    PROPER EC LEVELS HELPS ENSURE YOUR PLANTS RECEIVE BALANCED
    NUTRIENTS WITHOUT OVERFEEDING.
  </TEXT>
</VIEW>

{/* WATER LEVEL */}
<VIEW STYLE={STYLES.CARD}>
  <VIEW STYLE={STYLES.CARDHEADER}>
    <ICON NAME="OPACITY" SIZE={22} COLOR="#33691E" />
    <TEXT STYLE={STYLES.CARDTITLE}>WATER LEVEL</TEXT>
  </VIEW>
  <TEXT STYLE={STYLES.CARDSUBTITLE}>TANK AND PUMP CONTROL</TEXT>
  <TEXT STYLE={STYLES.CARDTEXT}>
    PROVIDES REAL-TIME UPDATES OF YOUR TANK’S WATER LEVEL AND ALLOWS
    YOU TO ACTIVATE THE PUMP OR ENABLE AUTOMATIC WATERING DIRECTLY.
  </TEXT>
</VIEW>

{/* LIGHTING CONTROL */}
<VIEW STYLE={STYLES.CARD}>
  <VIEW STYLE={STYLES.CARDHEADER}>
    <ICON NAME="WB-SUNNY" SIZE={22} COLOR="#33691E" />
    <TEXT STYLE={STYLES.CARDTITLE}>LIGHTING CONTROL</TEXT>
  </VIEW>
  <TEXT STYLE={STYLES.CARDSUBTITLE}>ADJUST PLANT LIGHTING</TEXT>
  <TEXT STYLE={STYLES.CARDTEXT}>
    MANAGE THE BRIGHTNESS OF YOUR LED GROW LIGHTS. ADJUST INTENSITY TO
    PROVIDE OPTIMAL LIGHT CONDITIONS FOR PLANT GROWTH AND DEVELOPMENT.
  </TEXT>
</VIEW>

{/* ACCOUNT AND SETTINGS */}
```

```
          <View style={styles.card}>
            <View style={styles.cardHeader}>
              <Icon name="person" size={22} color="#33691E" />
              <Text style={styles.cardTitle}>Account & Settings</Text>
            </View>
            <Text style={styles.cardSubtitle}>Manage Your Profile</Text>
            <Text style={styles.cardText}>
              Access account details, adjust app preferences, manage passwords,
              and customize your Smart Plant Tower experience easily and safely.
            </Text>
          </View>

          {/* GO BACK BUTTON */}
          <TouchableOpacity style={styles.button} onPress={() => router.back()}>
            <Icon name="arrow-back" size={22} color="#fff" />
            <Text style={styles.buttonText}>Go Back</Text>
          </TouchableOpacity>
        </View>
      </Animated.ScrollView>
    </View>
  );
}

const styles = StyleSheet.create({
  container: { flex: 1, backgroundColor: "#fff" },
  headerContainer: { position: "relative", height: 270 },
  greenCurve: {
    position: "absolute",
    top: 0,
    right: 0,
    width: "75%",
    height: "100%",
    backgroundColor: "#8BC34A",
    borderBottomLeftRadius: 160,
  },
  image: {
    position: "absolute",
    top: 60,
    left: 25,
    width: 170,
    height: 170,
    borderRadius: 100,
    borderWidth: 3,
    borderColor: "#fff",
  },
  headerText: {
```

```
 position: "absolute",
 top: 50,
 right: 30,
 fontSize: 30,
 fontWeight: "bold",
 color: "#fff",
},
infoContainer: {
 backgroundColor: "#fff",
 marginTop: 10,
 borderTopLeftRadius: 40,
 borderTopRightRadius: 40,
 paddingHorizontal: 20,
 paddingTop: 30,
},
title: {
 fontSize: 24,
 fontWeight: "700",
 color: "#33691E",
 marginBottom: 10,
},
desc: { fontSize: 15, color: "#555", marginBottom: 25 },
sectionLabel: {
 fontSize: 18,
 fontWeight: "700",
 color: "#558B2F",
 marginBottom: 12,
},
card: {
 backgroundColor: "#fff",
 borderRadius: 18,
 padding: 18,
 marginBottom: 20,
 shadowColor: "#000",
 shadowOffset: { width: 0, height: 4 },
 shadowOpacity: 0.1,
 shadowRadius: 6,
 elevation: 3,
},
cardHeader: {
 flexDirection: "row",
 alignItems: "center",
 marginBottom: 6,
},
cardTitle: {
 fontSize: 17,
  fontWeight: "700",
  color: "#33691E",
  marginLeft: 8,
 },
 cardSubtitle: {
  fontSize: 14,
  color: "#689F38",
  marginBottom: 6,
  fontWeight: "600",
 },
 cardText: {
  fontSize: 14,
  color: "#444",
  lineHeight: 21,
 },
 button: {
  flexDirection: "row",
  alignItems: "center",
  justifyContent: "center",
  backgroundColor: "#8BC34A",
  borderRadius: 25,
  paddingVertical: 12,
  marginTop: 10,
  marginBottom: 40,
 },
 buttonText: {
  color: "#fff",
  fontSize: 16,
  fontWeight: "600",
  marginLeft: 8,
 },
});

_layout
import { Tabs } from "expo-router";
import MyTabBar from "../../components/customNavBar";

function TabsLayout(){
  return <Tabs tabBar={(props)=> <MyTabBar {...props} />}  screenOptions={{
headerShown: false,
    tabBarShowLabel: false,
  }}
  >
    <Tabs.Screen name="mainapp" options={{title:"home"}} />
    <Tabs.Screen name="Notifications" options={{title:"local-florist"}} />
    <Tabs.Screen name="WaterLevel" options={{title:"spa"}} />
    <Tabs.Screen name="LightingControl" options={{title:"flash-on"}} />
    <Tabs.Screen name="Settings" options={{title:"account-circle"}} />
  </Tabs>
}

export default TabsLayout;
LightingControl
import React, { useState, useEffect } from "react";
import { View, Text, StyleSheet, TouchableOpacity, ScrollView, Image,
Dimensions } from "react-native";
import Ionicons from "react-native-vector-icons/Ionicons";
import { db } from "../../config/firebaseConfig";
import { ref, set, onValue } from "firebase/database";
import { useNavigation } from "@react-navigation/native";

const { width } = Dimensions.get("window");
const CURVE_HEIGHT = 400;
const levels = [115, 124, 133, 142, 151, 160, 169, 178, 195, 255];

const mapValueToIndex = (val: number) => {
 let closest = 0;
 let minDiff = Infinity;
 levels.forEach((lvl, idx) => {
  const diff = Math.abs(val - lvl);
  if (diff < minDiff) {
   minDiff = diff;
   closest = idx;
  }
 });
 return closest;
};

const indexToProgress = (index: number) => index / (levels.length - 1);
const progressToValue = (progress: number) => {
 const index = Math.round(progress * (levels.length - 1));
 return levels[index];
};

const LightingSystem = () => {
 const navigation = useNavigation();
 const [progress, setProgress] = useState([0, 0, 0, 0]);

 const updateBrightness = (ledIndex: number, value: number) => {
  const path = ref(db, `leds/led${ledIndex + 1}`);
  set(path, progressToValue(value));
 };
 const increaseBrightness = (ledIndex: number) => {
  setProgress((prev) => {
   const newProgress = [...prev];
   newProgress[ledIndex] = Math.min(newProgress[ledIndex] + 0.1, 1);
   updateBrightness(ledIndex, newProgress[ledIndex]);
   return newProgress;
  });
 };

 const decreaseBrightness = (ledIndex: number) => {
  setProgress((prev) => {
   const newProgress = [...prev];
   newProgress[ledIndex] = Math.max(newProgress[ledIndex] - 0.1, 0);
   updateBrightness(ledIndex, newProgress[ledIndex]);
   return newProgress;
  });
 };

 useEffect(() => {
  const unsubscribes: any[] = [];
  for (let i = 0; i < 4; i++) {
   const path = ref(db, `leds/led${i + 1}`);
   const unsubscribe = onValue(path, (snapshot) => {
    const val = snapshot.val();
    if (val !== null) {
     setProgress((prev) => {
      const newProgress = [...prev];
      newProgress[i] = indexToProgress(mapValueToIndex(val));
      return newProgress;
     });
    }
   });
   unsubscribes.push(unsubscribe);
  }
  return () => unsubscribes.forEach((u) => u());
 }, []);

 return (
  <ScrollView style={styles.screen} contentContainerStyle={{
paddingBottom: 20 }}>
   {/* Back Button */}
   <TouchableOpacity style={styles.backBtn} onPress={() =>
navigation.goBack()}>
    <View style={styles.backCircle}>
     <Ionicons name="arrow-back" size={22} color="#fff" />
```

```
      </View>
    </TouchableOpacity>

    {/* Plant Image on Top */}
    <View style={styles.plantImageContainer}>
      <Image
        source={require("../../assets/images/imgplant2.png")}
        style={styles.plantImage}
      />
    </View>

    {/* Info Section */}
    <View style={styles.infoSection}>
      <Text style={styles.title}>About Lighting System</Text>
      <Text style={styles.subtitle}>Control the brightness of your plant
lights</Text>
      <Text style={styles.desc}>
        Adjust the intensity of each LED to optimize plant growth. Dim or
brighten each light and monitor real-time progress.
      </Text>
    </View>

    {/* Green Curve with LEDs */}
    <View style={styles.curveContainer}>
      <View style={styles.curveShape} />
      <View style={styles.statsRow}>
        {progress.map((ledProgress, i) => (
          <View key={i} style={styles.statItem}>
            <Ionicons
              name="sunny"
              size={35}
              color={ledProgress > 0 ? "#FFD700" : "#FFFFFF99"}
              style={{ marginBottom: 4 }}
            />
            <Text style={styles.statLabel}>LED {i + 1}</Text>
            <Text style={styles.statValue}>{Math.round(ledProgress *
100)}%</Text>

            <View style={styles.buttonRow}>
              <TouchableOpacity
                style={[styles.controlBtn, { backgroundColor: "#ffffff33",
paddingVertical: 4, paddingHorizontal: 8 }]}
                onPress={() => decreaseBrightness(i)}
              >
                <Ionicons name="remove" size={14} color="#fff" />
              </TouchableOpacity>
            <TouchableOpacity
             style={[styles.controlBtn, { backgroundColor: "#ffffff",
paddingVertical: 4, paddingHorizontal: 8 }]}
             onPress={() => increaseBrightness(i)}
            >
             <Ionicons name="add" size={14} color="#7CB342" />
            </TouchableOpacity>
           </View>
          </View>
         ))}
        </View>
       </View>
  </ScrollView>
 );
};

const styles = StyleSheet.create({
 screen: { flex: 1, backgroundColor: "#F0F8F1" },

 backBtn: {
  position: "absolute",
  top: 45,
  left: 20,
  zIndex: 20,
 },
 backCircle: {
  backgroundColor: "rgba(124, 179, 66, 0.3)",
  width: 40,
  height: 40,
  borderRadius: 20,
  justifyContent: "center",
  alignItems: "center",
 },

 plantImageContainer: { width: "100%", alignItems: "center", marginTop: 80,
marginBottom: 16 },
 plantImage: { width: 350, height: 350, resizeMode: "contain", borderRadius: 16
},

 infoSection: { paddingHorizontal: 24, marginBottom: -100 },
 title: { fontSize: 22, fontWeight: "700", color: "#33691E", marginBottom: 4 },
 subtitle: { fontSize: 16, fontWeight: "600", color: "#558B2F", marginBottom: 6
},
 desc: { fontSize: 13, color: "#4F4F4F", marginBottom: 16 },

 curveContainer: {
    width: "100%",
    height: 500,
    justifyContent: "center",
    alignItems: "center",
  },
  curveShape: {
    position: "absolute",
    bottom: 0,
    width: 400,
    height: 400,
    backgroundColor: "#7CB342",
    borderTopLeftRadius: 200,
    transform: [{ scaleX: 1.2 }],
    zIndex: 0,
  },
  statsRow: {
    flexDirection: "row",
    justifyContent: "space-around",
    width: "100%",
    zIndex: 0,
    marginTop: -40,
  },
  statItem: { alignItems: "center" },
  statLabel: { fontSize: 12, color: "#fff", fontWeight: "600" },
  statValue: { fontSize: 12, color: "#fff", marginTop: 1 },
  buttonRow: { flexDirection: "row", marginTop: 4 },
  controlBtn: { borderRadius: 50, marginHorizontal: 4 },
});

export default LightingSystem;
```

MainApp

```
import React, { useState, useRef, useEffect } from "react";
import {
  View,
  Text,
  TextInput,
  Image,
  ScrollView,
  TouchableOpacity,
  Animated,
  Dimensions,
  SafeAreaView,
  StyleSheet,
  Modal,
  ActivityIndicator,
} FROM "REACT-NATIVE";
IMPORT { USEROUTER } FROM "EXPO-ROUTER";
IMPORT ICON FROM "REACT-NATIVE-VECTOR-ICONS/MATERIALICONS";
IMPORT { AUTH, DB } FROM "../../CONFIG/FIREBASECONFIG";
IMPORT { REF, ONVALUE } FROM "FIREBASE/DATABASE";
IMPORT { BLURVIEW } FROM "EXPO-BLUR";
IMPORT STYLES FROM "../../STYLES/MAINAPPSTYLES";

CONST { WIDTH } = DIMENSIONS.GET("WINDOW");
CONST ITEM_WIDTH = WIDTH * 0.7;
CONST SPACING = 10;

INTERFACE NOTIFICATIONITEM {
  ID: STRING;
  TYPE: STRING;
  STATUS: STRING;
  COLOR: STRING;
}

EXPORT DEFAULT FUNCTION MAINAPP() {
  CONST ROUTER = USEROUTER();
  CONST [ACTIVETAB, SETACTIVETAB] = USESTATE("HOME");
  CONST SCROLLY = USEREF(NEW ANIMATED.VALUE(0)).CURRENT;
  CONST SCROLLX = USEREF(NEW ANIMATED.VALUE(0)).CURRENT;
  CONST SCROLLXREF = USEREF(0);
  CONST SCROLLREF = USEREF<SCROLLVIEW | NULL>(NULL);
  CONST [USERNAME, SETUSERNAME] = USESTATE("");
  CONST [SEARCHQUERY, SETSEARCHQUERY] = USESTATE("");

  CONST [HUMIDITY, SETHUMIDITY] = USESTATE<NUMBER | NULL>(NULL);
  CONST [TEMPERATURE, SETTEMPERATURE] = USESTATE<NUMBER | NULL>(NULL);
  CONST [PH, SETPH] = USESTATE<NUMBER | NULL>(NULL);
  CONST [TANKSTATUS, SETTANKSTATUS] = USESTATE<STRING>("");

  CONST [NOTIFICATIONS, SETNOTIFICATIONS] = USESTATE<NOTIFICATIONITEM[]>([]);
  CONST [LOADINGNOTIFICATIONS, SETLOADINGNOTIFICATIONS] = USESTATE(TRUE);
  CONST [MODALVISIBLE, SETMODALVISIBLE] = USESTATE(FALSE);

  USEEFFECT(() => {
    CONST USER = AUTH.CURRENTUSER;
    IF (USER) SETUSERNAME(USER.DISPLAYNAME || "GUEST");
  }, []);

  USEEFFECT(() => {
    CONST REFS = {
      HUMIDITY: REF(DB, "ENVIRONMENT/HUMIDITY"),
    TEMP: REF(DB, "ENVIRONMENT/TEMP"),
    PH: REF(DB, "ENVIRONMENT/PH"),
    TANK: REF(DB, "IRRIGATION/WATERTANK/STATUS"),
   };

   CONST UNSUBSCRIBERS = [
    ONVALUE(REFS.HUMIDITY, (SNAP) => SETHUMIDITY(SNAP.VAL())),
    ONVALUE(REFS.TEMP, (SNAP) => SETTEMPERATURE(SNAP.VAL())),
    ONVALUE(REFS.PH, (SNAP) => SETPH(SNAP.VAL())),
    ONVALUE(REFS.TANK, (SNAP) => SETTANKSTATUS(SNAP.VAL())),
   ];

   RETURN () => UNSUBSCRIBERS.FOREACH((UNSUB) => UNSUB());
  }, []);

  CONST SENSORDATA = [
   {
    ID: 1,
    NAME: "HUMIDITY",
    IMG: "HTTPS://MIRO.MEDIUM.COM/V2/RESIZE:FIT:1400/0*A_U0CNMQW_QKDATI",
    ROUTE: "/(SCREENS)/HUMIDITY" AS CONST,
    ISUNHEALTHY: HUMIDITY !== NULL && (HUMIDITY < 40 || HUMIDITY > 80),
   },
   {
    ID: 2,
    NAME: "TEMPERATURE",
    IMG: "HTTPS://IMAGES.UNSPLASH.COM/PHOTO-1533038590840-
1CDE6E668A91?Q=80&W=1974&AUTO=FORMAT&FIT=CROP",
    ROUTE: "/(SCREENS)/TEMPERATURE" AS CONST,
    ISUNHEALTHY: TEMPERATURE !== NULL && (TEMPERATURE < 20 || TEMPERATURE >
32),
   },
   {
    ID: 3,
    NAME: "WATER LEVEL",
    IMG: "HTTPS://IMAGES.UNSPLASH.COM/PHOTO-1586817100452-
0B6DEFC0D2B5?Q=80&W=1974&AUTO=FORMAT&FIT=CROP",
    ROUTE: "/(TABS)/WATERLEVEL" AS CONST,
    ISUNHEALTHY: TANKSTATUS === "LOW" || TANKSTATUS === "EMPTY",
   },
   {
    ID: 4,
    NAME: "PH LEVEL",
    IMG: "HTTPS://IMAGES.UNSPLASH.COM/PHOTO-1536882240095-
0379873FEB4E?Q=80&W=1171&AUTO=FORMAT&FIT=CROP",
    ROUTE: "/(SCREENS)/PH" AS CONST,
    isUnhealthy: ph !== null && (ph < 5.5 || ph > 7.5),
   },
  ];

  const hasUnhealthySensor = sensorData.some((sensor) =>
sensor.isUnhealthy);
  const [filteredSensors, setFilteredSensors] = useState(sensorData);

  useEffect(() => {
   const updatedNotifications: NotificationItem[] = [];

   if (humidity !== null && (humidity < 40 || humidity > 80))
    updatedNotifications.push({
     id: "humidity",
     type: "Humidity",
     status: humidity < 40 ? "Low" : "High",
     color: humidity < 40 ? "#FDD835" : "#E53935",
    });

   if (temperature !== null && (temperature < 20 || temperature > 32))
    updatedNotifications.push({
     id: "temperature",
     type: "Temperature",
     status: temperature < 20 ? "Too Cold" : "Too Hot",
     color: temperature < 20 ? "#039BE5" : "#E53935",
    });

   if (tankStatus === "LOW" || tankStatus === "EMPTY")
    updatedNotifications.push({
     id: "tank",
     type: "Water Tank",
     status: tankStatus,
     color: "#E53935",
    });

   if (ph !== null && (ph < 5.5 || ph > 7.5))
    updatedNotifications.push({
     id: "ph",
     type: "pH Level",
     status: `Unhealthy (${ph})`,
     color: "#E53935",
    });

   setNotifications(updatedNotifications);
   setLoadingNotifications(false);
  }, [humidity, temperature, tankStatus, ph]);
```

```
// Filter sensors
useEffect(() => {
 const filtered = sensorData.filter((sensor) =>
  sensor.name.toLowerCase().includes(searchQuery.toLowerCase())
 );
 setFilteredSensors(filtered);
}, [searchQuery, humidity, temperature, ph, tankStatus]);

// Auto-scroll animation
useEffect(() => {
 const listener = scrollX.addListener(({ value }) => {
  scrollXRef.current = value;
 });
 return () => scrollX.removeListener(listener);
}, []);

useEffect(() => {
 const interval = setInterval(() => {
  if (scrollRef.current) {
   const nextOffset = scrollXRef.current + ITEM_WIDTH + SPACING;
   if (nextOffset >= filteredSensors.length * (ITEM_WIDTH + SPACING)) {
    scrollRef.current?.scrollTo({ x: 0, animated: false });
   } else {
    scrollRef.current?.scrollTo({ x: nextOffset, animated: true });
   }
  }
 }, 3000);
 return () => clearInterval(interval);
}, [filteredSensors]);

const handleScroll = Animated.event(
 [{ nativeEvent: { contentOffset: { x: scrollX } } }],
 { useNativeDriver: false }
);

return (
 <SafeAreaView style={styles.container}>
  <View style={styles.container}>
   <Animated.View
    style={[styles.header, { backgroundColor: "#fff" }]}
   >
    <View style={styles.topRow}>
     <TouchableOpacity
      style={styles.profileRow}
      onPress={() => console.log("Profile tapped")}
      >
       <IMAGE
         SOURCE={{
           URI: "HTTPS://CDN-ICONS-
PNG.FREEPIK.COM/256/847/847969.PNG?GA=GA1.1.819490133.1749625733",
          }}
         STYLE={STYLES.PROFILEPIC}
       />
       <TEXT STYLE={STYLES.WELCOMETEXT}>WELCOME, {USERNAME}</TEXT>
      </TOUCHABLEOPACITY>
    </VIEW>

    <TOUCHABLEOPACITY
      STYLE={STYLES.NOTIFICATIONICONCONTAINER}
      ONPRESS={() => SETMODALVISIBLE(TRUE)}
    >
      <VIEW STYLE={STYLES.NOTIFICATIONCIRCLE}>
        <ICON NAME="NOTIFICATIONS-NONE" SIZE={24} COLOR="#AAA" />
      </VIEW>
      {HASUNHEALTHYSENSOR && <VIEW STYLE={STYLES.NOTIFICATIONDOT} />}
    </TOUCHABLEOPACITY>

    <VIEW STYLE={STYLES.SEARCHROW}>
      <VIEW STYLE={STYLES.SEARCHBARCONTAINER}>
        <ICON
          NAME="SEARCH"
          SIZE={24}
          COLOR="#AAA"
          STYLE={STYLES.SEARCHICON}
        />
        <TEXTINPUT
          STYLE={STYLES.SEARCHBAR}
          PLACEHOLDER="TYPE SENSOR NAME HERE..."
          PLACEHOLDERTEXTCOLOR="#AAA"
          VALUE={SEARCHQUERY}
          ONCHANGETEXT={SETSEARCHQUERY}
        />
      </VIEW>
    </VIEW>
  </ANIMATED.VIEW>

  {/* HIGHLIGHT SECTION */}
  <VIEW STYLE={STYLES.GIFTSECTION}>
    <IMAGE
      SOURCE={{
        uri: "https://www.pngarts.com/files/10/Flower-Pot-Transparent-
Background-PNG.png",
       }}
       style={styles.giftImage}
      />
      <View style={styles.giftTextContainer}>
       <Text style={styles.giftTitle}>
        Support your plant’s growth with smart illumination
       </Text>
       <TouchableOpacity
        style={styles.giftButton}
        onPress={() => router.push("/(tabs)/LightingControl")}
       >
        <Text style={styles.giftButtonText}>Adjust</Text>
       </TouchableOpacity>
      </View>
     </View>

     {/* Content */}
     <ScrollView
      style={styles.content}
      contentContainerStyle={{ flexGrow: 1 }}
      showsVerticalScrollIndicator={false}
     >
      <Text style={styles.sectionTitle}>All My Sensors</Text>
      <ScrollView horizontal showsHorizontalScrollIndicator={false}>
       {filteredSensors.length > 0 ? (
        filteredSensors.map((plant) => (
         <TouchableOpacity key={plant.id} style={styles.plantCard}>
          <View style={styles.imageContainer}>
           <Image
            source={{ uri: plant.img }}
            style={styles.plantImage}
           />
           {plant.isUnhealthy && <View style={styles.statusDot} />}
          </View>
          <View style={styles.plantInfo}>
           <Text style={styles.plantName}>{plant.name}</Text>
           <TouchableOpacity
            onPress={() => router.push(plant.route)}
            style={styles.arrowButton}
           >
            <Icon name="arrow-forward-ios" size={16} color="#9ACD32" />
           </TouchableOpacity>
          </View>
         </TouchableOpacity>
      ))
    ) : (
      <Text style={{ marginLeft: 20, color: "#666", marginTop: 10 }}>
        No sensors found.
      </Text>
    )}
  </ScrollView>

  {/* Learn More Section */}
  <Text style={styles.sectionTitle}>Learn More</Text>
  <ScrollView
    style={styles.learnMoreContainer}
    horizontal
    showsHorizontalScrollIndicator={false}
  >
    <TouchableOpacity
      style={styles.largeWidget}
      onPress={() => router.push("/AboutScreen")}
    >
      <Icon name="info" size={60} color="#689F38" />
      <Text style={styles.largeWidgetTitle}>About</Text>
      <Text style={styles.largeWidgetText}>
        Learn about plant care, watering schedules, and more.
      </Text>
    </TouchableOpacity>

    <TouchableOpacity
      style={styles.largeWidget}
      onPress={() => router.push("/TipsScreen")}
    >
      <Icon name="lightbulb" size={60} color="#FFB300" />
      <Text style={styles.largeWidgetTitle}>Tips</Text>
      <Text style={styles.largeWidgetText}>
        Get helpful tips on plant maintenance and growth.
      </Text>
    </TouchableOpacity>
  </ScrollView>
</ScrollView>

{/* Notification Overlay */}
<Modal
  visible={modalVisible}
  transparent
  animationType="fade"
  onRequestClose={() => setModalVisible(false)}
>
```

```
      <View style={overlayStyles.overlayContainer}>
       <BlurView intensity={80} tint="light" style={StyleSheet.absoluteFill}
/>
       <TouchableOpacity
        style={StyleSheet.absoluteFillObject}
        activeOpacity={1}
        onPress={() => setModalVisible(false)}
       />

       <View style={overlayStyles.modalCard}>
        <Text style={overlayStyles.modalTitle}>Notifications</Text>

        {loadingNotifications ? (
         <ActivityIndicator size="large" color="#8BC34A" style={{ marginTop:
10 }} />
        ) : notifications.length > 0 ? (
         <ScrollView style={{ width: "100%" }} contentContainerStyle={{
paddingVertical: 8 }}>
          {notifications.map((item) => (
           <View key={item.id} style={overlayStyles.notificationCard}>
            <Icon name="warning" size={22} color={item.color}
style={overlayStyles.icon} />
            <View style={{ flex: 1 }}>
             <Text style={overlayStyles.notificationTitle}>{item.type}</Text>
             <Text style={[overlayStyles.notificationText, { color: item.color
}]}>{item.status}</Text>
            </View>
           </View>
          ))}
         </ScrollView>
        ) : (
         <Text style={overlayStyles.emptyText}>All sensors are healthy
🌿</Text>
        )}

        <TouchableOpacity style={overlayStyles.modalClose} onPress={() =>
setModalVisible(false)}>
         <Text style={{ color: "#fff", fontWeight: "700" }}>Close</Text>
        </TouchableOpacity>
       </View>
      </View>
     </Modal>
    </View>
   </SafeAreaView>
 );
}
```

```
const overlayStyles = StyleSheet.create({
  overlayContainer: {
    ...StyleSheet.absoluteFillObject,
    justifyContent: "center",
    alignItems: "center",
    zIndex: 1000,
  },
  modalCard: {
    width: "86%",
    maxHeight: "72%",
    backgroundColor: "#fff",
    borderRadius: 14,
    padding: 14,
    elevation: 8,
    shadowColor: "#000",
    shadowOpacity: 0.16,
    shadowOffset: { width: 0, height: 6 },
    shadowRadius: 12,
  },
  modalTitle: { fontSize: 18, fontWeight: "700", marginBottom: 8, color:
"#263238" },
  modalClose: {
    marginTop: 12,
    backgroundColor: "#8BC34A",
    paddingVertical: 10,
    borderRadius: 10,
    alignItems: "center",
  },
  notificationCard: {
    flexDirection: "row",
    alignItems: "center",
    backgroundColor: "#fff",
    borderRadius: 12,
    padding: 12,
    marginBottom: 10,
    shadowColor: "#000",
    shadowOpacity: 0.08,
    shadowOffset: { width: 0, height: 2 },
    shadowRadius: 5,
    elevation: 2,
  },
  icon: { marginRight: 12 },
  notificationTitle: { fontSize: 16, fontWeight: "bold", color: "#333" },
  notificationText: { fontSize: 14, marginTop: 3 },
  emptyText: { textAlign: "center", color: "#777", fontSize: 16, marginTop: 20 },
});
```

EC Dashboard

```
import React, { useEffect, useState } from "react";
import { View, Text, StyleSheet, Image, TouchableOpacity, Dimensions,
ActivityIndicator, ScrollView } from "react-native";
import Ionicons from "react-native-vector-icons/Ionicons";
import { getDatabase, ref, onValue } from "firebase/database";
import { router } from "expo-router";

const { height, width } = Dimensions.get("window");
const CURVE_HEIGHT = 200;

export default function ECSensors() {
 const [data, setData] = useState({
  temperature: 0,
  voltage: 0,
  ec: 0,
  tds: 0,
 });
 const [loading, setLoading] = useState(true);

 useEffect(() => {
  const db = getDatabase();
  const sensorsRef = ref(db, "sensors/");

  const unsubscribe = onValue(sensorsRef, (snapshot) => {
   if (snapshot.exists()) {
    const val = snapshot.val();
    setData({
     temperature: val.temperature ?? 0,
     voltage: val.voltage ?? 0,
     ec: val.ec ?? 0,
     tds: val.tds ?? 0,
    });
   }
   setLoading(false);
  });

  return () => unsubscribe();
 }, []);

 if (loading) {
  return (
   <View style={[styles.screen, { justifyContent: "center", alignItems:
"center" }]}>
```

```
      <ActivityIndicator size="large" color="#33691E" />
    </View>
  );
 }

 return (
  <ScrollView style={styles.screen}
showsVerticalScrollIndicator={false}>
    {/* Back Button */}
    <TouchableOpacity style={styles.backButton} onPress={() =>
router.back()}>
      <Ionicons name="arrow-back" size={28} color="#33691E" />
    </TouchableOpacity>

    {/* Top Image */}
    <Image source={require("../../assets/images/imgplant2.png")}
style={styles.image} />

    {/* Info Section */}
    <View style={styles.infoSection}>
     <Text style={styles.title}>About EC Sensors</Text>
     <Text style={styles.subtitle}>Water Quality for Healthy Plants</Text>
     <Text style={styles.desc}>
      The EC sensor helps ensure your plant water has the right nutrient
concentration. Monitoring EC, TDS, and voltage keeps your plants hydrated
and safe.
     </Text>
    </View>



          {/* 🌿 Horizontal Plant Line Image */}
          <Image
           source={require("../../assets/images/plantline.png")}
           style={styles.plantLineImage}
          />

    {/* Green Curve Background */}
    <View style={styles.curveContainer}>
     <View style={styles.curveShape} />

     {/* Stats just below the curve */}
     <View style={styles.statsRow}>
      {[
       { icon: "water-outline", label: "EC", value: `${data.ec.toFixed(2)} µS/cm`
},
        { icon: "speedometer-outline", label: "TDS", value:
`${data.tds.toFixed(2)} ppm` },
        { icon: "thermometer-outline", label: "Temp", value:
`${data.temperature.toFixed(2)} °C` },
        { icon: "flash-outline", label: "Volt", value: `${data.voltage.toFixed(2)}
V` },
       ].map((item, index) => (
        <View key={index} style={styles.statCircle}>
         <Ionicons name={item.icon} size={22} color="#fff" />
         <Text style={styles.statLabel}>{item.label}</Text>
         <Text style={styles.statValue}>{item.value}</Text>
        </View>
       ))}
      </View>
     </View>
   </ScrollView>
 );
}

const styles = StyleSheet.create({
 screen: {
  flex: 1,
  backgroundColor: "#F0F8F1",
 },
 backButton: {
  position: "absolute",
  top: 45,
  left: 20,
  zIndex: 10,
  backgroundColor: "#E8F5E9",
  padding: 8,
  borderRadius: 50,
  elevation: 4,
 },
 image: {
  width: "100%",
  height:350,
  resizeMode: "contain",
  marginTop: 100,
 },
 infoSection: {
  paddingHorizontal: 24,
  marginTop: 10,
 },
 title: {
  fontSize: 22,
  fontWeight: "700",
  color: "#33691E",
 },
 subtitle: {
  fontSize: 16,
  fontWeight: "600",
  color: "#558B2F",
  marginBottom: 6,
 },
 desc: {
  fontSize: 13,
  color: "#4F4F4F",
  marginBottom: 20,
 },
 curveContainer: {
  width: "100%",
  height: height * 0.5,
  justifyContent: "flex-end",
  alignItems: "center",
 },
 curveShape: {
  position: "absolute",
  bottom: 0,
  width: "100%",
  height: "100%",
  backgroundColor: "#7CB342",
  borderTopLeftRadius: 200,
  borderTopRightRadius: 0,
  transform: [{ scaleX: 1.2 }],
  zIndex: 0,
 },
 statsRow: {
  flexDirection: "row",
  alignItems: "center",
  justifyContent: "space-evenly",
  width: "100%",
  marginBottom: 270,
  zIndex: 1,
 },
 statCircle: {
  width: 70,
  height: 70,
  borderRadius: 35,
  backgroundColor: "rgba(255,255,255,0.25)",
  justifyContent: "center",
  alignItems: "center",
 },
 statLabel: {
  fontSize: 11,
  color: "#fff",
  fontWeight: "600",
  marginTop: 2,
 },
 statValue: {
  fontSize: 10,
  color: "#fff",
  marginTop: 1,
 },

plantLineImage: {
 width: width * 1.5,
 height: 180,
 resizeMode: "contain",
 position: "absolute",
 top: CURVE_HEIGHT + 510,
 left: -width * 0.25,
 zIndex: 10,
},
});

WaterLevel
import React, { useEffect, useState, useRef } from "react";
import {
 View,
 Text,
 TouchableOpacity,
 ScrollView,
 StyleSheet,
 Image,
 Dimensions,
} from "react-native";
import Ionicons from "react-native-vector-icons/Ionicons";
import { db } from "../../config/firebaseConfig";
import { ref, onValue, set } from "firebase/database";
import { router } from "expo-router";

const { width, height } = Dimensions.get("window");
const CURVE_HEIGHT = 180;
const ACCENT = "#7CB342";
const ACCENT_DARK = "#33691E";
const CARD_RADIUS = 12;
```

```
interface Pot {
 id: number;
 status: string;
 duration: number;
 interval: number;
}

export default function WateringSystem() {
 const [env, setEnv] = useState({
  temp: 0,
  humidity: 0,
  distance: 0,
  tankStatus: "UNKNOWN",
 });

 const [autoWatering, setAutoWatering] = useState(false);
 const [pump1, setPump1] = useState(false);
 const [pump2, setPump2] = useState(false);

 const [pots, setPots] = useState<Pot[]>(
  Array.from({ length: 12 }, (_, i) => ({
   id: i + 1,
   status: "idle",
   duration: 0,
   interval: 0,
  }))
 );

 const wateringTimeouts = useRef<{ [key: number]: ReturnType<typeof
setTimeout> | undefined }>({});
 const intervalTimeouts = useRef<{ [key: number]: ReturnType<typeof
setTimeout> | undefined }>({});

 useEffect(() => {
  onValue(ref(db, "/irrigation/environment/temp"), (snap) =>
   setEnv((p) => ({ ...p, temp: snap.val() ?? 0 }))
  );
  onValue(ref(db, "/irrigation/environment/humidity"), (snap) =>
   setEnv((p) => ({ ...p, humidity: snap.val() ?? 0 }))
  );
  onValue(ref(db, "/irrigation/watertank/distance"), (snap) =>
   setEnv((p) => ({ ...p, distance: snap.val() ?? 0 }))
  );
  onValue(ref(db, "/irrigation/watertank/status"), (snap) =>
   setEnv((p) => ({ ...p, tankStatus: snap.val() ?? "UNKNOWN" }))
  );
  onValue(ref(db, "/irrigation/autoMode"), (snap) =>
   setAutoWatering(snap.val() ?? false)
  );

  onValue(ref(db, "/irrigation/pump1"), (snap) => setPump1(snap.val() ?? false));
  onValue(ref(db, "/irrigation/pump2"), (snap) => setPump2(snap.val() ?? false));
 }, []);

 useEffect(() => {
  pots.forEach((p) => {
   const base = p.id <= 6 ? `/irrigation/pots/pot${p.id}` : `/pots/pot${p.id}`;

   onValue(ref(db, `${base}/status`), (snap) => {
    const status = snap.val() ?? "idle";
    setPots((prev) => prev.map((x) => (x.id === p.id ? { ...x, status } : x)));
   });

   onValue(ref(db, `${base}/manualDuration`), (snap) => {
    const val = snap.val();
    if (val !== null && val !== undefined) {
     setPots((prev) => prev.map((x) => (x.id === p.id ? { ...x, duration: val } : x)));
    }
   });

   onValue(ref(db, `${base}/interval`), (snap) => {
    const val = snap.val();
    if (val !== null && val !== undefined) {
     setPots((prev) => prev.map((x) => (x.id === p.id ? { ...x, interval: val } : x)));
    }
   });
  });
 }, []);

 const waterPot = (id: number, on: boolean) => {
  const pot = pots.find((p) => p.id === id);
  if (!pot) return;
  const base = id <= 6 ? `/irrigation/pots/pot${id}` : `/pots/pot${id}`;
  set(ref(db, `${base}/manual`), on);
 };

 const waterAll = (on: boolean) => {
  set(ref(db, "/irrigation/waterAll"), on);
  pots.forEach((pot) => {
   const base = pot.id <= 6 ? `/irrigation/pots/pot${pot.id}` : `/pots/pot${pot.id}`;
   set(ref(db, `${base}/manual`), on);
  });
 };

 CONST TOGGLEAUTOWATERING = () => {
  CONST NEWVAL = !AUTOWATERING;
  SET(REF(DB, "/IRRIGATION/AUTOMODE"), NEWVAL);
  SETAUTOWATERING(NEWVAL);
 };

 CONST TOGGLEPUMP1 = () => {
  CONST NEWVAL = !PUMP1;
  SET(REF(DB, "/IRRIGATION/PUMP1"), NEWVAL);
  SETPUMP1(NEWVAL);
 };

 CONST TOGGLEPUMP2 = () => {
  CONST NEWVAL = !PUMP2;
  SET(REF(DB, "/IRRIGATION/PUMP2"), NEWVAL);
  SETPUMP2(NEWVAL);
 };

 CONST ADJUSTDURATION = (ID: NUMBER, DELTA: NUMBER) => {
 SETPOTS((PREV) =>
  PREV.MAP((P) => {
   IF (P.ID === ID) {
    CONST NEWDUR = MATH.MAX(0, MATH.MIN(600, P.DURATION + DELTA));
    CONST BASE = ID <= 6 ? `/IRRIGATION/POTS/POT${ID}` : `/POTS/POT${ID}`;
    SET(REF(DB, `${BASE}/MANUALDURATION`), NEWDUR);
    RETURN { ...P, DURATION: NEWDUR };
   }
   RETURN P;
  })
 );
};

CONST ADJUSTINTERVAL = (ID: NUMBER, DELTA: NUMBER) => {
 SETPOTS((PREV) =>
  PREV.MAP((P) => {
   IF (P.ID === ID) {
    CONST NEWINT = MATH.MAX(0, MATH.MIN(3600, P.INTERVAL + DELTA));
    CONST BASE = ID <= 6 ? `/IRRIGATION/POTS/POT${ID}` : `/POTS/POT${ID}`;
    SET(REF(DB, `${BASE}/INTERVAL`), NEWINT);
    RETURN { ...P, INTERVAL: NEWINT };
   }
   RETURN P;
  })
 );
```

```
};

 const startWaterCycle = (id: number) => {
  const pot = pots.find((p) => p.id === id);
  if (!pot) return;

  const base = id <= 6 ? `/irrigation/pots/pot${id}` : `/pots/pot${id}`;
  if (wateringTimeouts.current[id]) return;

  set(ref(db, `${base}/manual`), true);

  wateringTimeouts.current[id] = setTimeout(() => {
   set(ref(db, `${base}/manual`), false);

   intervalTimeouts.current[id] = setTimeout(() => {
    wateringTimeouts.current[id] = undefined;
    intervalTimeouts.current[id] = undefined;
   }, pot.interval * 1000);
  }, pot.duration * 1000);
 };

 return (
  <View style={styles.screen}>
   <TouchableOpacity style={styles.backButton} onPress={() =>
router.back()}>
    <Ionicons name="arrow-back" size={26} color={ACCENT_DARK} />
   </TouchableOpacity>

   <ScrollView contentContainerStyle={styles.scrollContent}>
    <Image source={require("../../assets/images/plantpot4.png")}
style={styles.image} />

    <View style={styles.infoSection}>
     <Text style={styles.title}>About Watering System</Text>
     <Text style={styles.subtitle}>Automated Care for Your Plants</Text>
     <Text style={styles.desc}>
      The watering system ensures your plants receive the right amount of
water.
      You can manually water individual pots or enable automatic watering
to maintain balanced soil moisture.
     </Text>
    </View>

    <View style={styles.curveContainer}>
     <View style={styles.curveShape} />
     <Image
  source={require("../../assets/images/plantline.png")}
  style={styles.plantLineImage}
 />

 <View style={styles.statsRow}>
  <View style={styles.statItem}>
   <View style={styles.statIconCircle}>
    <Ionicons name="thermometer-outline" size={22} color="#fff" />
   </View>
   <Text style={styles.statLabel}>Temp</Text>
   <Text style={styles.statValue}>{env.temp.toFixed(1)}°C</Text>
  </View>

  <View style={styles.statItem}>
   <View style={styles.statIconCircle}>
    <Ionicons name="water-outline" size={22} color="#fff" />
   </View>
   <Text style={styles.statLabel}>Humidity</Text>
   <Text style={styles.statValue}>{env.humidity.toFixed(1)}%</Text>
  </View>

  <View style={styles.statItem}>
   <View style={styles.statIconCircle}>
    <Ionicons name="speedometer-outline" size={22} color="#fff" />
   </View>
   <Text style={styles.statLabel}>Water Level</Text>
   <Text style={styles.statValue}>{env.distance.toFixed(1)} cm</Text>
  </View>

  <View style={styles.statItem}>
   <View style={styles.statIconCircle}>
    <Ionicons name="cube-outline" size={22} color="#fff" />
   </View>
   <Text style={styles.statLabel}>Status</Text>
   <Text style={styles.statValue}>{env.tankStatus}</Text>
  </View>
 </View>
</View>

<Text style={styles.sectionTitle}>Water Each Pot</Text>
<View style={styles.potsGrid}>
 {pots.map((p) => (
  <View key={p.id} style={styles.potCard}>
   <View style={styles.potHeader}>
    <Text style={styles.potTitle}>Pot {p.id}</Text>
    <Text style={[styles.potStatus, { color: ACCENT_DARK }]}>
```

```
         {P.STATUS === "WATERING" ? "WATERING..." : "IDLE"}
        </TEXT>
       </VIEW>

       <TOUCHABLEOPACITY
        ONPRESS={() => STARTWATERCYCLE(P.ID)}
        STYLE={[STYLES.HOLDBTN, { BACKGROUNDCOLOR: ACCENT }]}
       >
        <IONICONS NAME="WATER" SIZE={16} COLOR="#FFF" />
        <TEXT STYLE={STYLES.HOLDBTNTEXT}> WATER ONCE</TEXT>
       </TOUCHABLEOPACITY>

       <VIEW STYLE={STYLES.ADJUSTROW}>
        <TEXT STYLE={STYLES.ADJUSTLABEL}>DURATION</TEXT>
        <VIEW STYLE={STYLES.ADJUSTCONTROLS}>
         <TOUCHABLEOPACITY
 STYLE={STYLES.ADJUSTBTN}
 ONPRESS={() => ADJUSTDURATION(P.ID, -10)}
>
 <TEXT STYLE={STYLES.ADJUSTBTNTEXT}>-</TEXT>
</TOUCHABLEOPACITY>
<TEXT STYLE={STYLES.ADJUSTVALUE}>{P.DURATION} S</TEXT>
<TOUCHABLEOPACITY
 STYLE={STYLES.ADJUSTBTN}
 ONPRESS={() => ADJUSTDURATION(P.ID, 10)}
>
 <TEXT STYLE={STYLES.ADJUSTBTNTEXT}>+</TEXT>
</TOUCHABLEOPACITY>

        </VIEW>
       </VIEW>

       <VIEW STYLE={STYLES.ADJUSTROW}>
        <TEXT STYLE={STYLES.ADJUSTLABEL}>INTERVAL</TEXT>
        <VIEW STYLE={STYLES.ADJUSTCONTROLS}>
         <TOUCHABLEOPACITY
          STYLE={STYLES.ADJUSTBTN}
          ONPRESS={() => ADJUSTINTERVAL(P.ID, -10)}
         >
          <TEXT STYLE={STYLES.ADJUSTBTNTEXT}>-</TEXT>
         </TOUCHABLEOPACITY>
         <TEXT STYLE={STYLES.ADJUSTVALUE}>{P.INTERVAL} S</TEXT>
         <TOUCHABLEOPACITY
          STYLE={STYLES.ADJUSTBTN}
          ONPRESS={() => ADJUSTINTERVAL(P.ID, 10)}
         >
```

```
          <Text style={styles.adjustBtnText}>+</Text>
         </TouchableOpacity>
        </View>
       </View>
      </View>
     ))}
    </View>

    <Text style={styles.sectionTitle}>Water All</Text>
    <View style={styles.cardRow}>
     <TouchableOpacity
      onPressIn={() => waterAll(true)}
      onPressOut={() => waterAll(false)}
      style={styles.holdAllBtn}
     >
      <Ionicons name="rainy-outline" size={18} color="#fff" />
      <Text style={styles.holdAllText}> Hold to Water All</Text>
     </TouchableOpacity>
    </View>

    <Text style={styles.sectionTitle}>Pump Controls</Text>
    <View style={styles.row}>
     <TouchableOpacity
      style={[styles.pumpBtn, { backgroundColor: pump1 ? "#43A047" :
"#E57373" }]}
      onPress={togglePump1}
     >
      <Text style={styles.pumpText}>{pump1 ? "Pump 1 ON" : "Pump 1
OFF"}</Text>
     </TouchableOpacity>
     <TouchableOpacity
      style={[styles.pumpBtn, { backgroundColor: pump2 ? "#43A047" :
"#E57373" }]}
      onPress={togglePump2}
     >
      <Text style={styles.pumpText}>{pump2 ? "Pump 2 ON" : "Pump 2
OFF"}</Text>
     </TouchableOpacity>
    </View>

    <Text style={styles.sectionTitle}>Auto Watering</Text>
    <View style={styles.cardRow}>
     <TouchableOpacity
      style={[
       styles.autoBtn,
       { backgroundColor: autoWatering ? "#FF7043" : ACCENT },
      ]}
      onPress={toggleAutoWatering}
     >
      <Ionicons name="refresh-outline" size={18} color="#fff" />
      <Text style={styles.autoText}>
       {autoWatering ? " Disable Auto Watering" : " Enable Auto Watering"}
      </Text>
     </TouchableOpacity>
    </View>

    <View style={{ height: 80 }} />
   </ScrollView>
  </View>
 );
}

const styles = StyleSheet.create({
 screen: { flex: 1, backgroundColor: "#F0F8F1" },
 scrollContent: { paddingBottom: 40 },
 backButton: {
  position: "absolute",
  top: 44,
  left: 16,
  zIndex: 40,
  backgroundColor: "rgba(124,179,66,0.2)",
  padding: 8,
  borderRadius: 50,
  elevation: 4,
 },
 image: { width: "100%", height: 400, resizeMode: "contain", marginTop: 80 },
 infoSection: { paddingHorizontal: 20, marginBottom: 10 },
 title: { fontSize: 22, fontWeight: "700", color: ACCENT_DARK,
marginBottom: 6 },
 subtitle: { fontSize: 16, fontWeight: "600", color: "#558B2F", marginBottom: 8
},
 desc: { fontSize: 13, color: "#4F4F4F", marginBottom: 6 },
 curveContainer: { position: "relative", width: "100%", height:
CURVE_HEIGHT, justifyContent: "flex-end", marginBottom: 14 },
 curveShape: {
  position: "absolute",
  bottom: -50,
  width: "100%",
  height: CURVE_HEIGHT + 60,
  backgroundColor: ACCENT,
  borderTopLeftRadius: 200,
  transform: [{ scaleX: 1.2 }],
  },
  statsRow: { flexDirection: "row", alignItems: "center", justifyContent: "space-evenly", height: "100%" },
  statItem: { alignItems: "center" },
  statIconCircle: { width: 56, height: 56, borderRadius: 28, backgroundColor: "rgba(255,255,255,0.18)", justifyContent: "center", alignItems: "center", marginBottom: 6 },
  statLabel: { fontSize: 12, color: "#fff", fontWeight: "600" },
  statValue: { fontSize: 13, color: "#fff", fontWeight: "700", marginTop: 4 },
  plantLineImage: { width: width * 1.5, height: 180, resizeMode: "contain", position: "absolute", top: CURVE_HEIGHT - 70, left: -width * 0.25, zIndex: 10 },
  sectionTitle: { fontSize: 18, fontWeight: "700", marginBottom: 8, color: ACCENT_DARK, alignSelf: "center", marginTop: 90 },
  potsGrid: { flexDirection: "row", flexWrap: "wrap", justifyContent: "center", paddingHorizontal: 4 },
  potCard: { width: "95%", backgroundColor: "#fff", marginVertical: 6, borderRadius: CARD_RADIUS, padding: 14, shadowColor: "#000", shadowOffset: { width: 0, height: 2 }, shadowOpacity: 0.06, shadowRadius: 6, elevation: 2, borderWidth: 1, borderColor: "rgba(124,179,66,0.12)" },
  potHeader: { flexDirection: "row", justifyContent: "space-between", alignItems: "center", marginBottom: 8 },
  potTitle: { fontWeight: "700", color: ACCENT_DARK },
  potStatus: { fontSize: 12 },
  holdBtn: { marginTop: 6, flexDirection: "row", alignItems: "center", justifyContent: "center", paddingVertical: 10, borderRadius: CARD_RADIUS - 2 },
  holdBtnText: { color: "#fff", fontWeight: "700" },
  adjustRow: { flexDirection: "row", justifyContent: "space-between", alignItems: "center", marginTop: 8 },
  adjustLabel: { color: "#555", fontSize: 12, width: 70 },
  adjustControls: { flexDirection: "row", alignItems: "center" },
  adjustBtn: { backgroundColor: ACCENT, paddingHorizontal: 8, paddingVertical: 6, borderRadius: 6, marginHorizontal: 6 },
  adjustBtnText: { color: "#fff", fontWeight: "700" },
  adjustValue: { minWidth: 70, textAlign: "center", color: ACCENT_DARK, fontWeight: "700" },
  cardRow: { alignItems: "center", marginBottom: 12 },
  holdAllBtn: { flexDirection: "row", alignItems: "center", backgroundColor: ACCENT, paddingVertical: 14, paddingHorizontal: 18, borderRadius: 12 },
  holdAllText: { color: "#fff", fontWeight: "700", marginLeft: 8 },
  row: { flexDirection: "row", justifyContent: "space-evenly", marginVertical: 8, paddingHorizontal: 20 },
  pumpBtn: { flex: 1, marginHorizontal: 6, paddingVertical: 12, borderRadius: 12, alignItems: "center" },
  pumpText: { color: "#fff", fontWeight: "700" },
 autoBtn: { flexDirection: "row", alignItems: "center", paddingVertical: 12,
paddingHorizontal: 18, borderRadius: 12 },
 autoText: { color: "#fff", fontWeight: "700", marginLeft: 8 },
});
```

Settings

```
import React, { useEffect, useState } from "react";
import {
 View,
 Text,
 StyleSheet,
 TouchableOpacity,
 ImageBackground,
} from "react-native";
import Icon from "react-native-vector-icons/MaterialIcons";
import { getAuth, onAuthStateChanged, signOut } from "firebase/auth";
import { app } from "../../config/firebaseConfig";
import { useRouter } from "expo-router";

const ProfileScreen = () => {
 const [userName, setUserName] = useState<string | null>(null);
 const [userEmail, setUserEmail] = useState<string | null>(null);
 const router = useRouter();

 useEffect(() => {
  const auth = getAuth(app);
  const unsubscribe = onAuthStateChanged(auth, (user) => {
   if (user) {
    setUserEmail(user.email);
    setUserName(user.displayName || "Anonymous User");
   } else {
    setUserName(null);
    setUserEmail(null);
   }
  });
  return () => unsubscribe();
 }, []);

 const handleLogout = async () => {
  const auth = getAuth(app);
  try {
   await signOut(auth);
   console.log("✅ User logged out");
   router.replace("/");
  } catch (error) {
   console.error("❌ Logout error:", error);
  }
 };

 RETURN (
  <VIEW STYLE={STYLES.CONTAINER}>
   <TEXT STYLE={STYLES.HEADER}>PROFILE</TEXT>

   <IMAGEBACKGROUND
    SOURCE={{
     URI: "HTTPS://IMG.FREEPIK.COM/FREE-VECTOR/ABSTRACT-PAPER-CUT-SHAPE-WAVE-
BACKGROUND_474888-4434.JPG?W=740",
    }}
    STYLE={STYLES.PROFILECARD}
    IMAGESTYLE={{ BORDERRADIUS: 12 }}
   >
    <VIEW STYLE={STYLES.PROFILEICONWRAPPER}>
     <ICON NAME="PERSON" SIZE={40} COLOR="#A0C97F" />
    </VIEW>
    <VIEW STYLE={STYLES.USERINFO}>
     <TEXT STYLE={STYLES.USERNAME}>
      {USERNAME ? USERNAME : "LOADING..."}
     </TEXT>
     <TEXT STYLE={STYLES.USEREMAIL}>
      {USEREMAIL ? USEREMAIL : "NO EMAIL"}
     </TEXT>
    </VIEW>
   </IMAGEBACKGROUND>

   {/* ACCOUNT DETAILS */}
   <TOUCHABLEOPACITY
    STYLE={STYLES.OPTIONROW}
    ONPRESS={() => ROUTER.PUSH("/CHANGEPASSWORD")}
   >
    <VIEW STYLE={STYLES.ICONTEXT}>
     <ICON NAME="PERSON-OUTLINE" SIZE={24} COLOR="#2E4A2E" />
     <TEXT STYLE={STYLES.OPTIONTEXT}>ACCOUNT DETAILS</TEXT>
    </VIEW>
    <ICON NAME="KEYBOARD-ARROW-RIGHT" SIZE={24} COLOR="#888" />
   </TOUCHABLEOPACITY>

   {/* TERMS AND CONDITIONS */}
   <TOUCHABLEOPACITY
    STYLE={STYLES.OPTIONROW}
    ONPRESS={() => ROUTER.PUSH("/TERMSANDCONDITIONS")}
   >
    <VIEW STYLE={STYLES.ICONTEXT}>
```

```
        <ICON NAME="GAVEL" SIZE={24} COLOR="#2E4A2E" />
        <TEXT STYLE={STYLES.OPTIONTEXT}>TERMS AND CONDITIONS</TEXT>
      </VIEW>
      <ICON NAME="KEYBOARD-ARROW-RIGHT" SIZE={24} COLOR="#888" />
    </TOUCHABLEOPACITY>

    {/* LOGOUT BUTTON */}
    <TOUCHABLEOPACITY STYLE={STYLES.LOGOUTBUTTON}
ONPRESS={HANDLELOGOUT}>
      <TEXT STYLE={STYLES.LOGOUTTEXT}>LOGOUT</TEXT>
    </TOUCHABLEOPACITY>
  </VIEW>
 );
};

CONST STYLES = STYLESHEET.CREATE({
 CONTAINER: { FLEX: 1, BACKGROUNDCOLOR: "#FFFFFF", PADDINGHORIZONTAL: 20,
PADDINGTOP: 50 },
 HEADER: { FONTSIZE: 22, FONTWEIGHT: "600", COLOR: "#2E4A2E", MARGINBOTTOM: 20
},
 PROFILECARD: {
  FLEXDIRECTION: "ROW",
  ALIGNITEMS: "CENTER",
  BORDERRADIUS: 12,
  OVERFLOW: "HIDDEN",
  PADDINGHORIZONTAL: 16,
  PADDINGVERTICAL: 24,
  MARGINBOTTOM: 30,
 },
 PROFILEICONWRAPPER: {
  WIDTH: 60,
  HEIGHT: 60,
  BORDERRADIUS: 30,
  BACKGROUNDCOLOR: "#FFF",
  JUSTIFYCONTENT: "CENTER",
  ALIGNITEMS: "CENTER",
  MARGINRIGHT: 15,
 },
 USERINFO: { FLEX: 1 },
 USERNAME: { FONTSIZE: 18, FONTWEIGHT: "600", COLOR: "#FFFFFF" },
 USEREMAIL: { FONTSIZE: 14, COLOR: "#F0F0F0", MARGINTOP: 4 },
 OPTIONROW: {
  FLEXDIRECTION: "ROW",
  JUSTIFYCONTENT: "SPACE-BETWEEN",
  ALIGNITEMS: "CENTER",
  PADDINGVERTICAL: 16,
    borderBottomColor: "#e0e0e0",
    borderBottomWidth: 1,
  },
  iconText: { flexDirection: "row", alignItems: "center", gap: 12 },
  optionText: { fontSize: 16, color: "#2e4a2e" },
  logoutButton: {
    marginTop: 40,
    backgroundColor: "#a0c97f",
    paddingVertical: 14,
    borderRadius: 30,
    alignItems: "center",
    elevation: 2,
    shadowColor: "#000",
    shadowOffset: { width: 0, height: 10 },
    shadowOpacity: 0.1,
    shadowRadius: 6,
  },
  logoutText: { color: "#ffffff", fontWeight: "bold", fontSize: 16 },
});

export default ProfileScreen;

aboutscreenstyles
import { StyleSheet } from "react-native";

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#f5f5f5",
  },
  topSection: {
    position: "relative",
    height: 300,
  },
  topImage: {
    width: "100%",
    height: "100%",
    resizeMode: "cover",
  },
  curve: {
    position: "absolute",
    bottom: 0,
    width: "100%",
    height: 100,
    backgroundColor: "#f5f5f5",
    borderTopLeftRadius: 60,
    borderTopRightRadius: 60,
  },
  aboutText: {
    position: "absolute",
    bottom: 30,
    left: 20,
    fontSize: 32,
    fontWeight: "bold",
    color: "#4caf50",
  },
  content: {
    padding: 20,
  },
  title: {
    fontSize: 24,
    fontWeight: "bold",
    color: "#388E3C",
    marginBottom: 10,
  },
  paragraph: {
    fontSize: 16,
    color: "#555",
    lineHeight: 24,
    marginBottom: 20,
  },
  goBackButton: {
    flexDirection: "row",
    alignItems: "center",
    justifyContent: "center",
    backgroundColor: "#4caf50",
    paddingVertical: 12,
    borderRadius: 30,
    marginHorizontal: 20,
    marginBottom: 20,
  },
  goBackText: {
    color: "#fff",
    fontSize: 18,
    fontWeight: "bold",
    marginLeft: 10,
  },
});

export default styles;
```

indexstyles

```
import { StyleSheet } from "react-native";

const styles = StyleSheet.create({
 background: {
  flex: 1,
  resizeMode: "cover",
  justifyContent: "center",
 },
 overlay: {
  flex: 1,
  justifyContent: "center",
  alignItems: "center",
  backgroundColor: "rgba(0, 0, 0, 0.5)",
 },
 mainTextContainer: {
  position: "absolute",
  top: 70,
  left: 20,
 },
 mainTextLine: {
  fontSize: 36,
  fontWeight: "bold",
  color: "#fff",
  lineHeight: 40,
  marginBottom: 10,
 },
 signInButton: {
  backgroundColor: "rgba(255, 255, 255, 0.3)",
  padding: 15,
  borderRadius: 30,
  width: "80%",
  alignItems: "center",
  marginBottom: 20,
 },
 signInButtonText: {
  color: "#fff",
  fontSize: 18,
  fontWeight: "bold",
 },
 createAccountText: {
  color: "#fff",
  fontSize: 16,
 },
});

export default styles;
```

```
loginstyles
import { StyleSheet } from "react-native";

const styles = StyleSheet.create({
 container: {
  flex: 1,
  backgroundColor: "#fff",
 },
 scrollContainer: {
  flexGrow: 1,
 },
 background: {
  height: 210,
  justifyContent: "flex-end",
 },
 wave: {
  position: "relative",
  top: 10,
 },
 backButton: {
  position: "absolute",
  top: 40,
  left: 20,
  zIndex: 10,
  backgroundColor: "rgba(255, 255, 255, 0.7)",
  opacity: 0.8,
  borderRadius: 50,
  padding: 8,
 },
 content: {
  padding: 20,
  alignItems: "center",
  backgroundColor: "#fff",
 },
 welcomeText: {
  fontSize: 32,
  fontWeight: "bold",
  color: "#1b5e20",
  marginBottom: 5,
 },
 loginPromptText: {
  fontSize: 16,
  color: "#555",
  marginBottom: 20,
 },
errorText: {
  color: "#ff3b30",
  fontSize: 14,
  marginBottom: 10,
  fontWeight: "bold",
},
inputContainer: {
  flexDirection: "row",
  alignItems: "center",
  backgroundColor: "#f1f1f1",
  borderRadius: 10,
  padding: 12,
  width: "100%",
  marginBottom: 15,
},
inputIcon: {
  marginRight: 10,
},
input: {
  flex: 1,
  fontSize: 16,
  color: "#333",
},
optionsContainer: {
  flexDirection: "row",
  alignItems: "center",
  justifyContent: "space-between",
  width: "100%",
  marginBottom: 20,
},
rememberMeCircle: {
  width: 16,
  height: 16,
  borderRadius: 8,
  borderWidth: 2,
  borderColor: "#b0b0b0",
  justifyContent: "center",
  alignItems: "center",
  marginRight: 5,
},
rememberMeCircleActive: {
  borderColor: "#1b5e20",
  backgroundColor: "#1b5e20",
},
checkIcon: {
  alignSelf: "center",
 },
 rememberMeText: {
  fontSize: 12,
  color: "#555",
  marginLeft: -110,
 },
 forgotPasswordText: {
  fontSize: 12,
  color: "#6c757d",
  textDecorationLine: "underline",
 },
 loginButton: {
  backgroundColor: "#1b5e20",
  padding: 15,
  borderRadius: 30,
  alignItems: "center",
  width: "80%",
  alignSelf: "center",
  marginBottom: 80,
 },
 buttonText: {
  color: "#fff",
  fontSize: 18,
  fontWeight: "bold",
 },
});

export default styles;

mainappStyles
import { StyleSheet, Dimensions } from "react-native";

const { width } = Dimensions.get("window");
const ITEM_WIDTH = width * 0.7;
const SPACING = 10;

const styles = StyleSheet.create({
 container: {
  flex: 1,
  backgroundColor: "#f5f5f5",
 },
  screenWrapper: {
  flex: 1,


 },
tabContentWrapper: {
 flex: 1,
 paddingBottom: 100,

 },
 navItem: {
 alignItems: "center",
 justifyContent: "center",
},
activeNavItem: {

 transform: [{ scale: 1.05 }],
},
 header: {
  paddingBottom: 12,
  paddingHorizontal: 15,
  backgroundColor: "#fff",
  borderBottomWidth: 1,
  borderBottomColor: "#ddd",
 },
 topRow: {
  display: "flex",
  flexDirection: "row",
  justifyContent: "space-between",
  alignItems: "center",

 },
 searchRow: {
  flexDirection: "row",
  alignItems: "center",
  marginTop: 10,
  marginBottom: 10,
 },
 profileRow: {
  paddingBottom: 5,
  flexDirection: "row",
  alignItems: "center",
  justifyContent: "center",
  gap: 10,
 },
 profilePic: {
  bottom: -10,
  width: 40,
  height: 40,
  borderRadius: 100,
 },
welcomeText: {

 bottom: -10,
 fontSize: 16,
 fontWeight: "bold",
 color: "#333",
},
iconContainer: {
 bottom: -10,
 backgroundColor: "#f0f0f0",
 borderRadius: 20,
 padding: 8,
},
searchBarContainer: {
 flexDirection: "row",
 alignItems: "center",
 backgroundColor: "#eee",
 borderRadius: 20,
 paddingLeft: 10,
 marginTop: 20,
 marginBottom: -5,
 width: "80%",
},
searchIcon: {
 marginRight: 5,
},
searchBar: {
 flex: 1,
 paddingVertical: 10,
},
filterIcon: {
 position: "absolute",
 top: 78,
 right: 15,
},
filterBackground: {
 backgroundColor: "#f0f0f0",
 borderRadius: 20,
 padding: 8,
},
content: {
 paddingBottom: 20,
 paddingHorizontal: 20,
},
sampleText: {
 fontSize: 18,
 marginBottom: 20,
},
listItem: {
 fontSize: 16,
 paddingVertical: 10,
 borderBottomWidth: 1,
 borderBottomColor: "#ccc",
},
horizontalScroll: {
 marginBottom: 20,
},
giftSection: {
 flexDirection: "row",
 alignItems: "center",
 backgroundColor: "#8bc34a",
 margin: 15,
 padding: 15,
 borderRadius: 10,
 overflow: "visible",
 zIndex: 1,
},
giftImage: {
 width: 180,
 height: 150,
 position: "absolute",
 top: -30,
 left: -15,
 zIndex: 2,
},
giftTextContainer: {
 flex: 1,
 marginLeft: 130,
},
giftTitle: {
 fontSize: 18,
 fontWeight: "bold",
 color: "#fff",
},
giftButton: {
 marginTop: 10,
 backgroundColor: "#689F38",
 opacity: 1,
 paddingVertical: 10,
 paddingHorizontal: 20,
 borderRadius: 30,
},
giftButtonText: {
 color: "#fff",
 fontWeight: "bold",
 opacity: 1,
},
sensContainer: {
 flex: 1,
 backgroundColor: "#f5f5f5",
 paddingTop: 50,
},
sensTitle: {
 fontSize: 22,
 fontWeight: "bold",
 marginBottom: 20,
 textAlign: "center",
},
sensorCard: {
 width: ITEM_WIDTH,
 height: 200,
 borderRadius: 16,
 alignItems: "center",
 justifyContent: "center",
 marginHorizontal: SPACING / 2,
 shadowColor: "#000",
 shadowOffset: { width: 0, height: 4 },
 shadowOpacity: 0.3,
 shadowRadius: 6,
 elevation: 8,
},
sensorTitle: {
 fontSize: 22,
 color: "#FFF",
 fontWeight: "bold",
},
sensorValue: {
 fontSize: 18,
 color: "#FFF",
},
sectionTitle: {
 fontSize: 18,
 fontWeight: "bold",
 color: "#000000",
 marginVertical: 10,
},
learnMoreContainer: {
 marginBottom: 100,
},
largeWidget: {
 width: 280,
 height: 180,
 marginRight: 20,
 backgroundColor: "#ffffff",
 borderRadius: 15,
 shadowColor: "#000",
 shadowOpacity: 0.1,
 shadowRadius: 8,
 elevation: 5,
 padding: 20,
 justifyContent: "center",
 alignItems: "center",
},
largeWidgetTitle: {
 fontSize: 22,
 fontWeight: "bold",
 color: "#2E7D32",
 marginVertical: 10,
},
largeWidgetText: {
 fontSize: 14,
 color: "#555",
 textAlign: "center",
 marginTop: 5,
},
plantCard: {
 width: 160,
 height: 220,
 borderRadius: 12,
 backgroundColor: "#fff",
 shadowColor: "#000",
 shadowOffset: { width: 0, height: 4 },
 shadowOpacity: 0.1,
 shadowRadius: 6,
 elevation: 5,
 marginRight: 15,
 overflow: "hidden",
 borderWidth: 1,
 borderColor: "#eee",
},

imageContainer: {
 width: "100%",
 height: 130,
  backgroundColor: "#fff",
  justifyContent: "center",
  alignItems: "center",
  borderBottomWidth: 1,
  borderColor: "#eee",
},

plantImage: {
  width: "90%",
  height: "90%",
  resizeMode: "cover",
  borderRadius: 8,
},

plantInfo: {
  flexDirection: "row",
  justifyContent: "space-between",
  alignItems: "center",
  paddingHorizontal: 12,
  paddingVertical: 10,
},

plantName: {
  fontSize: 16,
  fontWeight: "bold",
  color: "#333",
},

arrowButton: {
  width: 32,
  height: 32,
  borderRadius: 16,
  borderWidth: 2,
  borderColor: "#9ACD32",
  justifyContent: "center",
  alignItems: "center",
},

navBarContainer: {
  position: "absolute",
  bottom: 20,
  left: 20,
  right: 20,
  backgroundColor: "transparent",
  elevation: 10,
},
  navBar: {
    flexDirection: "row",
    justifyContent: "space-around",
    paddingVertical: 10,
    backgroundColor: "#fff",
    borderRadius: 30,
    shadowColor: "#000",
    shadowOpacity: 0.1,
    shadowRadius: 10,
    elevation: 5,
  },
  navCircle: {
    backgroundColor: "#f0f0f0",
    borderRadius: 30,
    padding: 10,
  },
  activeNavCircle: {
    backgroundColor: "#8bc34a",
    paddingHorizontal: 20,
  },
  statusDot: {
  position: "absolute",
  top: 8,
  right: 8,
  width: 12,
  height: 12,
  borderRadius: 6,
  backgroundColor: "red",
  borderWidth: 1.5,
  borderColor: "#fff",
  zIndex: 10,
},
notificationIconContainer: {
  position: "absolute",
  right: 20,
  top: 70,
},

notificationCircle: {
  backgroundColor: "#eee",
  padding: 10,
  borderRadius: 30,
  justifyContent: "center",
  alignItems: "center",
  elevation: 3,
},
notificationDot: {
 position: "absolute",
 top: 6,
 right: 6,
 width: 10,
 height: 10,
 borderRadius: 5,
 backgroundColor: "#FF5252",
},

});

export default styles;

registrationStyles
import { StyleSheet } from "react-native";

const styles = StyleSheet.create({
 scrollContainer: {
  flexGrow: 1,
  backgroundColor: "#fff",
 },
 backgroundContainer: {
  height: 290,
  overflow: "hidden",
 },
 background: {
  flex: 1,
  justifyContent: "flex-end",
 },
 wave: {
  position: "relative",
  top: -80,
 },
 backButton: {
  position: "absolute",
  top: 40,
  left: 20,
  zIndex: 3,
  backgroundColor: "rgba(255, 255, 255, 0.3)",
  padding: 10,
  borderRadius: 20,
 },
 content: {
  padding: 20,
},
header: {
 fontSize: 36,
 fontWeight: "bold",
 color: "#1b5e20",
 textAlign: "center",
 marginTop: -80,
},
loginPromptText: {
 fontSize: 16,
 color: "#666",
 textAlign: "center",
 marginBottom: 30,
},
inputContainer: {
 flexDirection: "row",
 alignItems: "center",
 backgroundColor: "#f9f9f9",
 borderRadius: 12,
 paddingHorizontal: 15,
 marginVertical: 10,
 borderWidth: 1,
 borderColor: "#ddd",
 shadowColor: "#000",
 shadowOpacity: 0.1,
 shadowOffset: { width: 0, height: 2 },
 shadowRadius: 5,
 elevation: 3,
},
inputIcon: {
 marginRight: 10,
},
input: {
 flex: 1,
 height: 50,
 fontSize: 16,
 color: "#333",
},
registerButton: {
 backgroundColor: "#1b5e20",
 paddingVertical: 15,
 borderRadius: 30,
 alignItems: "center",
 justifyContent: "center",
 marginTop: 20,
 shadowColor: "#000",
  shadowOpacity: 0.2,
  shadowOffset: { width: 0, height: 2 },
  shadowRadius: 4,
  elevation: 4,
 },
 buttonText: {
  color: "#fff",
  fontWeight: "600",
  fontSize: 18,
 },
});

export default styles;

tipscreenStyles
import { StyleSheet } from "react-native";

const styles = StyleSheet.create({
 container: {
  flex: 1,
  backgroundColor: "#f5f5f5",
 },
 topSection: {
  position: "relative",
  height: 300,
 },
 topImage: {
  width: "100%",
  height: "100%",
  resizeMode: "cover",
 },
 curve: {
  position: "absolute",
  bottom: -1,
  width: "100%",
  height: 120,
  backgroundColor: "#f5f5f5",
  borderTopLeftRadius: 60,
  borderTopRightRadius: 60,
  zIndex: 1,
 },
 tipsText: {
  position: "absolute",
  bottom: 40,
  left: 20,
  fontSize: 32,
    fontWeight: "bold",
    color: "#4caf50",
    zIndex: 2,
  },
  content: {
    padding: 20,
    marginTop: -60,
    backgroundColor: "#f5f5f5",
    zIndex: 0,
  },
  title: {
    fontSize: 28,
    fontWeight: "bold",
    color: "#2E7D32",
    marginBottom: 15,
  },
  tipTitle: {
    fontSize: 20,
    fontWeight: "bold",
    color: "#388E3C",
    marginTop: 20,
  },
  description: {
    fontSize: 16,
    color: "#555",
    lineHeight: 24,
    marginBottom: 15,
  },
  goBackButton: {
    flexDirection: "row",
    alignItems: "center",
    justifyContent: "center",
    backgroundColor: "#4caf50",
    paddingVertical: 12,
    borderRadius: 30,
    marginHorizontal: 20,
    marginBottom: 20,
  },
  goBackText: {
    color: "#fff",
    fontSize: 18,
    fontWeight: "bold",
    marginLeft: 10,
  },
});
```

```
EXPORT DEFAULT STYLES;
```

APPENDIX D. USER'S MANUAL

Appendix C. User's Manual

User Guide

This guide is designed to help you monitor, manage, and control your Smart Plant Tower through the mobile application. The guide will walk you through how to navigate and use the key features of the system.

---

1. Getting Started

1.1. Creating an Account

1. Open the app using Expo Go.
2. On the welcome screen, tap Register.
3. Fill out the required fields: email address, and password.
4. Tap Sign Up to create your account.

1.2. Logging In

1. After account creation, enter your registered email and password on the login page.
2. Tap Login to access your account.
3. You will be directed to the Main Screen.

---

2. The Main Screen

The Main Screen provides an overview of your plant system through interactive cards.

2.1. Humidity Card

- Displays the current humidity reading in real time.

2.2. Temperature Card

- Displays the current temperature reading in real time.

2.3. Water Level Card

- Displays the tank's water status: Full, Half-Full, or Empty.

2.4. pH Card

- Displays pH values (Alkaline, Neutral, Acidic)

2.5. Lighting Adjustment

- Located above the cards.
- Allows you to manage the tower's lighting system.

3. Navigation Bar

The navigation bar at the bottom provides quick access to key features:

3.1. Main Screen

- Overview of sensor cards and lighting control.

3.2. EC Screen

- Displays EC (Electrical Conductivity) value.
- Dashboard shows Temperature, Voltage, EC, and TDS values.

3.3. Water Level Screen

- Provides a dedicated view of the tank's water level.

3.4. Lighting Screen

- Allows you to toggle and configure the lighting system.

3.5. Profile Screen

- Displays account information.
- Includes Logout option.

4. Controls

4.1. Solenoid Control for All Pots

- Enable / On → Activates the solenoids to water all pots simultaneously**.**
- Disable / Off → Deactivates the solenoids to stop watering all pots simultaneously**.**

5. Account Management

5.1. Updating Profile

- Accessible in the Profile Screen.
- Allows you to view and manage account details.
- Allows you to change password

5.2. Logging Out

1. Navigate to the Profile Screen.
2. Tap Logout to securely exit your account.

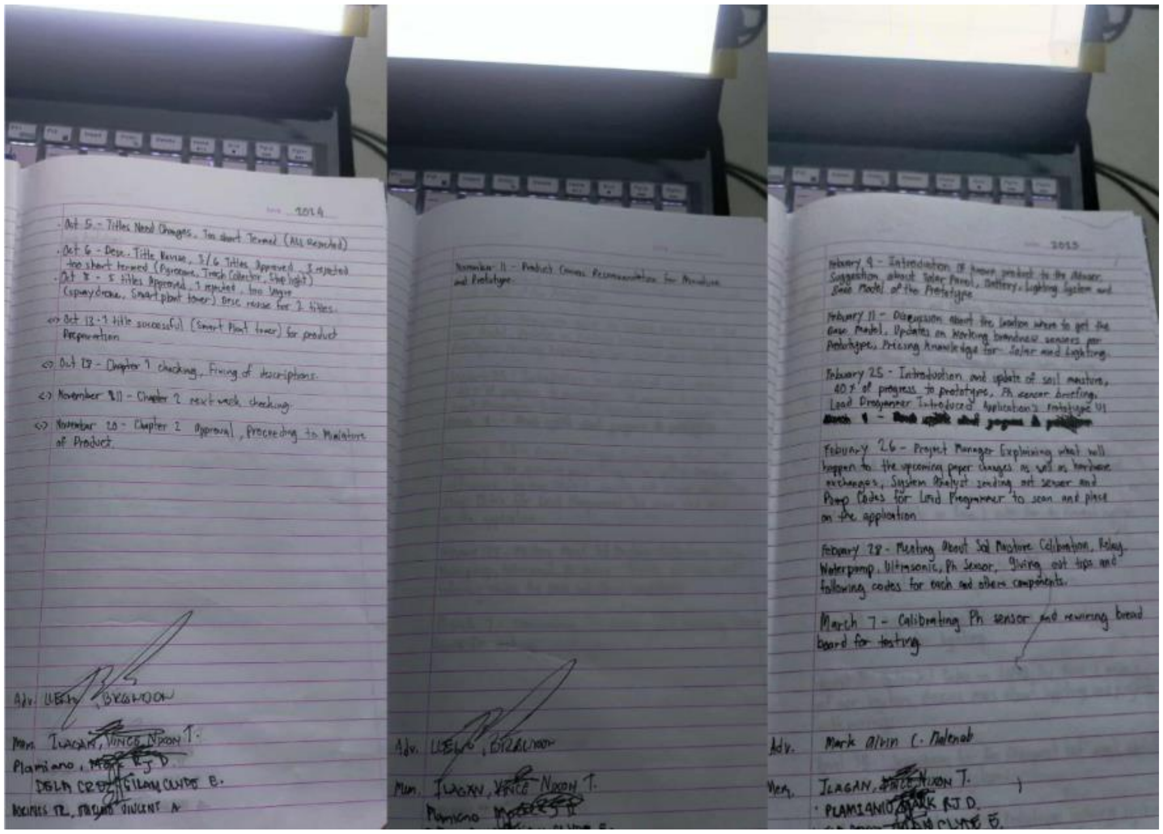
6. Troubleshooting

Issue	Cause	Solution
Cannot log in	Incorrect email or password	Verify credentials or reset password
No sensor data	ESP32 not connected to Wi-Fi	Restart ESP32 and check connection
Pots not responding	Firebase not syncing	Refresh the app or check database path
App not loading	Expo Go issue	Restart Expo Go and reload project

7. Notes

- The app is currently accessible through Expo Go only.
- A stable Wi-Fi connection is required for Firebase communication.
- ESP32 must remain powered and connected for accurate readings.

APPENDIX E. LOGBOOK



2024

- Oct 5 - Titles Need Changes. Too short Termed (All Rejected)
- Oct 6 - Desc. Title Revise, 3/6 Titles Approved, 3 rejected too short termed (Pyroscope, Trash Collector, Step light)
- Oct 8 - 5 titles Approved, 1 rejected, too vague (spray drone, Smart plant tower) Desc revise for 2 titles.

<> Oct 13 - 1 title successful (Smart Plant tower) for product Preparation

<> Oct 17 - Chapter 1 checking, Fixing of descriptions.

<> November 11 - Chapter 2 next week checking.

<> November 20 - Chapter 2 Approval, Proceeding to Miniature of Product.

Adv. LLENA, BRGWOON

Mem. ILAGAN, VINCE NIXON T.
Plamiano, MARK RJ D.
DELA CRUZ, GILLAN CLYDE B.
[illegible] JR, [illegible] VINCENT A.

November 11 - Product's Canvas Recommendation for Miniature and Prototype.

Adv. LLENA, BRGWOON

Mem. ILAGAN, VINCE NIXON T.
Plamiano MARK RJ D.

2025

Febuary 9 - Introduction of [illegible] product to the Adviser. Suggestion about Solar Panel, Battery, Lighting System and Base Model of the Prototype

Febuary 11 - Discussion about the location where to get the Base Model, Updates on Working brandnew sensors per Prototype, Pricing Knowledge for Solar and Lighting.

Febuary 25 - Introduction and update of soil moisture, 40% of progress to prototype, Ph sensor briefing, Lead Programmer Introduced Application's Prototype V1
~~March 1 - [illegible]~~

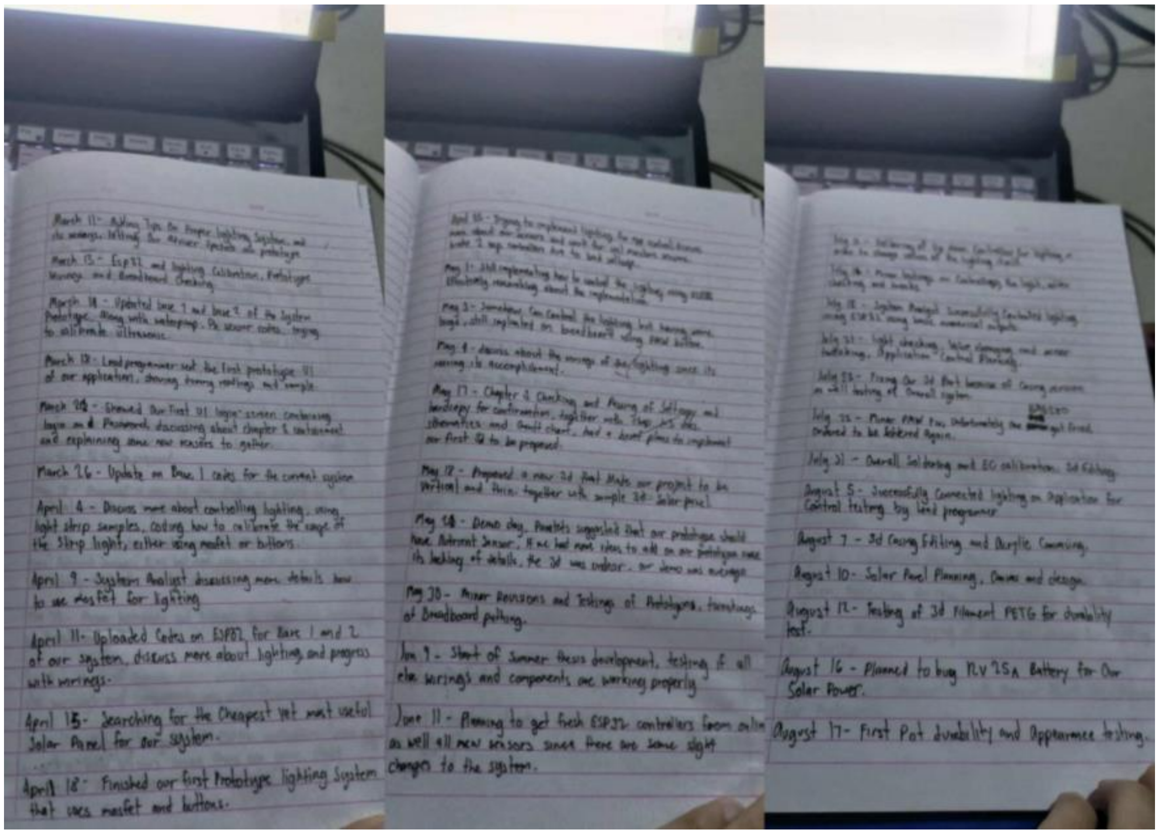
Febuary 26 - Project Manager Explaining what will happen to the upcoming paper changes as well as hardware exchanges, System Analyst sending out sensor and Pump Codes for Lead Programmer to scan and place on the application

Febuary 28 - Meeting About Soil Moisture Calibration, Relay, Waterpump, Ultrasonic, Ph Sensor, giving out tips and following codes for each and others components.

March 7 - Calibrating Ph sensor and rewiring bread board for testing

Adv. Mark Alvin C. Malenab

Mem. ILAGAN, VINCE NIXON T.
· PLAMIANO, MARK RJ D.
[illegible] GILLAN CLYDE B.



March 11 - Adding Tips for Proper lighting System and its wirings. Telling the Adviser Updates on prototype

March 15 - Esp32 and lighting Calibration, Prototype wirings and Breadboard Checking

March 18 - Updated base 1 and base 2 of the System Prototype along with waterpump, Ph sensor codes, trying to calibrate ultrasonic

March 12 - Lead programmer sent the first prototype V1 of our application, showing timing settings and sample

March 24 - Showed Our First UI login screen containing login and Password, discussing about chapter & [illegible] and explaining some new reasons to gather.

March 26 - Update on Base 1 codes for the current system

April 4 - Discuss more about controlling lighting, using light strip samples, coding how to calibrate the range of the Strip light, either using mosfet or buttons.

April 9 - System Analyst discussing more details how to use Mosfet for lighting

April 11 - Uploaded Codes on ESP32 for Base 1 and 2 of our system, discuss more about lighting and progress with wirings.

April 15 - Searching for the Cheapest yet most useful Solar Panel for our system.

April 18 - Finished our first Prototype lighting System that uses mosfet and buttons.

April 25 - Trying to implement lighting for the [illegible] [illegible] about our sensors and [illegible] for soil moisture [illegible] [illegible] 2 esp controllers due to [illegible] voltage.

May 1 - Still implementing how to control the lighting using [illegible] Effectively [illegible] about the implementation.

May 3 - Somehow can Control the lighting but having some bugs, still implemented on breadboard using [illegible] button.

May 4 - discuss about the wirings of [illegible] lighting since its nearing its accomplishment.

May 17 - Chapter 4 Checking and Fixing of Softcopy and hardcopy for confirmation, together with [illegible] [illegible] schematics and Gantt chart, had a [illegible] plans to implement our first [illegible] to be proposed.

May 17 - Proposed a new 3d Print Mats our project to be Vertical and thin together with simple 3d Solar panel

May 18 - Demo day, Panelists suggested that our prototype should have Nutrient Sensor, if we had more ideas to add on our prototype since its lacking of details, the 3d was unclear, our demo was average

May 30 - Minor Revisions and Testings of Prototypes, [illegible] of Breadboard putting.

June 9 - Start of Summer thesis development, testing if all the wirings and components are working properly

June 11 - Planning to get fresh ESP32 controllers from online as well all new sensors since there are some slight changes to the system.

July [illegible] - [illegible] of [illegible] Controller for lighting in order to change colors of the lighting itself.

July 16 - Minor testings on Calibrating the light, [illegible] checking and [illegible]

July 18 - System Analyst Successfully Controlled lighting using ESP32 using [illegible] numerical outputs.

July 21 - light checking, [illegible] and [illegible] [illegible], Application Control Planning

July 23 - Fixing Our 3d Part because of Casing [illegible] as well testing of Overall system.

July 25 - Minor [illegible] fix, Unfortunately the [illegible] got fried, ordered to be Ordered Again.

July 31 - Overall Soldering and EC calibration, 3d Editing

August 5 - Successfully Connected lighting on Application for Control testing by lead programmer

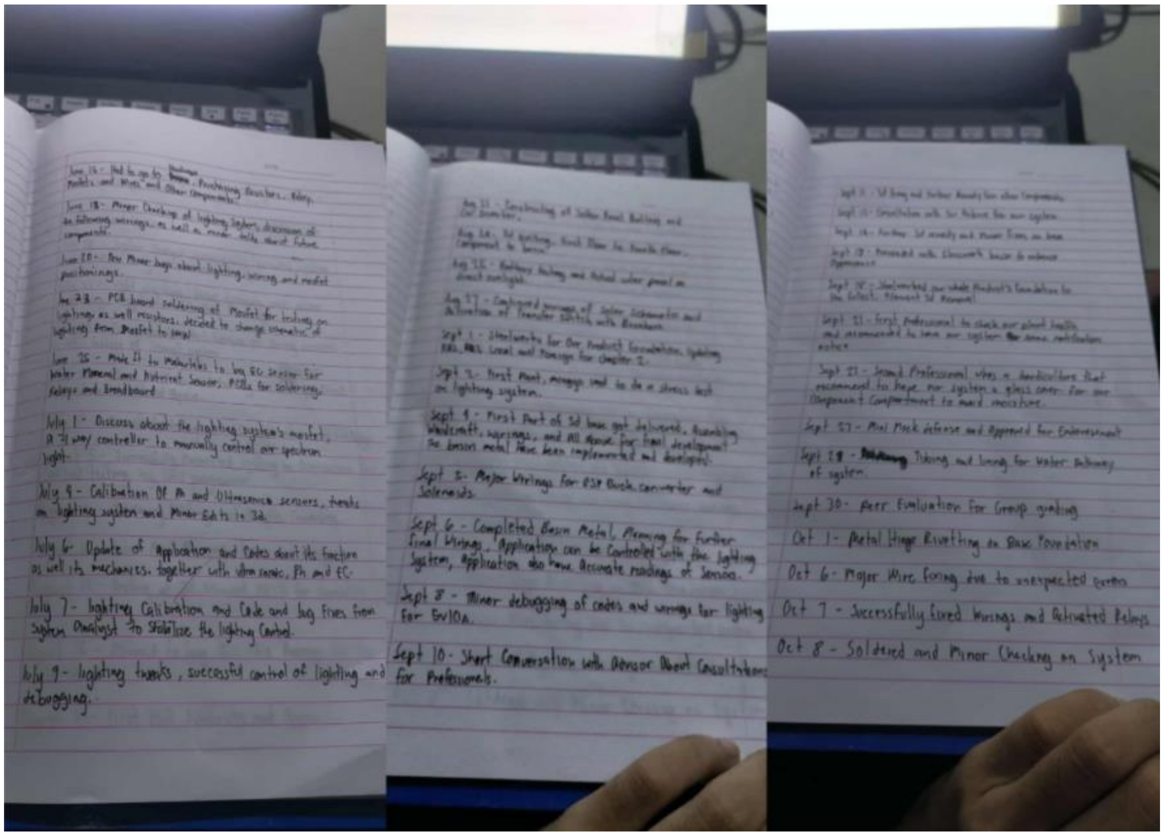
August 7 - 3d Casing Editing and Acrylic Cumming.

August 10 - Solar Panel Planning, [illegible] and design.

August 12 - Testing of 3d Filament PETG for durability test.

August 16 - Planned to buy 12V 25A Battery for Our Solar Power.

August 17 - First Pot durability and Appearance testing.



June 16 - Had to go to [illegible]. Purchasing Resistors, Relay, Mosfets and Wires and Other Components.

June 18 - Minor Checking of lighting System, discussion of the following wirings, as well as minor talks about future components.

June 20 - Few Minor bugs about lighting, wiring and mosfet positionings.

June 23 - PCB board soldering of Mosfet for testing on lightings as well resistors, decided to change schematic of lighting from Mosfet to [illegible]

June 25 - Made It to Malinta to buy EC sensor for Water Mineral and Nutrient Sensor, PCBs for soldering, Relays and Breadboard

July 1 - Discuss about the lighting system's mosfet, A 3 way controller to manually control our spectrum light.

July 4 - Calibration Of Ph and Ultrasonic sensors, tweaks on lighting system and Minor Edits in 3d.

July 6 - Update of Application and Codes about its function as well its mechanics, together with ultrasonic, Ph and EC.

July 7 - lighting Calibration and Code and bug fixes from System Analyst to Stabilize the lighting Control.

July 9 - lighting tweaks, successful control of lighting and debugging.

Aug 22 - Constructing of Solar Panel Battery and DC Inverter.

Aug 24 - 3d [illegible], [illegible] Floor [illegible] Component to [illegible]

Aug 26 - Battery Testing and Actual solar panel on direct sunlight.

Aug 27 - Continued [illegible] of Solar Schematic and Activation of Transfer Switch with Breadboard.

Sept 1 - Steelworks for Our Product Foundation, Updating [illegible] Local and Foreign for Chapter 2.

Sept 2 - First Plant, [illegible] to do a stress test on lighting system.

Sept 4 - First Part of 3d base got delivered, Assembling Woodcraft, wirings, and All Above for final development the basin metal Have been implemented and developed.

Sept 5 - Major Wirings for ESP Buck converter and Solenoids

Sept 6 - Completed Basin Metal, Planning for further final Wirings, Application can be Controlled with the lighting System, Application also have Accurate readings of Sensors.

Sept 8 - Minor debugging of codes and wirings for lighting for 5v10A.

Sept 10 - Short Conversation with Adviser About Consultations for Professionals.

Sept 11 - 3d [illegible] and further [illegible] for other Components

Sept 13 - Consultation with Sir [illegible] for our system

Sept 14 - Further 3d [illegible] and [illegible] on base

Sept 17 - [illegible] with [illegible] to [illegible]

Sept 19 - [illegible] our whole Product's Foundation to the latest Element 3d [illegible]

Sept 21 - First Professional to check our plant health and recommended to have our system [illegible] notification [illegible]

Sept 23 - Second Professional who's a horticulture that recommend to have our system a glass cover for our Component Compartment to avoid moisture.

Sept 27 - Mini Mock defense and Approved for Endorsement

Sept 28 - ~~[illegible]~~ Tubing and lining for Water Pathway of system.

Sept 30 - Peer Evaluation for Group grading

Oct 1 - Metal Hinge Rivetting on Base Foundation

Oct 6 - Major Wire fixing due to unexpected errors

Oct 7 - Successfully fixed Wirings and Activated Relays

Oct 8 - Soldered and Minor Checking on System

APPENDIX F. CURRICULUM VITAE OF RESEARCHERS

Joselito Vincent A. Borines Jr.
Jade St. 15 Union Village Bagumbong Caloocan City Brgy 171
Borines.291929@novaliches.sti.edu.ph
09947782247

EDUCATIONAL BACKGROUND

Level	Inclusive Dates	Name of school/ Institution
Tertiary	2024	STI College Novaliches
TechVoc	2022	Caloocan City Busines High School
High School	2019	St. Therese of Rose School
Elementary	2016	Union Village Christian Academy

PROFESSIONAL OR VOLUNTEER EXPERIENCE

Inclusive Dates	Nature of Experience/ Job Title	Name and Address of Company or Organization
2023	NSTP Service	National Service Training Program

AFFILIATIONS

Inclusive Dates	Name of Organization	Position
2023	ACES	Member
2024	ICPEP	Member

SKILLS

SKILLS	Level of Competency	Date Acquired
Influencer	Average	2023
Weight Lifting	Advanced	2021
Mathematics	Average	2017
Drawing	Average	2012
Application Development	Average	2024

TRAININGS, SEMINARS OR WORKSHOP ATTENDED

Inclusive Dates	Title of Training, Seminar or Workshop
May –June 2023	NSTP Seminar
April 2024 – Sept 2025	ICPEP Seminar
July 2025	SO1 Bosh

Vince Nixon T. Ilagan
B5 L20 Sampaguita St. Saranay Homes Caloocan City Brgy 173
Ilagan.288217@novaliches.sti.edu.ph
09761400974

EDUCATIONAL BACKGROUND

Level	Inclusive Dates	Name of school/ Institution
Tertiary	2023	STI College Novaliches
TechVoc	2022	St. Therese Of Rose School
High School	2019	St. Therese Of Rose School
Elementary	2016	Congress Elementary School

PROFESSIONAL OR VOLUNTEER EXPERIENCE

Inclusive Dates	Nature of Experience/ Job Title	Name and Address of Company or Organization
2025	Plate Encoder	Government Management
2023	NSTP Service	National Service Training Program
2022	Axie User	Local Management

AFFILIATIONS

Inclusive Dates	Name of Organization	Position
2023	ACES	Member
2024	ICPEP	Member

SKILLS

SKILLS	Level of Competency	Date Acquired
Research Specialist	Beginner	2023
Video Editing	Average	2020
3d Modelling	Beginner	2024

TRAININGS, SEMINARS OR WORKSHOP ATTENDED

Inclusive Dates	Title of Training, Seminar or Workshop
July 2025	SO1 Bosh
May - June 2023	NSTP Seminar
April 2024 – Sept 2025	ICPEP Seminar

Mark Rj D. Plamiano
Blk 179 Lot 01, Canase Street, St.Cecilia,
Deca Homes, Loma De Gato, Marilao, Bulacan
plamianomarkrj@gmail.com
(+63) 0905-2122-834

EDUCATIONAL BACKGROUND

Level	Inclusive Dates	Name of school/ Institution
Tertiary	August 2022	STI College Novaliches
TechVoc	August 2020 - June 2022	Ismael Mathay Sr. High School
High School	June 2016 - April 2020	Ismael Mathay Sr. High School
Elementary	June 2013 - March 2016	FSS Patulo Elementary School
Elementary	June 2010 - March 2013	Christ's Achievers Montessori

PROFESSIONAL OR VOLUNTEER EXPERIENCE

Inclusive Dates	Nature of Experience/ Job Title	Name and Address of Company or Organization
May2023-June2023	NSTP Service	National Service Training Program
June 2015	Manager	RJ-Angel's Coolers
February 2018- March 2018	Coordinator	Ismael Mathay Sr. High School

AFFILIATIONS

Inclusive Dates	Name of Organization	Position
2023	ACES	Member
2024	ICPEP	Member
April 2020	Samahan ng Kabataan	Member
June 2015	RJ-Angels Coolers	Member
June 2013	Boy Scout of the Philippines	Member

SKILLS

SKILLS	Level of Competency	Date Acquired
Programming Language (Java, Python)	Average	2019
Management	Advanced	2015
Mini Controller Interfacing	Average	2024
Embedded System Programming	Basic	2024

TRAININGS, SEMINARS OR WORKSHOP ATTENDED

Inclusive Dates	Title of Training, Seminar or Workshop
July 2025	SO1 Bosh
May - June 2023	NSTP Seminar
April 2024 – Sept 2025	ICPEP Seminar

Gilan Clyde E. Dela Cruz
NPC Area-C, Diamond St. Interville I, Subdivision, Brgy. 164,
Talipapa, Caloocan, City
delacruz.300273@novaliches.sti.edu.ph
(+63) 0999-4788-992/#366-7411

EDUCATIONAL BACKGROUND

Level	Inclusive Dates	Name of school/ Institution
Tertiary	August 2022	STI College Novaliches
TechVoc	June 2020-March 2022	Dr. Carlos S. Lanting College
High School	June 2016-March 2020	Talipapa High School
Elementary	June 2012-March 2016	Talipapa Elementary School
Elementary	June 2010-March 2011	Placido Del Mundo Elementary School

PROFESSIONAL OR VOLUNTEER EXPERIENCE

Inclusive Dates	Nature of Experience/ Job Title	Name and Address of Company or Organization
October 2019-March 2020	Philippine Red Cross Volunteer	Philippine Red Cross
June 2022	Parish Pastoral Council for Resposible Voting	Parokya ng Banal na Sakramento Youth Community
December 2019	Red Cross Volunteer-Typhoon Tisoy	Philippine Red Cross
May 2021	Ecology or Tree Farming	Boy Scout of the Philippines

AFFILIATIONS

Inclusive Dates	Name of Organization	Position
October 2019	Philippine Red Cross	Volunteer
August 2018-2024	Boy Scout of the Philippines	4thRanking/Venturer
May 2019	Parish Pastoral Council for Responsible Voting	Logistics Committee
March 2024	Social Action Ministry Community Caravan	Volunteer/Logistics Committee

SKILLS

SKILLS	Level of Competency	Date Acquired
Powerpoint / Word / Excel	Average	2020
3d Modelling	Average	2023
Hardware Design & Troubleshooting	Average	2024
Emerging Tech Adaptability & Continuous Learning	Average	2024

TRAININGS, SEMINARS OR WORKSHOP ATTENDED

Inclusive Dates	Title of Training, Seminar or Workshop
July 2019	Pioneering and First Aiding Committee
May - June 2023	NSTP Seminar
April 2024 – Sept 2025	ICPEP Seminar
March 2023	Completion of Basic Java Programming
July 2025	SO1 BOSH