**UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI**
**Group Project Final Report**


# Soil Condition Monitoring Application

*Members*
Nguyễn Nhật Anh - BI12-022
Nguyễn Trần Đức Quý - BI12-376
Trần Huy Nguyên - BI12-327
Nguyễn Đăng Linh - BI11-153
Bùi Minh Quang - BI12-373
Dương Hải Long - BI11-160
Nguyễn Quang Anh - BA10-002

*Supervisors*
Kiều Quốc Việt

*ICT Labs*
January, 2023

**TABLE OF CONTENTS**

**TABLE OF FIGURE**

# I, Introduction

## 1. Motivation

### 1.1. Problem Statement

Plants play a crucial role in the ecosystem, enriching and maintaining the balance of the surrounding environment. To ensure the robust growth and vitality of plants, several critical factors need attention and maintenance. Ensuring an adequate water supply, maintaining suitable temperature and humidity, as well as ensuring that plants receive sufficient sunlight are essential factors for keeping plants in the best condition. Neglecting these factors can lead to suboptimal plant health and significantly impact plant development. However, monitoring and controlling the plant's condition over an extended period have become a challenge for plant growers. Faced with this issue, utilizing the Internet of Things can become an effective solution.

### 1.2. IOT

IoT, or Internet of things is a system that connects devices and objects through the internet, enabling them to interact and share data without direct human intervention. This opens up possibilities for automation and optimizing information management, ranging from household devices to industrial and agricultural applications. IoT plays a crucial role in creating smart and efficient environments.

The Internet of Things (IoT) has brought significant innovations to the field of flower cultivation, delivering notable benefits and optimizing the care process for plants. Some of the advantages include smart irrigation, environmental monitoring, remote management systems, and many other benefits.

### 1.3. Overview

#### 1.3.1. Introduction to tree data measuring apps around the world

Worldwide, the use of IoT in agriculture is gradually becoming more common, and numerous applications have been developed to bring convenience to users in farming. However, among them, there are not many applications that directly measure soil moisture like the Flower Care app. This app is capable of directly measuring data but is not available to users in many places and only operates within the Xiaomi ecosystem. Most applications focus on listing various plant types and providing care instructions, such as the PictureThis app.

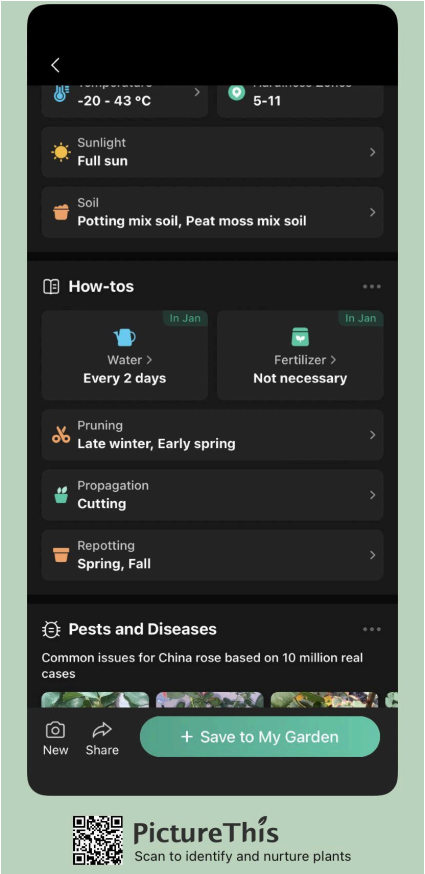


*Figure 1.1: Picture This*

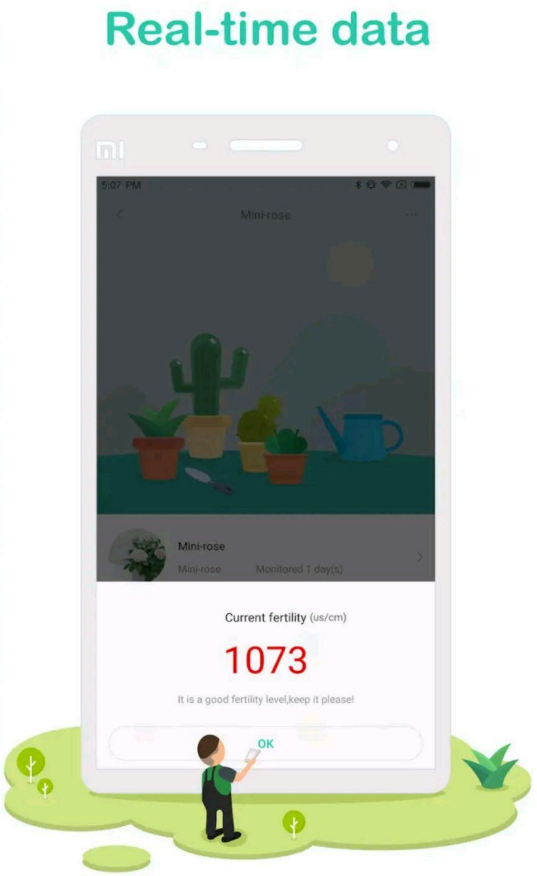


*Figure 1.2: FlowerCare*

### 1.3.2. Brief introduction about our application

It can be seen that IoT has brought many benefits to the field of flower cultivation, and our application is no exception. We have set the goal of developing a mobile app with essential features to support data measurement, featuring a user-friendly and flexible interface. This is some objectives of our application:

- The app not only allows users to view information about various types of plants worldwide but also provides the capability to directly monitor detailed data from their plant pots. This enables users to conveniently check the moisture, temperature of the environment around the plant, and the soil moisture, regardless of their location.
- The app also offers the ability to track long-term data, allowing plant growers to monitor the plant's parameters over different time intervals.
- Another crucial feature of the app is its ability to recognize when the plant needs watering or sunlight, sending alerts to the user.

This is our process in making this product:


*Figure 1.3: Overall pipeline for our application*

In the beginning, we prepared all required components for the Arduino and built a completed device. This device helps in collecting data from plants and their surrounding environment so that users can keep track of the plant's health. Because data is being updated throughout the device running, we need a database to store data. Another database responsible for the app is the real-time database, which serves to display the data in real-time to users. For the mobile app, we design basic UI and write functions that users can interact with the app. In the connection process, we connect the previous databases and web services to the mobile app. The last process is for updating our product. For the device, we modified it to make it portable and more stable, with the mobile app, we also updated new features for the app and beautilized the UI for better experience to the users.

**2. Report Structure**

*I. Introduction*: Introduce the problem, propose the approach, and present the solution leading to the benefits of using the app. Provide an overview of the app, its functionalities, and purposes.

*II. Requirement Analysis*: Describe the operation, functionality, and user convenience. Provide a detailed explanation of how the app operates and the programming techniques used, illustrated through diagrams.

*III. Design and Implementation*: Outline the sensors, their functions, and purposes. Explain how sensors are connected to Arduino, how information is received and stored on the server, the server's operation, and details about the databases and tools utilized.

*IV. Result and Discussion*: Present the images and content of the interfaces within the application, explaining their display purposes. Identify areas for improvement and highlight the challenges encountered during the project.

*V. Conclusion and Future Work*: Summarize the achieved features of the application, emphasize the key objectives of the application, propose solutions for challenges encountered during development, and outline future feature developments for the application.

## II. Requirement Analysis

### 1. System Requirements

#### 1.1. Functional Requirements

- Data Measurement: The application is centered around measuring crucial data related to plants and soil, encompassing temperature, environmental humidity, and soil moisture.
- Visualization: Users have the capability to view the values of each parameter and monitor data changes through interactive graphs.
- Plant Library: The app includes a comprehensive plants library, allowing users to access detailed information about various plants and soil types. Users can save their preferred plants for personalized monitoring.
- Customization: Users can personalize the application's interface and language settings according to their preferences.

#### 1.2. Non-functional Requirements

- Response Speed: The application demonstrates an average response speed, typically taking 3-4 seconds for users to receive results from the server upon accessing the app.
- User Interface Design: The interface is meticulously designed for optimal convenience and ease of use.
- Data Collection Process: Sensors collect environmental data, transmitting signals to Arduino. Python reads this information, sending it to the server for processing. The server converts data into numerical values and informational displays for users.
- User Request Handling: When users submit requests, the server processes the information, communicates with the database, and returns the processed results to the application. The interaction process is straightforward and user-friendly.

### 2. Use Case

#### 2.1. Use-Case diagram

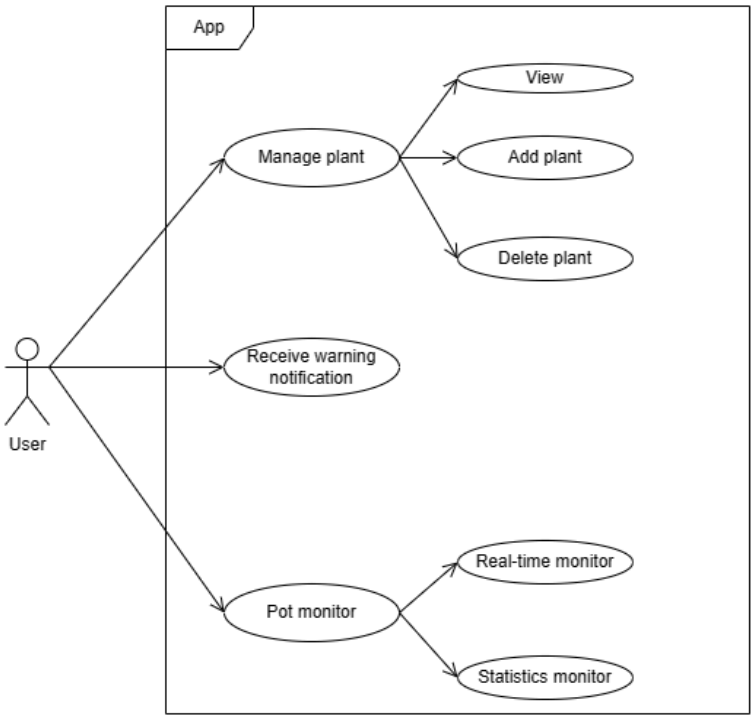


*Figure 2.1: Use-case diagram*

## 2.2. Use-Case Analysis

### 2.2.1. View plant

For the need to have a comprehensive list of various plant species worldwide to assist users in understanding and obtaining information about different types of plants such as flowering frequency, sunlight requirements, watering needs, and growth cycles—we utilized the API from the Perenual website. This platform provides detailed information about a wide range of plant species globally. By integrating this API into the application, users can easily view and track information about the plant species they are interested in.

### 2.2.2. Add plant

Monitoring and capturing information about plants based on a complete list of all plant species worldwide can make it challenging and inconvenient for users to manage and control their plants. To address this issue, we have added a feature that allows users the freedom to add new plants to their local data on their devices. The goal of this feature is to empower users to manage and control their plants in detail. With the ability to add plants, users can conveniently track and manage specific details about each plant they are cultivating in a personalized way.

### 2.2.3. Delete plant

When a plant is no longer in good condition within the management data or users switch to cultivating a different plant, it becomes essential for users to track and maintain an effective plant list containing only information about the plant species they are interested in. To enable users to efficiently manage and organize their relevant plant information, we have integrated a crucial feature into our application— the ability to delete a plant. This allows users to easily remove a plant from their management data. The delete plant feature provides convenience in managing and facilitates the organization of information that users care about while caring for their plants.

### 2.2.4. View your plant

With the issue of users adding plants without knowing their whereabouts, allowing users to view their plants directly on the main screen of the application will help them address the problem they are facing. Through this feature, users can monitor and manage all their plants right on the main screen of the application.

### 2.2.5. View real-time statistics

When the user checks statistics (temperature, humidity, soil moisture), there are 3 fragments corresponding to each statistics that are shown in a line with their title and displayed data next to the title. The data is fetched in a real-time database(Firebase) to serve the user in tracking the plant in real-time so that the user can decide on a suitable care plan for the plant(s).

### 2.2.6. View statistics graph

To visualize to the user the changes of data in the past, a graph is built in each title of statistics when the user clicks on the fragment. All three graphs are built in the UI system and the data is fetched in the storage database (MySQL). Once the plant is added to the data a planter is interested in, they will want to track important tree information such as average tree temperature and humidity levels to be able to create ideal conditions. about the environment for plants to grow. If the data index exceeds the average level, the application will send a warning to the user to maintain the measurement. This feature supports monitoring fluctuation in plant(s), helping the user understand better the changes in the environment, soil for better caring.

## 3. Sequence diagram

### 3.1 View Plant

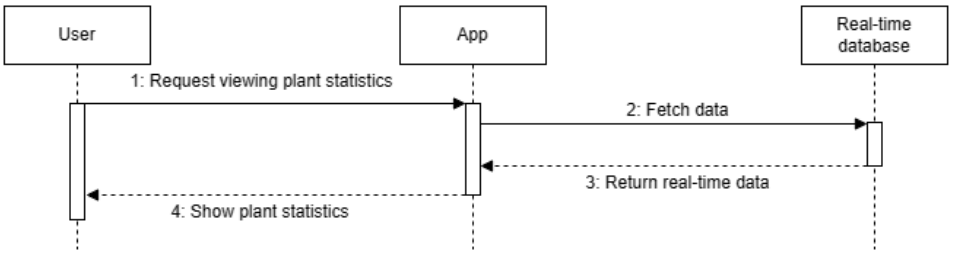


*Figure 2.2: View plant statistics in sequence diagram*

To view statistics of plants, the application fetches data from a real-time database then returns real-time data and visualizes it to the user.

### 3.2 Delete Plant

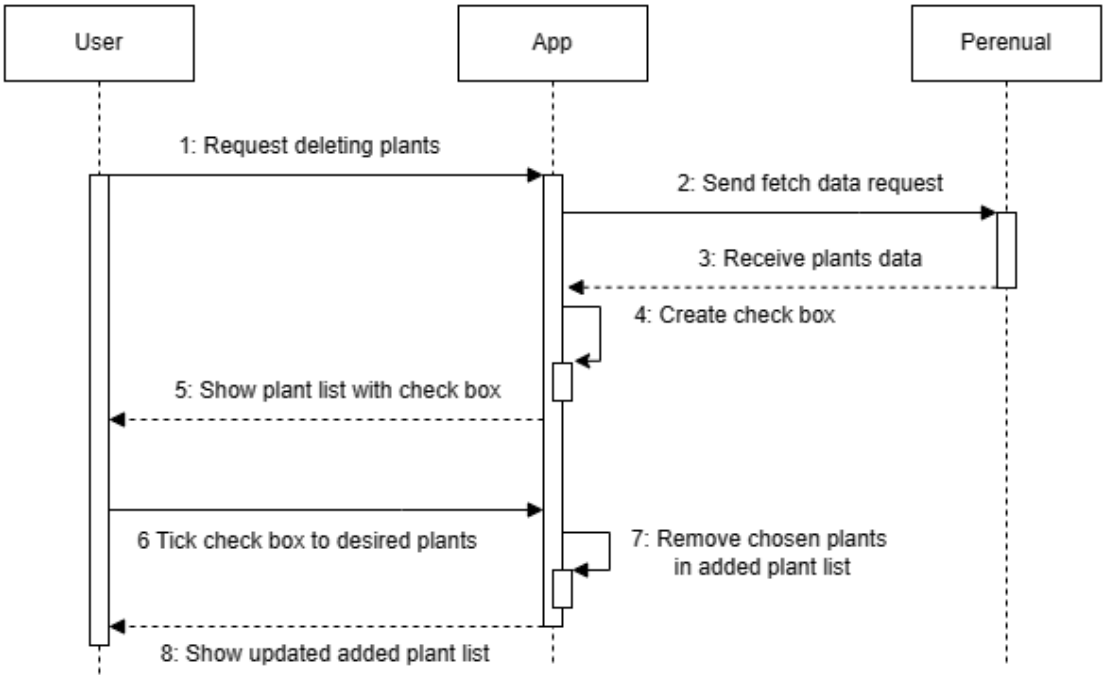


*Figure 2.3: Delete plant in sequence diagram*

When the user decides to delete more than one plant, the application will show a UI to let the user interact. After that, the application sends a request to Perenual then receives plant information and shows only the added plants before with a checkbox for multiple selection. Finishing selecting plants to delete, the application removes the chosen plants from the added list and returns the updated added plant list.

### 3.3 Add Plant

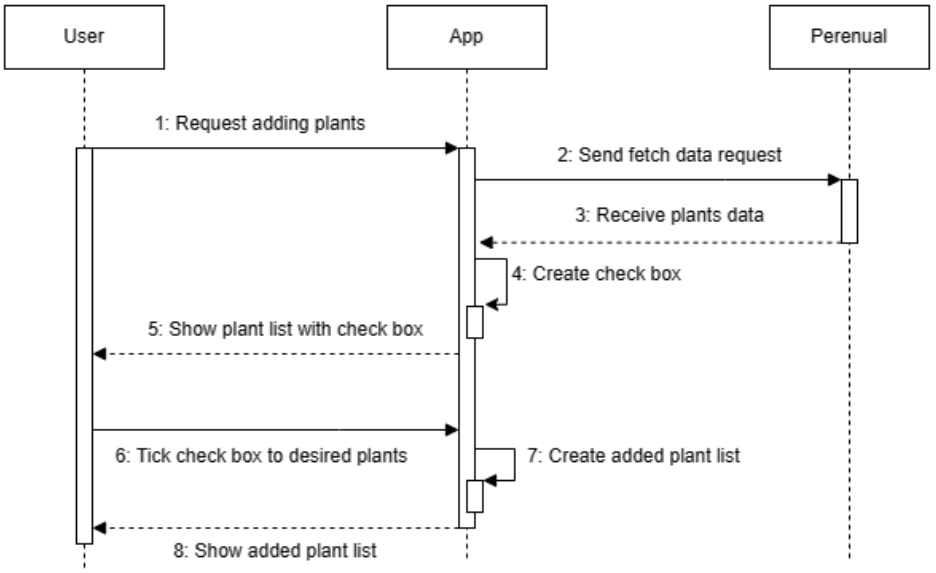


*Figure 2.4: Add plant in sequence diagram*

The user desires to add a plant or multiple plants, user needs to interact through the application UI. While interacting, the application sends a request to Perenual through API, the website gives the information of plants back to the application and shows all information in a list with a check box so that the user can choose multiple plants. After choosing plants and clicking the button, the application itself creates a list of added plants and gives them back to the user.

### 3.4 Show Plant Statistics Graph

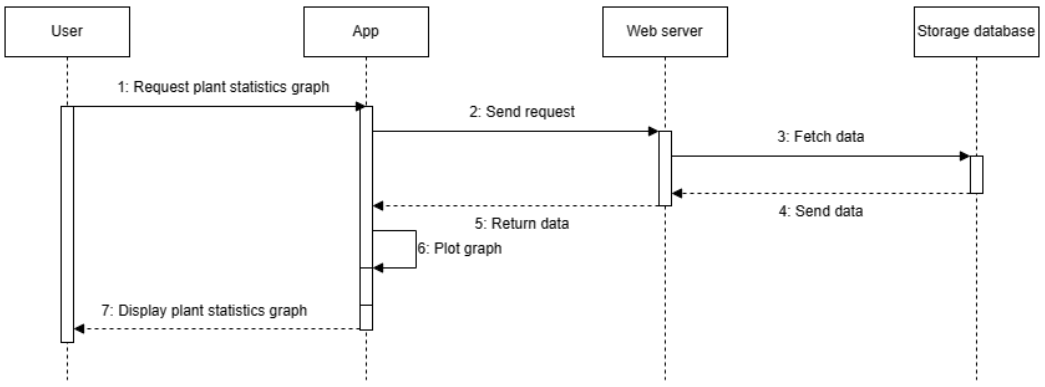


*Figure 2.5: Plant statistics graph in sequence diagram*

For showing the plant statistics graph, the application sends a request to the web server so that it can fetch data from the storage database to the application. Then a graph is plotted and shown to the user.

## III. Design and Implementation

### 1. System Architecture

#### 1.1. Overview

DHT sensor (environment sensor) and TH sensor (soil moisture sensor) are sent commands to read and receive data through Arduino Uno kit, Python calls the Arduino kit to push data from 2 sensors into a real-time database and storage database. In the real-time database, data is called directly by its dependency. For storage, our database to store data is MySQL so MySQL data is converted into JSON format then is called its API to the user. Another element used in the system is the Perenual JSON is called by its API to the user.

#### 1.2. System architecture diagram

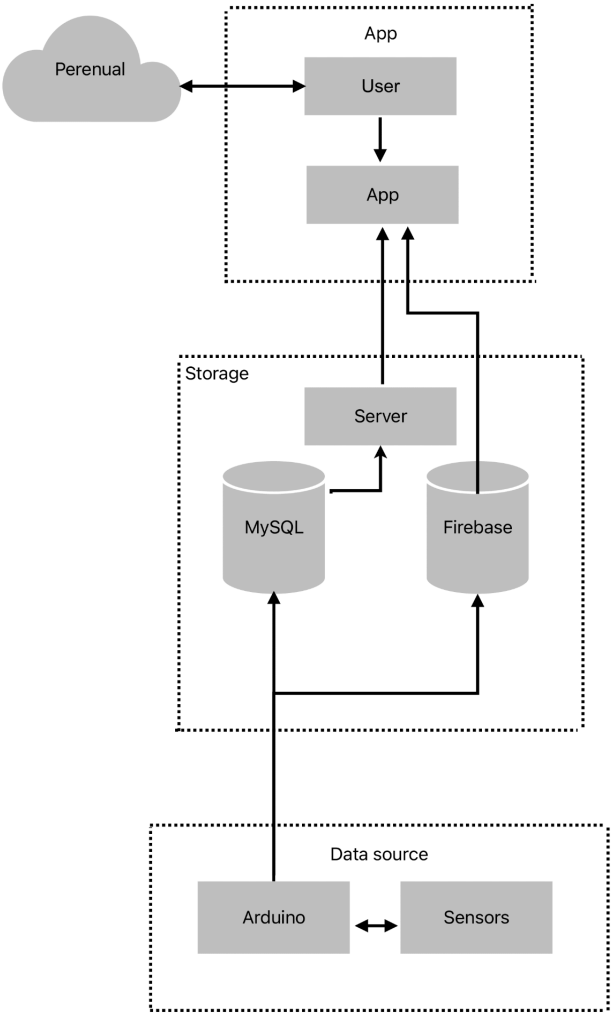


*Figure 3.1: System architecture diagram*

At first, Arduino read data from sensors then stored data in 2 databases: real-time database (Firebase) and Storage database (MySQL) by Python implementation. Through a Web server, data in the Storage database is sent to the application when the server receives a request and the real-time data from the real-time database is sent directly to the application. Another data serving to the application is the plant list from Perenual website is fetched through API.

### 1.3. Components

#### 1.3.1. Arduino set-up

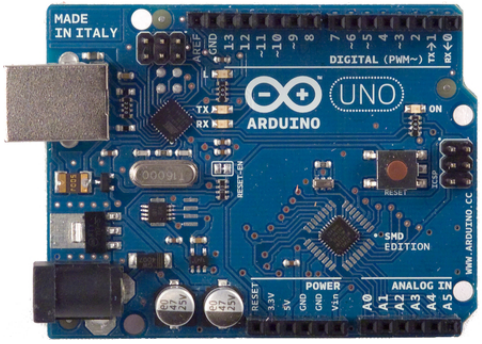


*Figure 3.2: Arduino Uno R3*

The Arduino Uno R3 is a microcontroller board based on a removable, dual-inline-package (DIP) ATmega328 AVR microcontroller, this is responsible for electronics and coding.

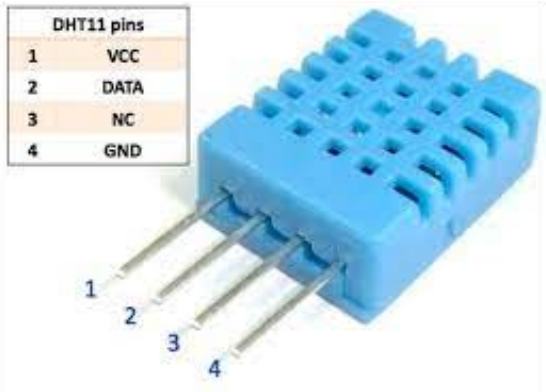


*Figure 3.3: DHT11 sensor*

The DHT11 is a basic, ultra low-cost digital temperature and humidity sensor. It uses a capacitive humidity sensor and a thermistor to measure the surrounding air and spits out a digital signal on the data pin.

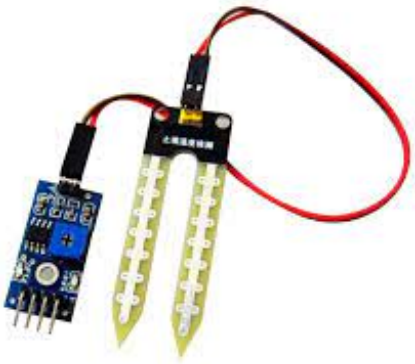

*Figure 3.4: TH sensor*

TH sensor is a basic and low-cost soil moisture sensor. It is used for measuring or estimating the amount of water in the soil.

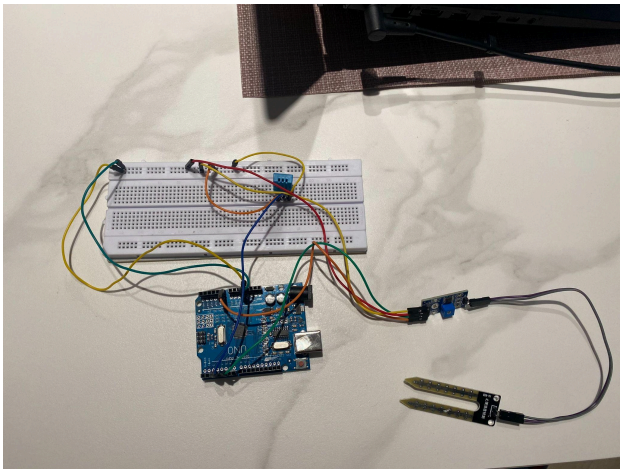

*Figure 3.5: Set up arduino*

For the DHT11 sensor, we plug wire into port 2 of the Arduino Uno R3 then connect to the DATA connector and another wire for the GND connector corresponds to the Arduino plug. Since the voltage requirement is 3.3V - 5.5V so we supply 5V to the sensor from Arduino by plugging in a 5V port. To make TH sensor works properly, there are 2 required components: an electronic sensor module, and soil moisture detector. At first, we connect soil moisture detector to electronic sensor module, we plug A0 connector from sensor module to A0 socket in Arduino kit, DO connector connect to the port 4, GND connector following to the GND socket in Arduino, we supply 5V for sensor the same as the previous sensor. In the VCC connector.

### 1.3.2. Arduino data retrieval

The Arduino Uno R3 recognizes connected ports from DHT11 and TH sensor. Implement code by using Arduino IDE, sensors are able to read real-world data and display in IDE terminal.

### 1.3.3. Python implementation

We implement Python code for reading Arduino data sensors and push data to storage database(MySQL) and real-time database(Firebase). In the beginning, we initialize the Firebase admin SDK, define Arduino serial port and Firebase Realtime Database reference. Next, we set up 2 databases, read data from Arduino and load data into both databases and set delay time for 2 seconds for next reading.

### 1.3.4. Convert Database to JSON

To process and display the data, We need to transfer them from the MySQL database to the application. However, the format of MySQL data is presented in a tabular form. Although the application has the ability to process such format, in the end, we decide to convert them into JSON format. The reason is due to the fact that the data retrieved from the perenual API is presented in JSON form, and using the same data format will simplify the process of developing the method to fetch data. Additionally, we utilize PHP to operate a local server and connect to the database. The server retrieves data from the database, converts it into

JSON format, sends the results back to the application, and repeats this process every 2 seconds.

| ID | temperature | humidity | soil_moisture | timestamps |
|---|---|---|---|---|
| 88 | 25.7 | 76 | 1019 | 2023-12-12 18:37:57 |
| 89 | 25.7 | 76 | 1019 | 2023-12-12 18:37:59 |
| 90 | 25.7 | 76 | 1018 | 2023-12-12 18:38:01 |
| 91 | 25.7 | 75 | 1018 | 2023-12-12 18:38:03 |
| 92 | 25.8 | 76 | 1019 | 2023-12-12 18:38:06 |

*Figure 3.6: Data in table format*


```
[{"ID":"88","temperature":"25.7","humidity":"76","soil_moisture":"1019","timestamps":"2023-12-12 18:37:57"},
{"ID":"89","temperature":"25.7","humidity":"76","soil_moisture":"1019","timestamps":"2023-12-12 18:37:59"},
{"ID":"90","temperature":"25.7","humidity":"76","soil_moisture":"1018","timestamps":"2023-12-12 18:38:01"},
{"ID":"91","temperature":"25.7","humidity":"75","soil_moisture":"1018","timestamps":"2023-12-12 18:38:03"},
{"ID":"92","temperature":"25.8","humidity":"76","soil_moisture":"1019","timestamps":"2023-12-12 18:38:06"},
{"ID":"93","temperature":"25.7","humidity":"75","soil_moisture":"1019","timestamps":"2023-12-12 18:38:08"},
```


*Figure 3.7: Data in JSON format*

## 2. Database design

### 2.1. Storage database

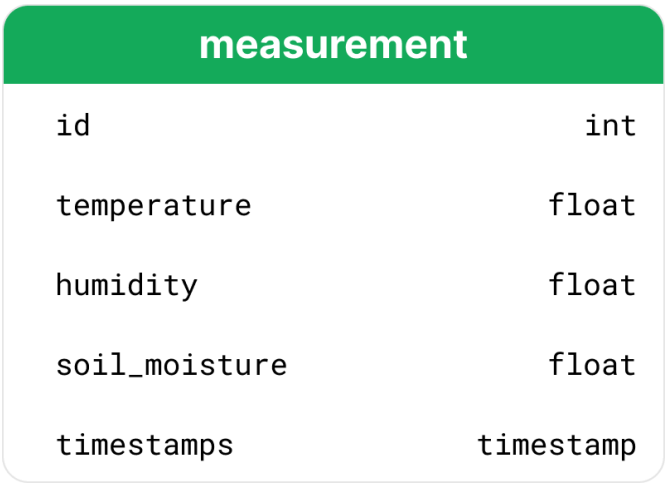


*Figure 3.8: Storage database*

- The “id” element represents the uniqueness of a measurement in a time.
- The “temperature” element is for the temperature of the environment.
- The “humidity” element is to store the humidity of the environment.
- The “soil_moistrure” is to store the data of soil moisture in the pot plant.
- The “timestamps" element is to mark the real-time measurement of each attemps.

### 2.2. Real-time database

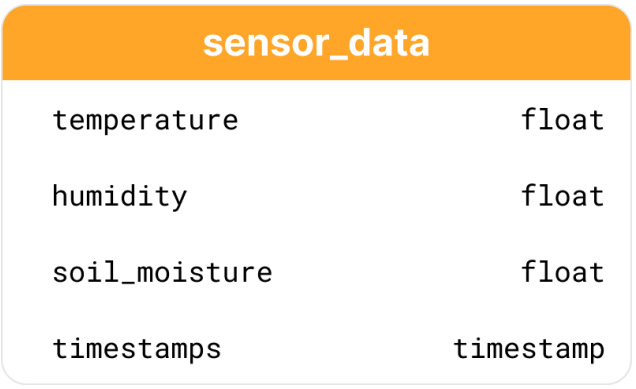


*Figure 3.9: Real-time database*

- The "temperature" part is all about the environment's temperature.
- The "humidity" part is there to keep track of how humid the environment is.
- The "soil_moisture," it's there to hold info about the moisture in the potted plant's soil.
- The "timestamps" part is meant for noting the real-time measurements during each attempt.

## 3. Implementation

### 3.1. Arduino

- DHT11: “DHT.h” an Arduino library for the DHT series of low-cost temperature/humidity sensors. We use this library to implement code to the sensor so that it can read and give environment data (humidity and temperature).
- TH: the sensor is implemented code immediately by defining I/O and set commands to read and return soil moisture data.
- Arduino Uno R3: set baud rate at 9600.

### 3.2. Android Studio

Android Studio is an environment dedicated to developing mobile applications running on the Android operating system. Regarded as the primary tool for building Android apps, it offers a range of features and comprehensive tools for developers. Android Studio stands out for its support of multiple screens, allowing developers to create applications for various devices. It integrates an Android emulator for testing apps on different versions. Android also supports languages such as Java and Kotlin, enabling developers to leverage new features, reduce development time, and optimize the user experience.

#### 3.2.1. Java

Java is a cross-platform, object-oriented programming language that is network-centric and can serve as a platform itself. It is a fast, secure, and reliable programming language used for coding everything from mobile applications and enterprise software to data applications and server-side technologies. Java is a versatile language designed for ease of use, making it a highly popular choice among software developers. Java is widely recognized as a leading programming language for creating applications for mobile devices running the Android operating system. This language is commonly chosen for building applications for both phones and tablets. With the provision of tools like the Android SDK, Java provides robust support in the development process of mobile applications.

### 3.2.2. API

APIs are essential software components that come with the Java platform. These are pre-written Java programs that can be "plugged in and played" to provide existing functionality in one's code. Every Java application written by developers typically combines both new and existing code from Java APIs and libraries.

### 3.2.3. Shared Preferences

Shared Preferences is the simplest and lightweight data storage mechanism in Android, used to store and retrieve key-value pairs. It is a common way to store application settings, such as the app's state, user preferences, or any small data that the app needs to store and retrieve when the app restarts. Due to its benefits and convenience, we have utilized Shared Preferences to store all data locally on the device and apply it to various features such as adding plants, removing plants, and storing information about previously added plants.

### 3.2.4. Dependences and External Libraries

PhilJay MPAndroidChart: To draw and display charts within the application, we used MPAndroidChart, an open-source library for Android designed to provide powerful and flexible chart components to help developers easily visualize data in charts and graphs quickly. We chose MPAndroidChart because it is a user-friendly library that supports various types of charts for different purposes. The data is divided into different datasets for easy control and modification. Each dataset contains a set of "input data," and each dataset has its attributes and descriptions, such as color, size, etc. Additionally, the charts include axes displaying values, legends for identifying datasets in the chart, and many other features.

### 3.2.5. Fetch Data

To search for and load data from multiple sources into MainActivity, we have utilized various tools. Firstly, we use Retrofit and OkHttpClient to search for and load data from the Perenual API. OkHttpClient is an HTTP client library for Java and Android, performing well in handling HTTP requests and interacting with web APIs. It is a crucial component that can be used independently or in conjunction with Retrofit to execute network requests. Retrofit acts as a type-safe HTTP client for Android and Java. Next, Firebase is managed through FirebaseManager to monitor real-time values. Data from SQL is externally converted into JSON through PHP. Subsequently, we use Retrofit and OkHttpClient to retrieve data from JSON and integrate it into MainActivity. With this approach, the database is updated every 2 seconds through the PHP task, ensuring data refresh for the graph. FirebaseManager also performs updates every 2 seconds, ensuring that both sources provide the latest data. The process is designed to efficiently fetch data from different sources and provide real-time updates to MainActivity, creating a dynamic and responsive user experience.

### 3.2.6. Picasso

Picasso is a widely used image processing library in Android Studio, developed by Square. It facilitates the management and display of images from various sources such as the Internet or local storage. In this project, Picasso serves as a flexible tool to load and display tree images for users.

### 3.3. Databases

Real-time database: Firebase is a mobile and web application development platform provided by Firebase inc (Google). We utilize its noSQL cloud database to store and synchronize the data in real-time. Data received from the arduino is transferred to the firebase database through a python program. The database is then connected with the application through Android Studio built-in library and tools. In a cycle of two seconds, the data will be stored in the database and at the same time, the application will read the updated value and display it for the users. We chose Firebase because of its ease of use and the app's requirement for a real-time database to store and synchronize data directly across user devices. We implemented Firebase to bring data to the app quickly and accurately.

Storage database: The storage database is based on the MySQL database. MySQL is an Oracle-backed open source relational database management system (RDBMS) based on Structured Query Language (SQL). To connect the application to the database, a web server is used - Apache, this web server is to receive requests from the application and return data from the database to the application. Thanks to its connectivity, speed, and security make it highly suited for accessing databases on the internet, MySQL became our optimal option for storage.

PHP: We implement the web server for interactions between the Storage database and the application using PHP. PHP is a server-sided programming language, which means it helps create the logic and functionality that operates on the web server. The web server’s role is to take requests from the application and find the correct data from the database, then it returns the data retrieved to the application for further use.

## IV. Result and Discussion

### 1. Result

### 1.1. Mainactivity

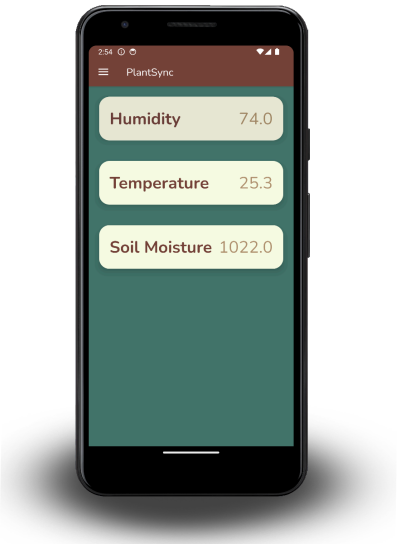


*Figure 4.1: Interface when the app is opened*


*Figure 4.2: Detailed chart*


*Figure 4.3: Detailed chart*

In the main activity interface, users can view basic information such as temperature, humidity, and soil moisture in three card views. The actual data retrieved from Firebase is displayed on the right side of each card view. After successfully adding a plant, the user will be returned to the main activity, as mentioned above. Then, the main activity will have an additional card view at the top to display the newly added plants. Users can select each desired plant to display the average bar on the chart. When selecting any plant, based on the pre-saved ID, the app will retrieve watering and sunlight data and calculate the plant's average level. It will then return to the user an average bar on the three charts in the main activity. The average bar helps users easily identify the plant's current status, such as when the plant is stable, needs watering, or needs to dry further. If the average bar exceeds the threshold, the chart will change color to alert the user. If there are no changes, the chart will remain green. If there is no longer a need for a plant, users can remove it from their plant list using the remove button in the navigation drawer.

### 1.2. Navigation Bar Drawer


*Figure 4.4: Navigation bar is opened*

The Navigation Bar functions as a switch frame between different activities in the app. When the user clicks on the three horizontal lines in the top-left corner, they open the navigation bar and can choose different items to customize, such as adding trees, deleting trees, settings, and the list of trees.

### 1.3. Plant ListActivity

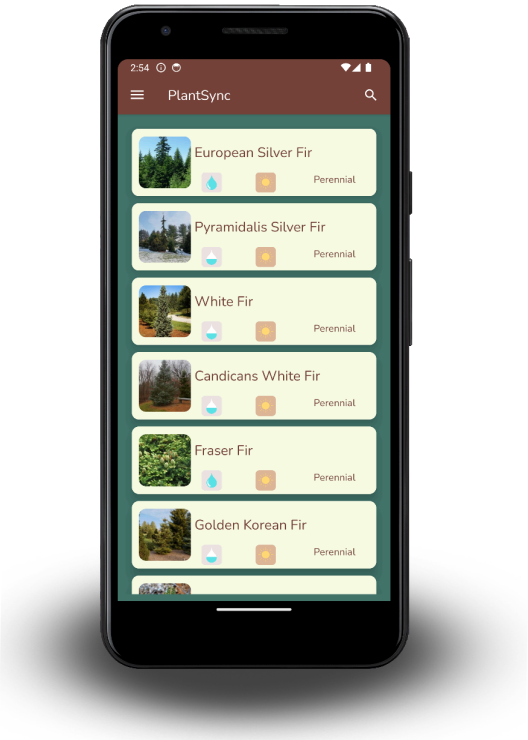


*Figure 4.5: View plants list bar*

This is where the suggested plant types are displayed, imported from Perenual. Using a RecyclerView and employing repeated fragments to display each added plant type. We use RecyclerView because it can handle large amounts of data and efficiently reuse fragments outside the visible area of the phone, appearing as we scroll down. When opening each fragment, it leads to a detailed activity, making it easy for users to interact and grasp information. Users can search for plants based on preferences or by specific existing types. When searching, the system will dynamically update the list without the need to press a search button.

### 1.4. Details Activity

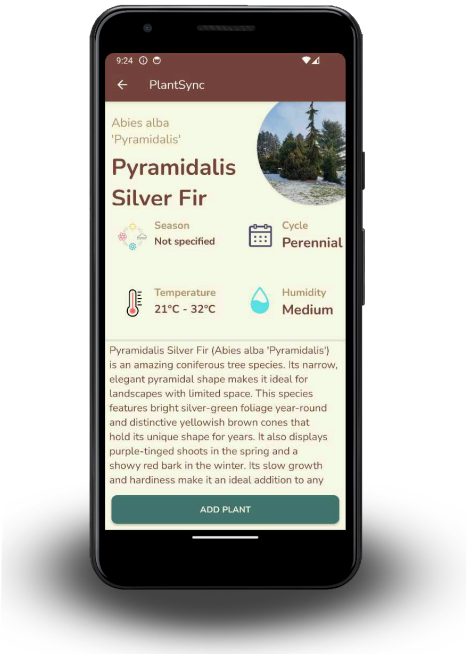


*Figure 4.6: Details activity with remove button*

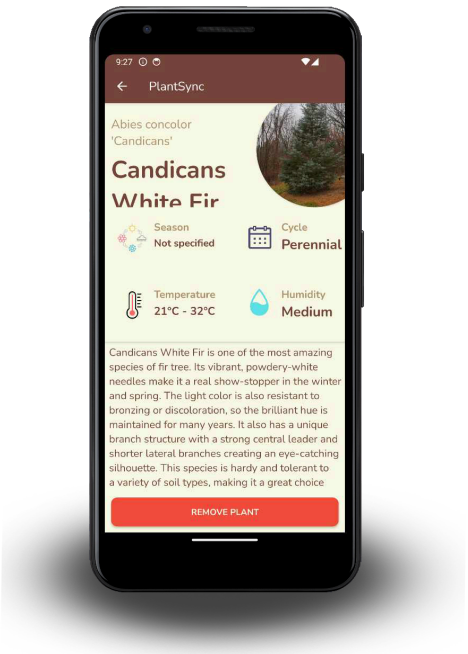


*Figure 4.7: Details activity with add button*

The Details Activity will again fetch data from the Perenual API, using the Details API to display more detailed information about the selected plant, such as flowering season, scientific name, description, etc. The Details Activity can be used to convert the sunlight data to temperature in a more readable format. For example, "Full sun" might be translated to a temperature range of 25 to 32 degrees, providing users with a clearer understanding of the plant's temperature requirements. This makes it easier for users to intervene in the plant and soil environment. It can also display corresponding icons based on the flowering season, such as spring, summer, autumn, or winter. If the specific season cannot be determined, a default icon will be displayed. We also added an "Add" button for users to add more plants if needed, with functions similar to the "Add" button in the Add Plants activity. Additionally, there is a "Remove" button to delete a plant when the user's needs are fulfilled or the plant is no longer in use. When pressing Details, the app will check chosen plant id with current plant id, the Add Plant button will appear when the chosen plants does not have id in current plants. After that, the Remove Plant button appears when the chosen plant id exists in our data.

### 1.5. Add Plants Activity

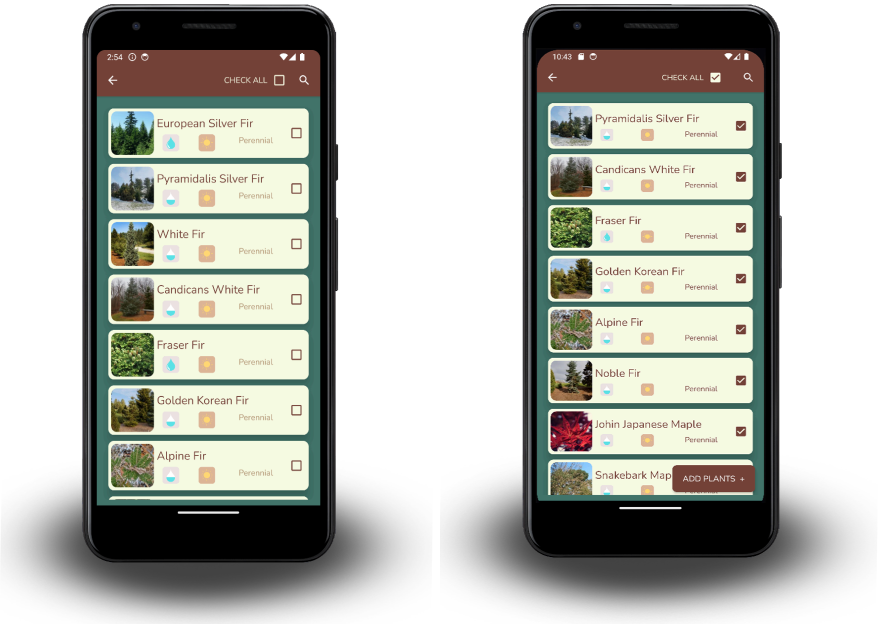


*Figure 4.8: Add plants with checkbox*

There will be a list similar to the "Plants List," but with the addition of checkboxes for users to tick to select the desired number of plants. After choosing the desired plants, there will be a "Check All" button in the top right corner for users to check all if they want to add all plants to the control section. Additionally, when checking at least one checkbox, an "Add Plants" button will appear for users to press. When pressed, it will save the ID using Shared Preferences to store locally and complete that activity. After that, users will be taken back to the main activity.

### 1.6. Remove Plants Activity

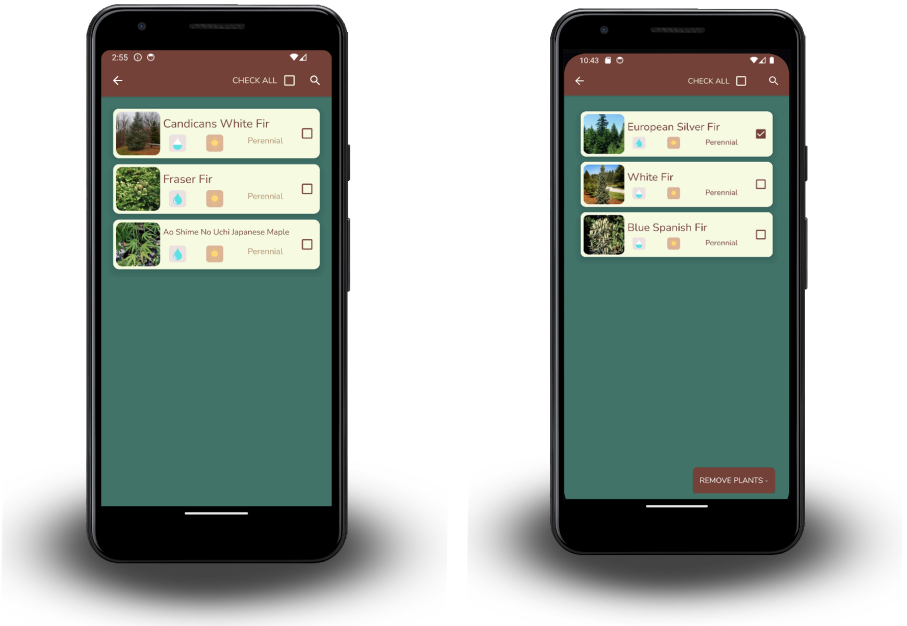


*Figure 4.9: Remove plants with a checkbox*

It will display a list similar to the "Add Plants" activity, but now it will show "Your Plants" based on the IDs saved in SharedPreferences instead of using the API. Additionally, there will still be checkboxes for users to select multiple plants they want to remove, with a "Check All" button in the top-right corner of the screen. Similar to adding plants, when at least one checkbox is checked, the "Remove" button will appear, and users can proceed to remove the selected plants. Upon removal, it will delete the corresponding IDs from SharedPreferences and return users to the main activity, finishing this activity. Users can directly check on the first CardView of the Main Activity or navigate to the "Your Plants" activity through the Navigation Bar Drawer.

### 1.7. Your Plants Activity

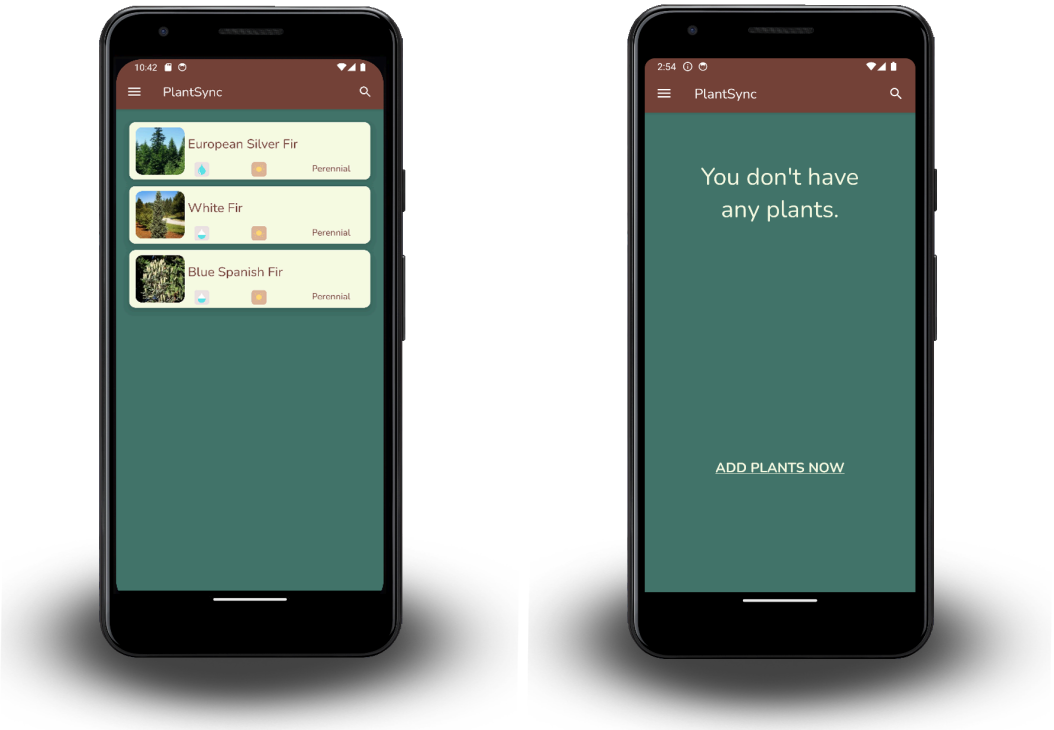


*Figure 4.10: Your plants activity interface before and after add plant*

The interface will display a list similar to the "Plants List" activity, but only show the list of the user's plants. If there are no plants, the interface will display a single line of text indicating that there are no available plants. If there are plants, users can access and view the details activity for each plant. In the details of each plant, there will be a "Remove" button instead of an "Add".

### 1.8. Setting Activity

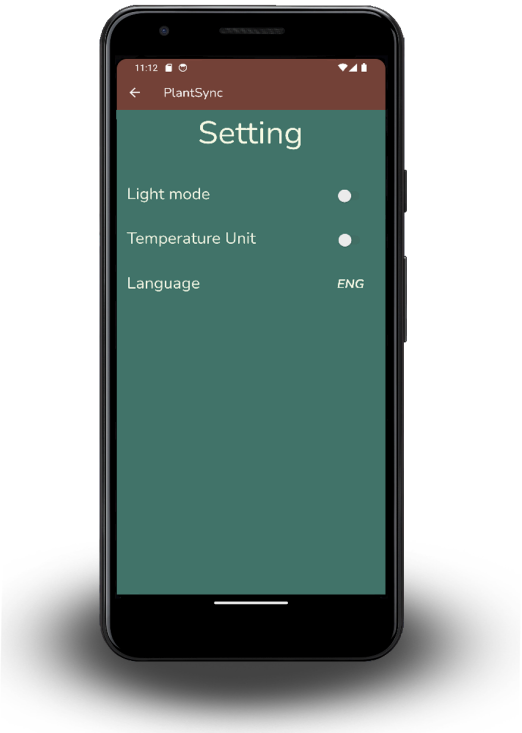


*Figure 4.11: Setting activity with custom functions*

This activity allows the user to modify the appearance of the application in three ways: switching between light and dark mode, switching the unit of temperature and changing the language of the app. There is a switch button next to the option “light mode”, permitting the user to change the display between light and dark mode, whichever suits the users’ preference. Next to the option “Temperature unit” there is also a switch button, allowing users to choose between Celsius or Fahrenheit. The last option “Language” will open a list of three possible languages: English, Vietnamese and French. This allows many foreign users to use the application without the concern of language barrier.

**2. Discussion**

The current state of the application reveals several challenges that need to be addressed for an improved user experience:

One prominent issue involves the handling of RecyclerView and ListView. The loss of checkbox status when scrolling down and the subsequent termination of the Fragment pose a significant inconvenience. Another critical concern lies in the performance of expanding CardView in the MainActivity, particularly when dealing with a substantial number of plants.

Security in network communication is an unaddressed aspect of the application. Using the laptop's IP for server access without implementing proper security measures poses potential risks. Another issue regarding security is the API key security. During development, the API key string is written directly into the source code. The reason is that it is more simple and flexible for the development team to test and switch between API strings in case one of the keys is blocked due to high requests.

Working with the API is also a challenge by itself. Because the app is using a third-party API for data, there are some noticeable problems. Firstly, the website provides us with API keys, each key can only allow a limited amount of data to be fetched. Therefore, a lot of API keys are blocked and that significantly slows down the developing process. Secondly, the data from the API sometimes does not return. This is due to the issues from the API provider and there is little we can do.

Lastly, for existing added plants, when the user desires adding another plants, the list still shows the same plant which was added previously, causing the duplication when adding the same plant one more time.

We have stated some solutions for our issues:

For the loss of the check box when scrolling down, we will add a boolean value into the plants list so when the checkbox is ticked, the boolean value is changed. From that, the chosen plants will always be checked even if the fragment is terminated when scrolling back, the plants are reloaded again with the same changed boolean value.

To prevent the expansion of Cardview when adding too many plants, we add the solid value for its height so that when expanding Cardview will always stay the same height as set, we also change RelativeLayout to ConstrainLayout to set the height limitation for the component.

We have addressed several issues to ensure a smoother user experience. Additionally, we have invested in the premium version of the Perenual API, enabling users to access a wider variety of plant images for reference and unlimited request time to get plants list.

For handling adding new plants, we solve it by returning data to plant list data so that it will not appear again if there is a similarity between Id and SharePreferences.

With a better budget, we can apply the nutrient sensor to the plant pot. When the nutrition rate is too low, the app will send warning notifications to remind users to fertilize the plants.

There are some features we have planned before such as an automatic watering system, which helps plant growers to water plants automatically when the plants are dehydrated but due to the time limitation, we are not able to add this feature to our app. Also, we planned to apply another measurement for the soil which is the nutrient measurement. The soil nutrient sensor ES-NPK-01 was planned to connect in Arduino Uno R3 and was applied similarly to previous sensors. Because the cost of the sensor is out of our budget, it is very challenging to approach this sensor at this moment.

## V. Conclusion and Future Work

### 1. Conclusion

Our application is designed to seamlessly read data from Arduino, ensuring real-time updates for users to monitor their plants from anywhere. The user interface is intentionally user-friendly, with a simplified, uncluttered design featuring soft colors and just enough features. This approach allows users to quickly grasp the app's functionality without the need for extensive learning. Our application serves as a valuable tool for users to access essential data such as soil moisture, temperature, and ambient humidity. This enables users to effortlessly monitor and comprehend the current status of their crops. We have incorporated the Perenual API, which furnishes comprehensive information on plant varieties worldwide, into our application. This integration empowers users to easily access and track relevant information.

The application caters to plant enthusiasts, offering solutions to challenges commonly faced in nurturing plants at home. To enhance user experience, we have included a chart outlining the optimal temperature and humidity levels for plants. If the data exceeds the average levels, the application promptly sends warnings to users, prompting them to take necessary measures and maintain the ideal environment for their plants.

In essence, our application aspires to be a valuable tool, aiding users in addressing the intricate challenges of effective plant care. We hope it becomes an indispensable resource for users seeking practical solutions to plant care dilemmas.

### 2. Future Work

One significant enhancement will be the introduction of an automatic watering feature. This feature will be facilitated through a relay connected to the Arduino, which will receive signals to activate the relay and water the plants if the humidity falls below specified levels. The user can water plants manually in the app or set the time to water, the user can water the plants according to the time cycle like for every 2 hours for example. This automated watering system will add a layer of convenience for users, ensuring their plants receive optimal care.

To bolster app security, we will implement fixes for network vulnerabilities. This will ensure a more secure environment for users, preventing unauthorized access. Our commitment to security will be reflected in these updates, providing users with a safe and reliable platform.

We will develop on the iOS platform, allowing a broader audience to benefit from our app's features. We will also develop a unique website for our app. We are dedicated to continually improving and expanding our application to meet the needs of plant enthusiasts everywhere. Last but not least, we will look forward to pushing the application to CH Play and App Store for users to download and experience the app.

## VI. References


[1] Vadivelu, V., Nachimuthu, P., Ravi, A., & Thatchanamurthy, A. K. (2023, August). An IoT adoption in agriculture: Challenges and futuristic directions. In *AIP Conference Proceedings* (Vol. 2857, No. 1). AIP Publishing.

[2] Ju, A., Wang, Z. (2023). A novel fully convolutional network based on marker-controlled watershed segmentation algorithm for industrial soot robot target segmentation. *Evol. Intel.* 16, 963–980.

[3] Hatfield, J. L., & Prueger, J. H. (2015). Temperature extremes: Effect on plant growth and development. Laboratory Director and Supervisory Plant Physiologist and Micrometeorologist, National Laboratory for Agriculture and the Environment, 2110 University Blvd, Ames, IA, USA.

[4] Monk, S., & McCabe, M. (2016). Programming Arduino: getting started with sketches (Vol. 176). New York: McGraw-Hill Education.

[5] Margolis, M., Jepson, B., & Weldin, N. R. (2020). Arduino cookbook: recipes to begin, expand, and enhance your projects. O'Reilly Media.

[6] Abbas, A. H., Mohammed, et al. (2014). Smart watering system for gardens using wireless sensor networks. 2014 International Conference on Engineering and Technology (ICET).

[7] Bin Sadli, M. D. D. (2019). An IoT-based Smart Garden with Weather Station System. In IEEE 9th Symposium on Computer Appl. & Indus. Electr. (ISCAIE) (pp. 38-43).

[8] N.Suma, Sandra Rhea Samson, S.Saranya, G.Shanmugapriya, R.Subhashri "IOT Based Smart Agriculture Monitoring System" International Journal on Recent and Innovation Trends in Computing and Communication, Volume: 5 Issue: 2, pages: 177 – 181.