Applied Information Technology and Computer Science Vol. 5 No. 1 (2024) 635-654
© Universiti Tun Hussein Onn Malaysia Publisher's Office

AITCS

Homepage: http://publisher.uthm.edu.my/periodicals/index.php/aitcs
e-ISSN :2773-5141

Design and Development of PM Application based on IoT Technology

Nur Hidayah Zaini[1], Hanayanti Hafit[1]*

Faculty of Computer Science and Information Technology,
Universiti Tun Hussein Onn Malaysia, Parit Raja, Batu Pahat, 86400, MALAYSIA

DOI: https://doi.org/10.30880/aitcs.2024.05.01.036
Received 24 June 2023; Accepted 18 May 2024; Available online 30 August 2024

Abstract: The Internet of Things (IoT) technology has widespread applications in sectors like agriculture, healthcare, business, and finance. This project focuses on using Microbit as a soil moisture sensor in agriculture. It connects to ThingSpeak and a mobile application that sends reminders to users for timely watering. The project aims to address the challenge of forgetting to water plants due to busy lifestyles, benefiting plant health. Objectives include designing the Microbit sensor, developing a monitoring app with reminder functionality, and conducting thorough testing. The project primarily supports farmers in dry regions of Perlis, Malaysia, aiming to enhance agricultural practices and manage plant moisture levels effectively for optimal growth. Based on testing with users, the application's user interface received positive feedback. Most features worked well, the reminder feature could be improved for timely notifications. Initially, setting up ThingSpeak and Microbit may require some time and understanding, but it becomes easier with experience.

Keywords: Plant, Microbit, Reminder, IoT, Application, Soil moisture, ThingSpeak

1. Introduction

Just as drinking water is necessary for human life, so too is the presence of water in a plant's environment for its continued existence. Plants receive their water from their natural source, which is precipitation in the form of rain. As a result, plants can rely only on rainwater during the rainy season, but they are unable to do so at other times of the year because there is a lack of rainfall during those other times of the year. Even when it's supposed to be raining, there is less precipitation in regions that are prone to drought. Therefore, in these kinds of situations, a method known as "reminder application," which is an artificial approach to water plants, is applied. This serves as a friendly reminder to ensure that plants receive the appropriate amount of water consistently. Because the application will prompt us to water the plant at the specific time that we have chosen, we won't forget to do it. Because of this, we can prevent the plant from dying by applying the solution, particularly on days when the temperature is high.

Besides that, the Internet of Things (IoT) is also important because it connects gadgets, vehicles, buildings, and other items with electronic devices and sensors to collect and exchange data. IoT combines wireless, microservices, the internet, and machine-to-machine (M2M) communication. M2M

*Corresponding author: hana@uthm.edu.my
This is an open access article under the CC BY-NC-SA 4.0 license.

communication involves machines connecting online. IoT is used in agriculture, health, business, and finance. As it relates to irrigation efficiency, soil monitoring is of utmost significance. Increased yields can be achieved by strategically timing irrigation to avoid overly dry soil. Hence, in this project, we used Microbit to measure the soil moisture which connects with an application to display a reminder to the users [1][2].

The application that will be developed is based on the Android and IoS operating systems. It is a plant care reminder application that will be designed to give a platform for users who want to be reminded when it is time to water their plants. The Microbit is used to establish a connection between this application and the soil moisture sensor. The application is geared toward farmers and gardeners who reside in arid areas. Hence, it will analyze the level of moisture in the plant's soil and send a notification to the user's device reminding them to provide the plant with moisture. The scope would be focusing on monitoring the plant in dry regions.

The objective of this project is to design a Microbit that will sense the soil moisture of plants. Besides that, it is to develop a plant monitoring application using a Microbit sensor to display a reminder. Lastly, to test the developed application.

2. Literature Review

There are several techniques utilized for monitoring plants and evaluating their health and growth. Visual inspection is a commonly employed method that involves observing plant characteristics, including leaf color, size, shape, and overall appearance. Changes in these visual features can indicate nutrient deficiencies, diseases, or other stress factors [3]. Other than that, the measurement of the leaf area index (LAI) is conducted using instruments like the LAI-2200 Plant Canopy Analyzer or through hemispherical photography. This technique offers insights into plant growth, light interception, and canopy structure [4]. There is also a thermal imaging camera that captures the infrared radiation emitted by plants, enabling the assessment of plant stress, water status, and thermal patterns [5]. But for this project, we used a soil moisture sensor that is utilized to measure the water content in the soil, providing information about the plant's water status and assisting in determining suitable irrigation schedules [6]. These techniques, along with others, provide valuable data for plant monitoring, facilitating informed decisions on plant care, resource management, and enhancing crop productivity.

2.1 Related Work

Various existing plant mobile applications are ready for use. This section is to study the features and functionality of the existing systems and makes a comparison with the proposed application. Three similar applications chosen are Plantnote: Plant Diary & Water [7], Blossom: Plant Identifier [8], and Plant Parent [9]. Table 1 shows the comparison of the mobile applications.

2.2.1 Plantnote: Plant Diary & Water

Plantnote is an app that can be used on smartphones that work with both the iOS and Android platforms. A committed bunch who think it's important to keep the plants alive and document their progress in a journal [7]

2.2.2 Blossom: Plant Identifier

Blossom is an application that works on Android and iOS. It has the capability of accurately diagnosing issues with plants. In addition to this, it offers the consumer assistance from a botanist if they have any inquiries concerning their plants [8].

2.2.3 Plant Parent

The application Plant Parent may be downloaded on both Android and iOS devices. It is supposed to be a very complete plant care and informative instructional application for anyone who appreciates plants, trees, shrubs, and all other types of landscape and natural plants [9]

Table 1: Comparison of the existing system

Characteristics	Plantnote	Blossom - Plant Identifier	Plant Parent	Proposed application
OS	Android, IOS	Android, IOS	Android, IOS	Android
Fee package	✓	✗	✗	✗
Module	Admin	User, Admin	Admin	User
External cable	✗	✗	✗	✓
Menu	✓	✓	✓	✓
Login requirement	✗	✓	✗	✓
Update personal information	✗	✓	✗	✓
Plant list	✓	✓	✓	✓
Diagnose disease	✗	✓	✓	✗
Embedded chart	✗	✗	✗	✓
Water reminder	✓	✓	✓	✓
Soil moisture sensor	✗	✗	✗	✓

3. Methodology

The term "methodology" refers either to the theoretical analysis of the procedures that are suited to a field of study or to the collection of methods and principles that are specific to a branch of knowledge [10]. Following this, a prototype is characterized by a certain role. One definition of prototypes states that they are models that are used to show the form and feel of a product [11], while another definition states that prototypes are full-scale pre-production models that are used for testing functionality and manufacturing processes [12].

Hence, one of the models for the software development life cycle is referred to as the Prototype model. In this model, a prototype is constructed using the bare minimum of needs. The client provides feedback, the prototype is evaluated and improved based on that feedback, and the process continues until a final prototype with the needed functionalities is generated. The finished prototype will also serve as the foundation for the product's ultimate form [13].

3.1 Prototype Model

3.1.1 Planning

During the planning phase, the project's foundation is established through a series of steps. Firstly, problem statements are identified, objectives are defined, and the project scope is outlined. The expected result and project significance are also considered at this stage. Secondly, comprehensive research is conducted on the project's title, gathering relevant information and building a deep understanding of the topic. This involves reading articles and exploring existing resources related to the subject matter. Additionally, an in-depth study of similar previous projects is undertaken to learn from their successes and challenges. Lastly, a Gantt Chart is created, providing a visual timeline and key milestones to ensure effective project management and timely completion. The deliverables for this planning phase include a project proposal that covers problem statements, objectives, and scope, a comparison analysis between the existing and proposed projects, and the finalized Gantt Chart.

3.1.2 Analysis

During the analysis phase, a questionnaire is created using Google Form and distributed to farmers or gardeners. Their responses are collected and analyzed to identify user requirements. Hardware and software requirements are examined, and functional and non-functional requirements are determined. The deliverables include the questionnaire in Google Form, user requirements, hardware and software requirements, and functional and non-functional requirements.

3.1.3 Design

In the design phase, wireframes and a prototype are created to design the user interface and interaction flow. Additionally, the database requirements are defined to determine how the project's data will be structured and stored. The deliverables for this phase include the wireframes, prototype, and database requirements.

3.1.4 Implementation

In the implementation phase, the project progresses through several important steps. Firstly, the application's modules are identified, allowing for the breakdown of functionality into manageable units. This helps organize the development process and ensure efficient implementation. Next, the application is integrated by combining individual modules and components into a unified whole. This involves coding in Dart language, implementing algorithms, and utilizing the Flutter framework along with necessary libraries. Furthermore, the application is connected to the Firebase Firestore database, enabling seamless storage and retrieval of data as required by the project. Additionally, integration with the Microbit device is established, allowing for communication and utilization of pertinent data. The main deliverable of the implementation phase is the completed application, encompassing integrated modules, database connectivity, and Microbit integration.

3.1.5 Testing

During the implementation phase, system testing is conducted by involving real users who provide feedback through a distributed Google Form. This feedback helps identify any flaws in the system. The main deliverable of this phase is to improve the system based on the feedback received and the results of testing. This includes addressing identified issues and refining features to enhance the system's overall quality and user experience.

4. Results and Discussion

The implementation phase ensures that the developed application aligns with the specified requirements, while the testing phase verifies that the application is error-free. Additionally, this phase focuses on discussing the project's implementation process and how it was carried out.

4.1 Analysis

The proposed system has eight functional requirements as shown in Table 2.

Table 2: Functional requirements

Function	Functionalities
Register	This function allows users to register a new user to the application with a valid email address and password.
Login	This function allows users to input a valid email address and password as in the database.
Logout	This function allows users to log out from the application.
User management	This function allows users to update their profile details such as email, password, and phone number.
Plant management	This function allows users to add or delete, categorise, descriptions, chart API, and field API to the plants.
View Plant details	This function allows users to view the plant's name, category, and description as well as its soil moisture chart.
Receive reminder	This function allows users to receive reminders from the Microbit sensor to water their plants.
Microbit sensor	This function allows the Microbit sensor to sense the moisture of the plant.

There are four categories of non-functional requirements, which are operational, performance, security, and usability as shown in Table 3.

Table 3: Non-functional requirements

Function	Functionalities
Performance	The Microbit sensor can sense 1 plant's moisture at a time.
Security	Only users with a valid email and password can access the application.
Reliability	The application can function 24 hours per day.
Operational	The application should be connected to an internet connection to receive real-time database updates.

4.2 Design

This section explained the system analysis and design that have been conducted for this project.

4.2.1 Use case diagram

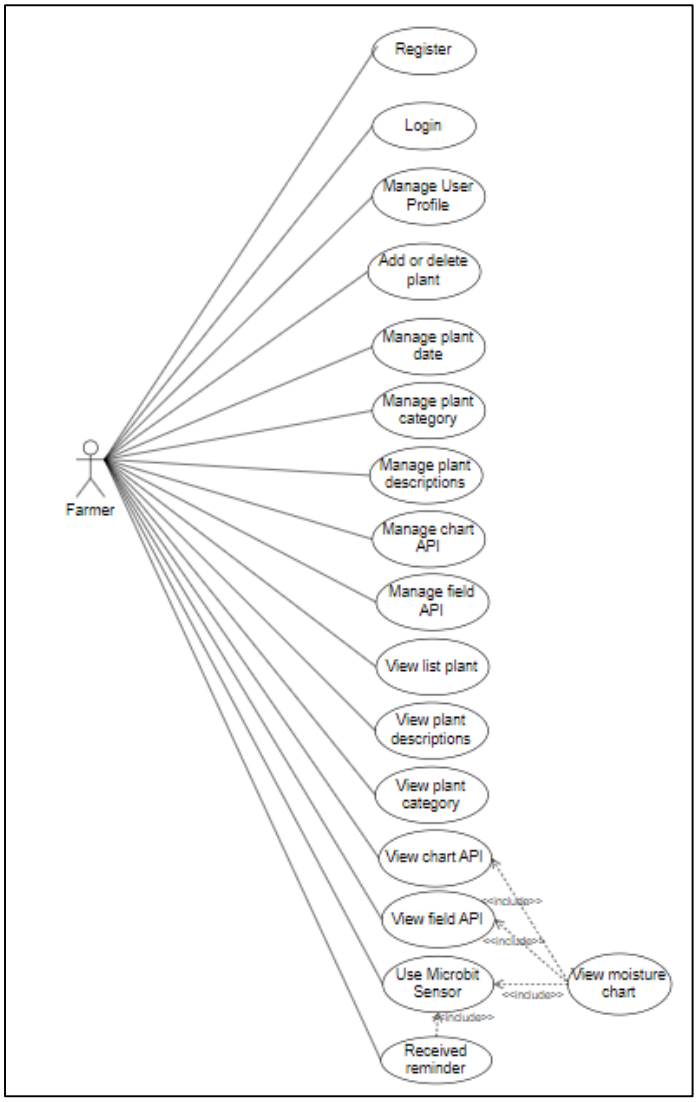

Figure 1: Use case diagram

Figure 1 shows the use case diagram for this project. Farmers can register, login, manage user profiles, add or delete plants, manage plant details, view list plants, view plant categories, use Microbit sensor, and receive reminders. They also can add plants to the application and set their details such as name, category, and descriptions. Most importantly, they will receive a reminder to water their plant on the application.

4.2.2 Sequence diagram

Register and Login diagram

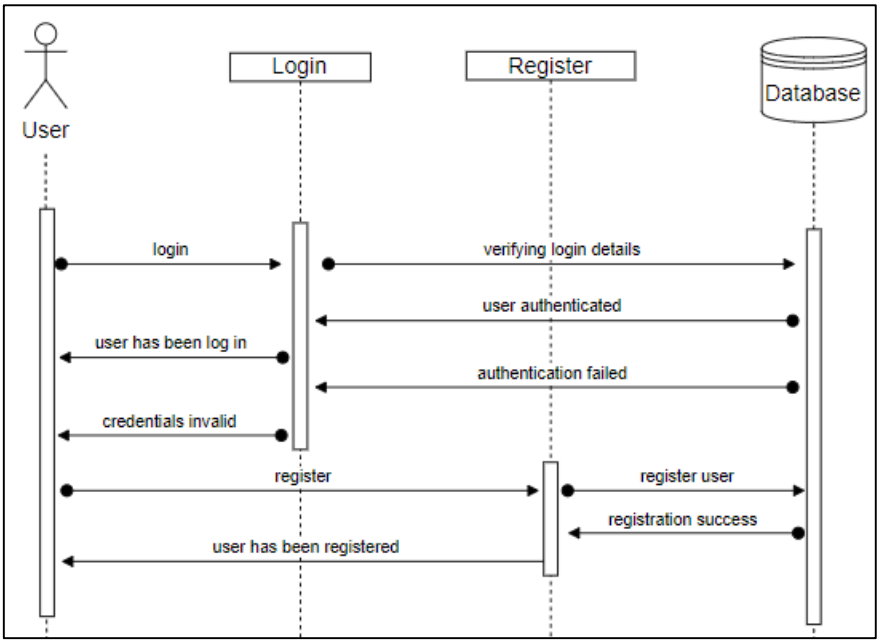

Figure 2: Register and login diagram

Figure 2 is a sequence diagram that illustrates the process of registering for and logging into the application. When a user attempts to register, the system will first check to see if the user's details are already stored in the database. If they are not, it will establish an account for the user and only the user will be able to use the application.

Plant Management diagram

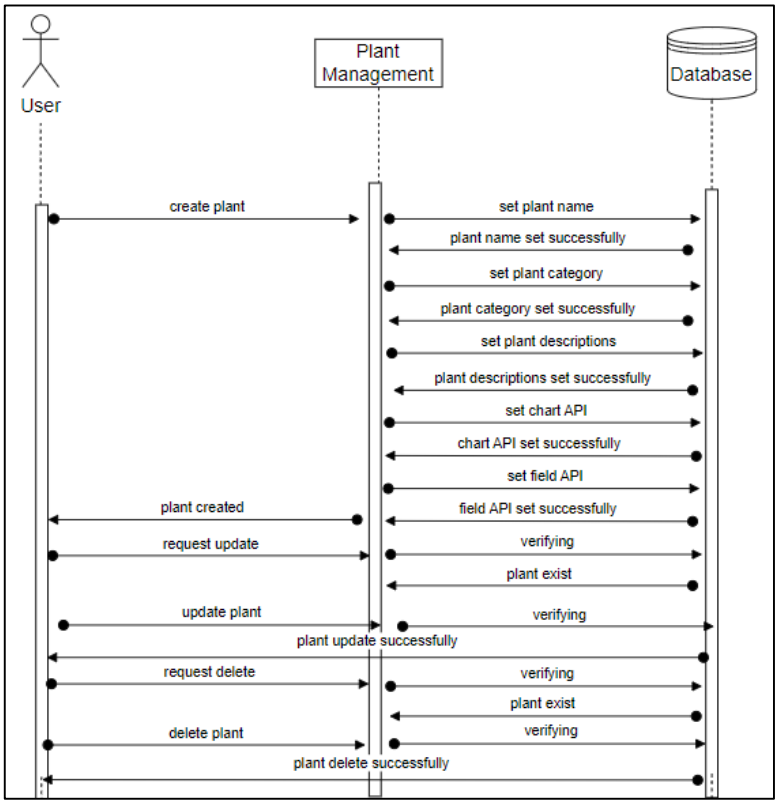

Figure 3: Plant management diagram

When it comes to plant management, it entails adding plants to the system by inputting their names, categories, descriptions, chart API, and field API. This is done so that they may be managed. The information will be stored in the database, and a link to the dashboard page will be provided to access it. The user can also alter the plant's details; but, before doing so, it will first check to verify if the plant already exists in the database. If the database is updated so that it conforms to Figure 3, then the specifics will be modified without any intervention on the user's part.

4.2.3 Activity diagram

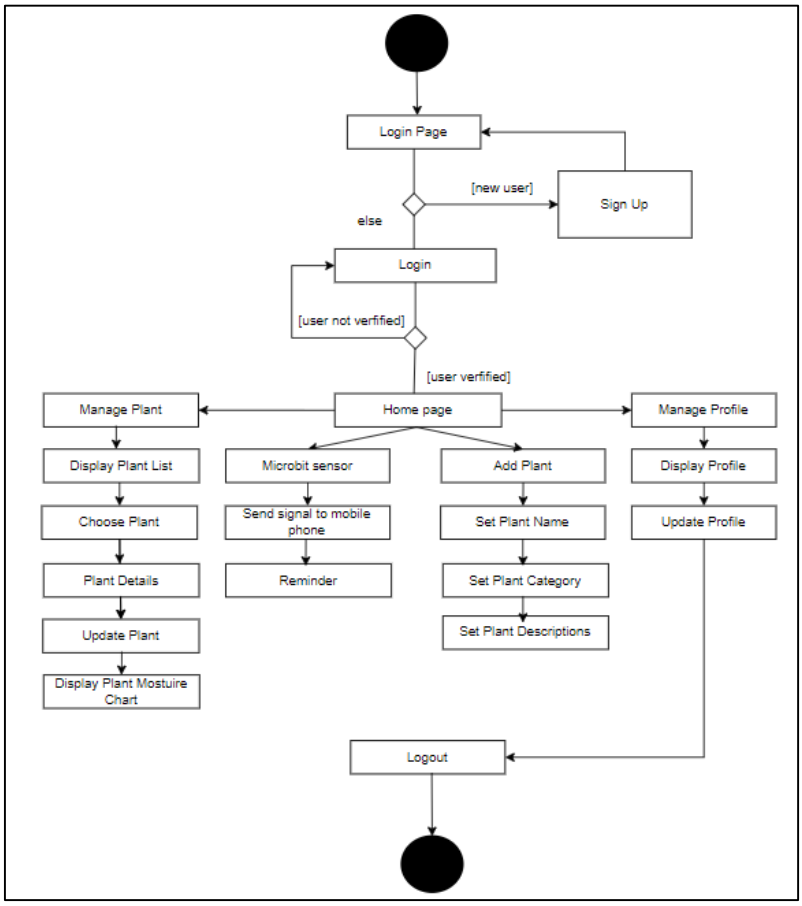

Figure 4: Activity diagram

Users of the application will be required to register their name, phone number, email address, and password. If they have not yet registered their account, they can do so by going to the signup page and entering their information from within the website itself. There are also pages for adding plants, reminders, and an update page for profiles. The user will receive a notification when it is time to water their plants because the device will send a signal to the application through its internet connection based on Figure 4.

4.2.4 Class diagram

There are 7 classes involved in this diagram. The user can access all the above classes by connecting the Microbit with the plant. This way it can read the plant's moisture chart and send reminders to the application if the plant has not enough water. A greater understanding of the overall schematics helps reduce the amount of time needed for repairs based on Figure 5.

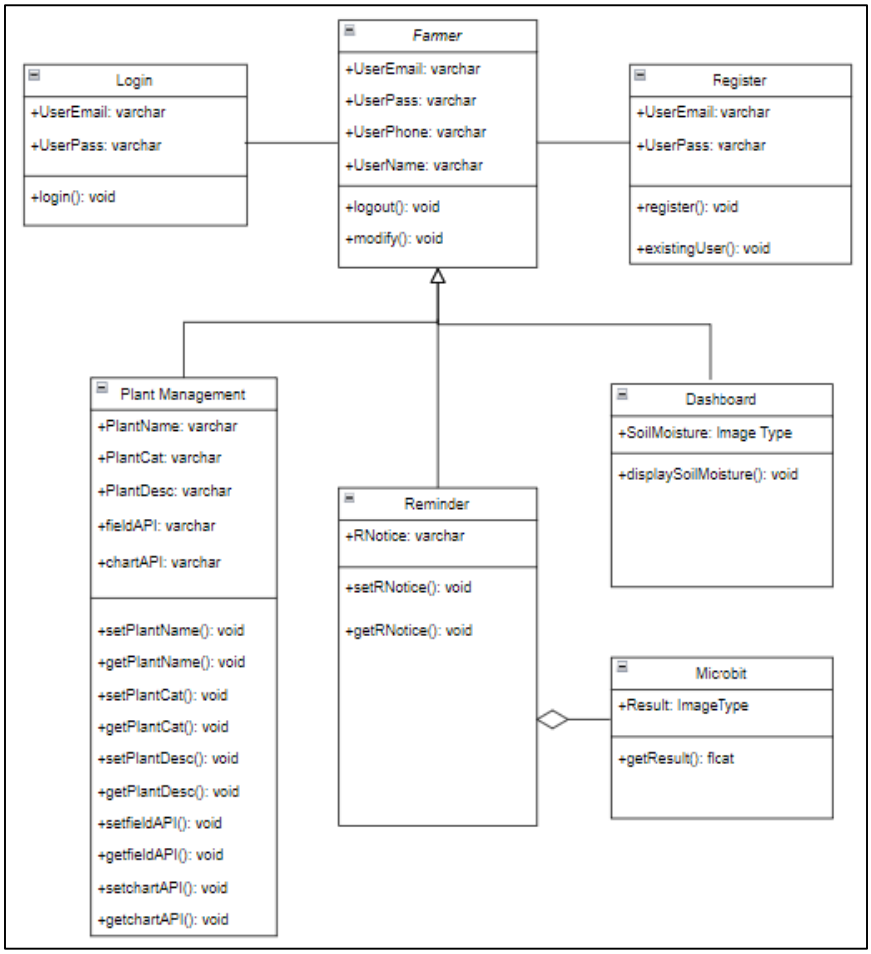

Figure 5: Class diagram

4.2.5 Data dictionary

This section will focus on the project's data dictionary for user, plant, and reminder.

User

Table 4 shows the user data dictionary which consists of the user's name, email, phone number, and password.

Table 4: User data dictionary

Field Name	Data Type	Field Length	Key	Description
email	Varchar	50	PK	User ID

Table 4: (cont.)

Field Name	Data Type	Field Length	Key	Description
fullName	Varchar	200		User full name
phoneNo	Varchar	12		User phone number
password	Varchar	12		User password

Plant

Table 5 shows the plant data dictionary which consists of plant's ID, name, category, and description.

Table 5: Plant data dictionary

Field Name	Data Type	Field Length	Key	Description
plantID	Varchar	50	PK	Plant ID
plantName	Varchar	200		Plant name
plantCat	Varchar	200		Plant category
plantDescription	Varchar	200		Plant description
chartAPI	Varchar	200		Chart API
fieldAPI	Varchar	200		Field API

Reminder

Table 6 shows the reminder data dictionary which consists of reminder's ID, and notice.

Table 6: Reminder data dictionary

Field Name	Data Type	Field Length	Key	Description
reminderID	Varchar	50	PK	Reminder ID
reminder	Varchar	50		Plant reminder

4.2.6 Wireframe

This section shows the wireframes of the project. Figure 6 are the interfaces for the register and login pages. Meanwhile, Figure 7 is a reminder and dashboard page.

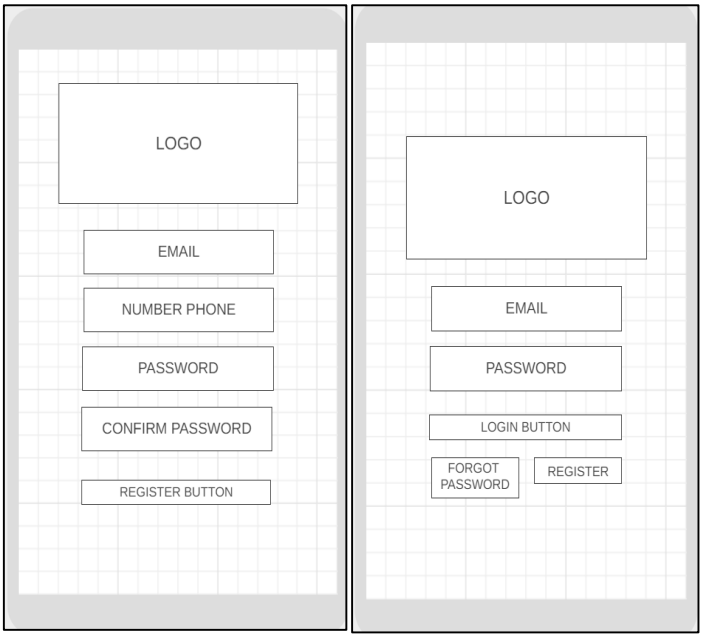

Figure 6: Register and login page interface

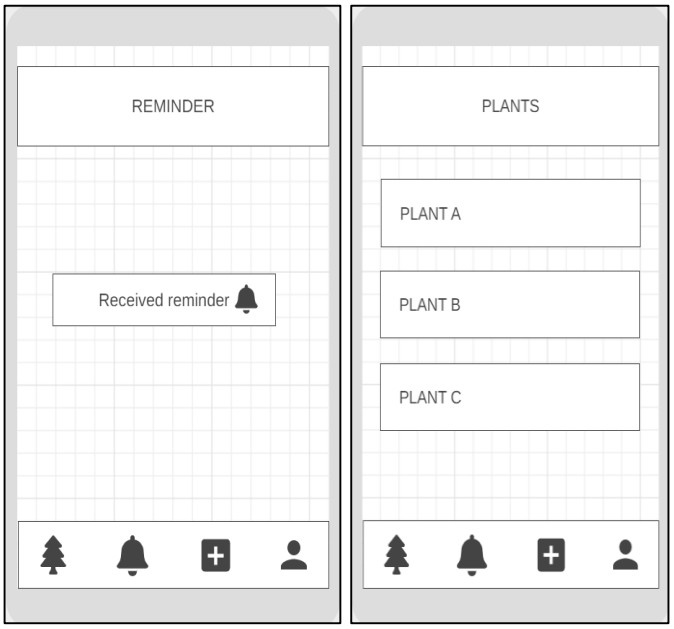

Figure 7: Reminder and dashboard page interface

4.3 Implementation

The PM mobile application is built using Flutter as a framework within the Android Studio IDE. The application utilizes Dart as the programming language for the mobile app itself, while JavaScript is employed for the Microbit integration. Firebase Cloud Firestore serves as the chosen storage platform. To gather data from the Microbit, the application utilizes the ThingSpeak platform, obtaining API keys from there to display the moisture chart. In Figure 8, the implementation of Firebase Cloud Firestore is demonstrated, specifically when a user registers into the application.

```
class DataUploader {
  FirebaseFirestore db = FirebaseFirestore.instance;

  void uploadProfile(List<ProfileForm> profileForm,
      {required Future<void> Function() onSuccess,
      required Future<void> Function() onFailed}) async {
    Map<String, String> data;

    if (profileForm.first.password.isEmpty) {
      data = {
        "name": profileForm.first.name,
        "phoneNumber": profileForm.first.phoneNumber
      };
    } else {
      final encryptPassword = encrypt(profileForm.first.password);
      data = {
        "name": profileForm.first.name,
        "phoneNumber": profileForm.first.phoneNumber,
        "password": encryptPassword
      };
    }

    final docRef = db.collection('account').doc(profileForm.first.email);
    await docRef.set(data, SetOptions(merge: true)).then((value) {
      onSuccess();
    }).onError((error, stackTrace) {
      onFailed();
    });
```

Figure 8: Code segment for data uploading to Firebase Cloud Firestore

The arrangement depicted below demonstrates how the Microbit is connected to a power source like a battery or computer. To send data to the cloud and measure the soil moisture of the plant, the Microbit board needs to be connected to the wifi module and the sensor. Figure 9 illustrates the setup that users should follow for this project.

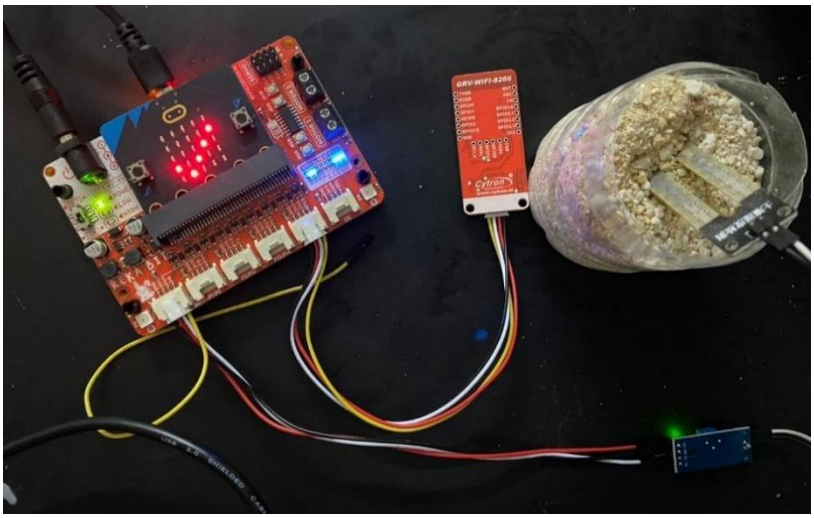

Figure 9: Code segment for data uploading to Firebase Cloud Firestore

Next, Figure 10 shows the code segment for the Microbit to send data to ThingSpeak, the Microbit device needs to declare the channel ID provided by ThingSpeak. This channel ID acts as a unique identifier for the specific channel where the data will be sent. By declaring the channel ID, the Microbit establishes a connection with ThingSpeak and becomes capable of transmitting data.

```
basic.forever(function on_forever() {

    basic.pause(1000)
    moisture = 1023 - pins.analogReadPin(AnalogPin.P0)
    basic.pause(1000)
    basic.showString("" + ("" + moisture))
    esp8266.uploadThingspeak("DTGIG5DU85FDFQ77", moisture)
    basic.showIcon(IconNames.Yes)
    serial.writeLine("" + ("" + moisture))
})
```

Figure 10: Code segment for sending data to ThingSpeak

Once the moisture level drops below a predefined threshold, a notification will be triggered and delivered through the notification bar as a reminder. This behavior visually illustrates the representation of the reminder notification. In Figure 11, code snippets are showcased, demonstrating the implementation of the logic for receiving and managing reminders within the codebase of the mobile application, specifically from ThingSpeak. These code snippets serve as a practical reference, aiding in the seamless integration of reminder functionality into the application.

```
void checkNotify() {
  _timer = Timer.periodic(const Duration(minutes: 1), (_) {
    final isEnable = prefs.getBool('notification') ?? false;
    if (isEnable) {
      debugPrint("NOTIFICATION ENABLED");

      for (var i = 0; i < plantForm.length; i++) {
        thingSpeak = fetchThingspeak(plantForm[i].fieldApi);
        thingSpeak.then((value) {
          debugPrint(value.field1);
          if (value.field1.isNotEmpty) {
            debugPrint("IS NOT EMPTY");
            if (int.parse(value.field1) < 800 && !_notify) {
              debugPrint("NOTIFY");
              notify(plantForm[i].name, value.field1);
              setState(() {
                _notify = true;
                _lastNotify += 1;
              });
            }
            if (_lastNotify > 30) {
              _notify = false;
              _lastNotify = 0;
            }
          }
        });
      }
```

Figure 11: Code segment for receiving reminder process

Furthermore, by inputting the correct APIs from the ThingSpeak platform as depicted in Figure 12 the user can access the moisture chart within this function. The chart has been optimized to fit various smartphone screens, ensuring clear visibility for the user. Additionally, users have the option to interact with the chart by clicking on it to obtain detailed information regarding the moisture levels.

```
child: Column(
  crossAxisAlignment: CrossAxisAlignment.start,
  children: [
    Text("${plantForm[_selected].name} - Moisture Chart", style: const TextStyle(fontSize: 20, fontWeight: FontWeight.bold),),
    SizedBox(
        height: MediaQuery.of(context).size.height / 2,
        child: WebViewWidget(controller: controller)), // SizedBox
    SizedBox(height: 10,),
    SizedBox(
      width: MediaQuery.of(context).size.width,
      child: FilledButton.tonal(
          onPressed: () {
            setState(() {
              showChart = false;
            });
          },
          child: const Text(backButton)), // FilledButton.tonal
    ), // SizedBox
```

Figure 12: Code segment for viewing process

The registration process for the application is depicted in Figure 13. Users are required to provide their email, phone number, and password during registration. Only registered users have the ability to log into the application.

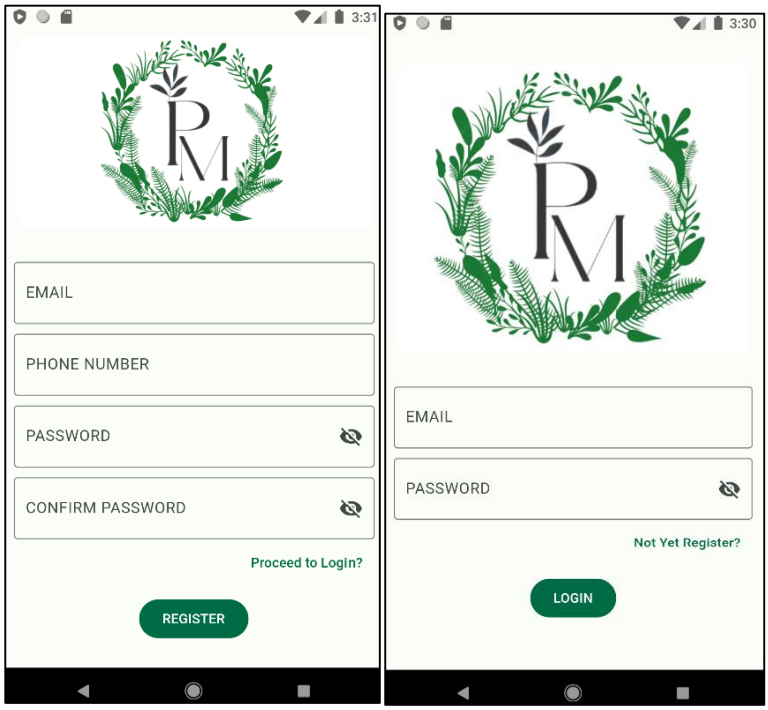

Figure 13: Register and Login interface for PM mobile application

Afterward, the user has the option to input the plant's name, category, and description. To access the chart API and field API, the user can retrieve them from the ThingSpeak page. The user needs to complete all the required forms; otherwise, they will not be able to add the plant to the database. Once the user has finished adding a plant, they can click the home button on the bottom navigation to view a list of all the plants they have created. On the homepage, the plant named 'Salad' will be displayed. Clicking on a specific plant will redirect the user to a dedicated page showing all the plant's details. Figure 14 provides a visual representation of these actions and their corresponding functionalities.

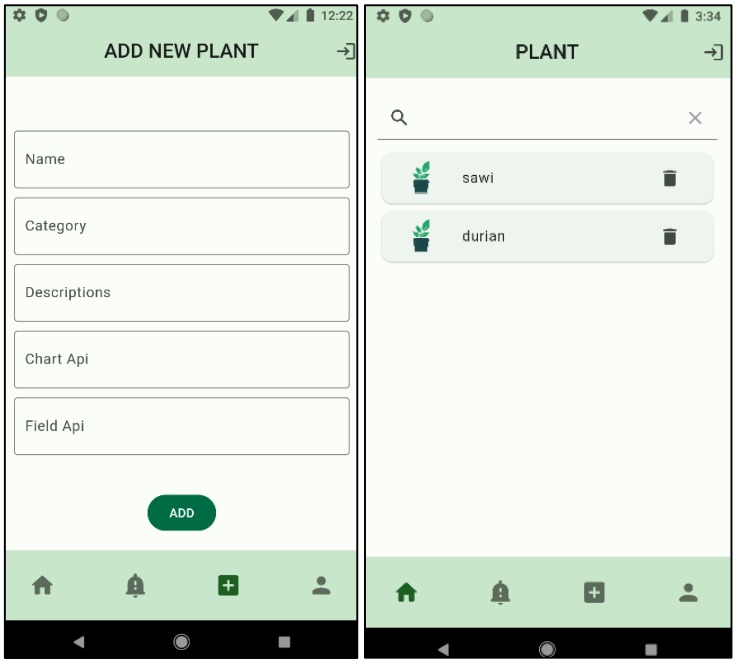

Figure 14: Add New Plant and Home interface for PM mobile application

By inputting the correct APIs from the ThingSpeak platform as depicted in Figure 15 the user can access the moisture chart within this function. The chart has been optimized to fit various smartphone screens, ensuring clear visibility for the user. Additionally, users have the option to interact with the chart by clicking on it to obtain detailed information regarding the moisture levels.

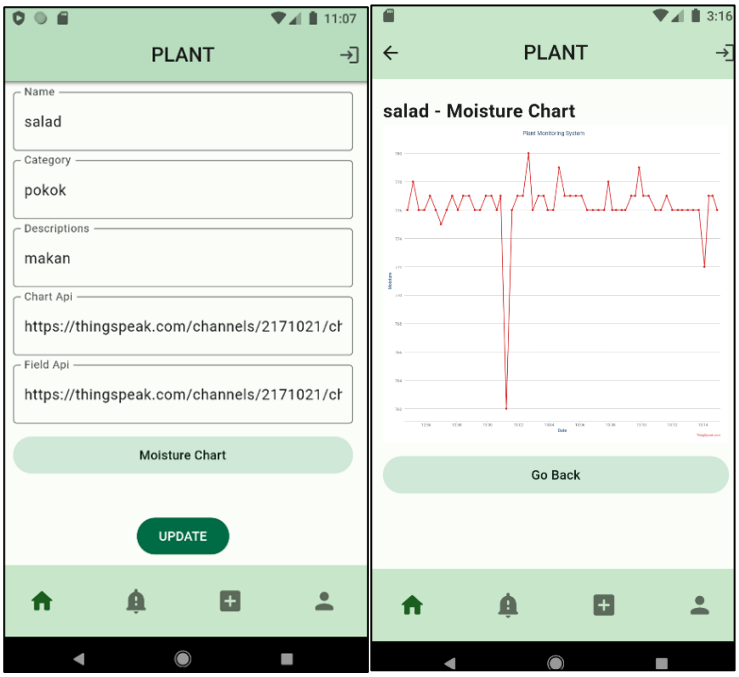

Figure 15: Plant details with its moisture chart interface for PM mobile application

Figure 16 showcases the user interface for the reminder function. Users can enable or disable the reminder based on their preferences. When the reminder is disabled, it is visually indicated by a red bar. Conversely, a green bar represents an enabled reminder, ready to receive notifications. Notifications will be displayed in the notification bar when the water level falls below 800.

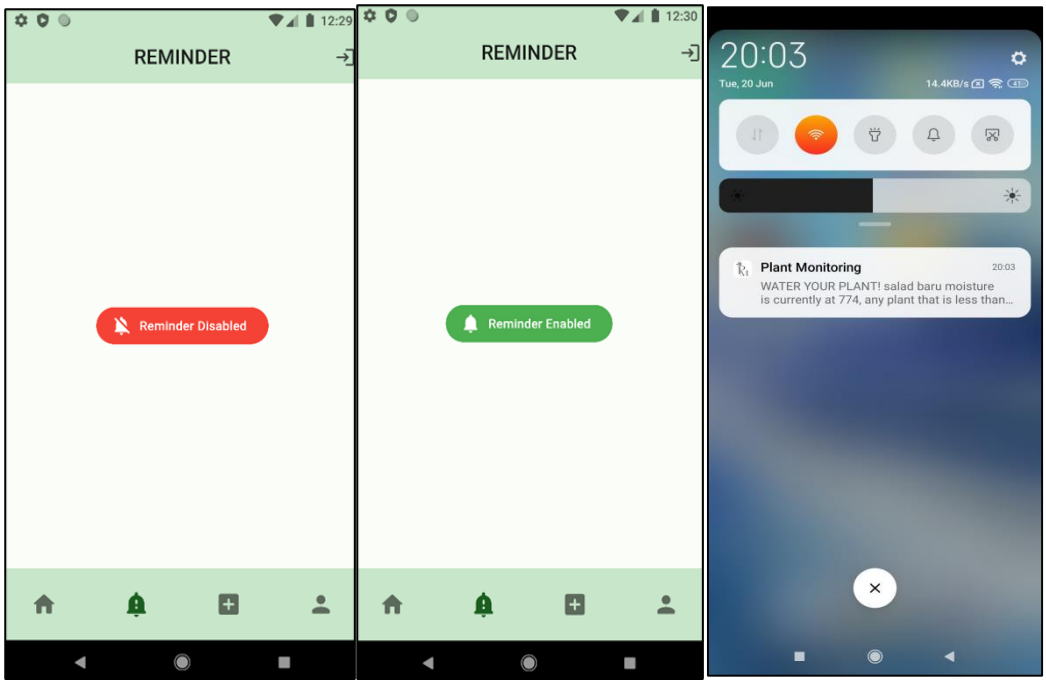

Figure 16: Reminder interface for PM mobile application

4.4 Application Testing

4.4.1 Functional Testing

This form of testing encompasses a thorough examination of various elements, such as application interfaces, functions, Microbit sensors, the connection between Microbit and the application, APIs, database functionality, and other relevant features. By scrutinizing these aspects, functional testing ensures that the system performs as expected, meets the specified requirements, and functions seamlessly across its various components. Table 7 and 8 shows the test plan conducted for the proposed application.

Table 7: Test plan of the application module and results

Module	Functions	Test case	Expected output	Actual output
User management	Registration	Incomplete data input	An alert message will display if the field is empty.	Pass
		Unique email	Display cross symbol which means the registration is failed.	Pass
		Password based on requirements	Display cross symbol which means the registration is failed.	Pass
		Complete registration form	Display right symbol which means the registration is a success and redirect to home page.	Pass
	Login	Incomplete data input	An alert message will display if the field is empty.	Pass
		Complete input with invalid email or password	Display cross symbol which means the login is failed.	Pass
		Complete form	Display right symbol which means the registration is a success and redirect to home page.	Pass

Table 8: (cont.)

Module	Functions	Test case	Expected output	Actual output
	User profile	Display the user profile data as registered	The profile page will show the username, email, and phone number.	Pass
		Update user profile with valid input	The user can update their username, phone number and password.	Pass
	Logout	Press logout button	A dialog box will appear for confirmation	Pass
Plant management	Add plant	Incomplete data input	An alert message will display if the field is empty.	Pass
		Complete add plant form	Display right symbol which means the registration is a success and redirect to home page.	Pass
	View plant	Display the plant details as set	The plant page will show the plant's name, category, descriptions, chart API, field API and moisture chart.	Pass
		Update user profile with valid input	The user can update the plant's name, category, descriptions, chart API and field API.	Pass
		Enter wrong chart API and field API	The moisture chart will appear as blank.	Pass
	ThingSpeak	Send data to application	The moisture chart will appear	Pass
	Delete plant	Press delete button	A dialog box will appear for confirmation	Pass
		Choose to delete	Display home page with the new plant list	Pass
	Search	Search plant	The list will show plant's list according to what the user search.	Pass
		Press the clear button	The search bar will be empty and the list will be back to normal	Pass
Reminder	Enable and disable button	Press enable button	The button will display the disable button.	Pass
		Press enable button	The button will display the disable button.	Pass
	Receive reminder	Display reminder	Receive reminder on notification bar of the user's phone	Pass
Microbit	Soil moisture sensor	Sense soil moisture	The LED on the Microbit will shows the	Pass
		Press enable button	The button will display the disable button.	Pass
		Press disable button	The button will display the enable button.	Pass
	Send data to ThingSpeak	Display plant's soil moisture on ThingSpeak	The moisture chart will state its moisture at the moment.	Pass

4.4.2 User Acceptance Testing

User acceptance testing involves gathering feedback and opinions from real users to evaluate their thoughts on the project. Considering the time limitations, only the 10 closest users were involved in this testing phase. The collected data was then assessed and presented in the form of a graph. Based on Figure 17 and Figure 18, shows the data collected from the user.

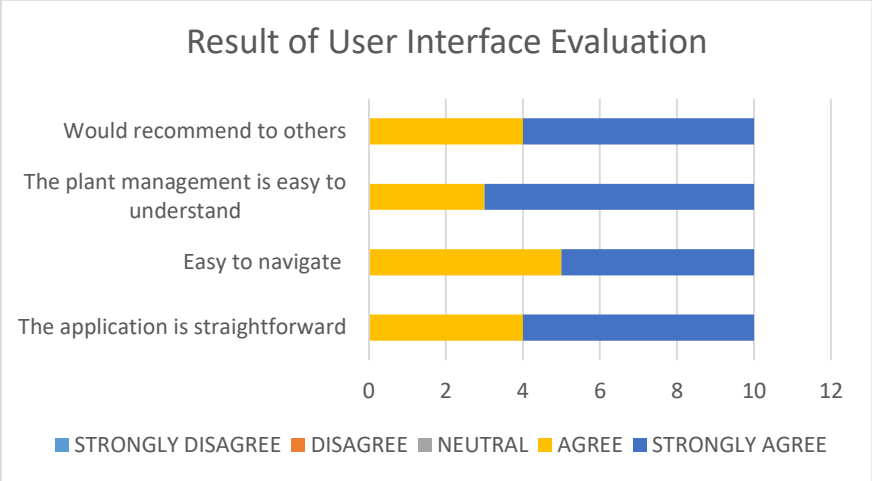

Figure 17: Result of User Interface Evaluation

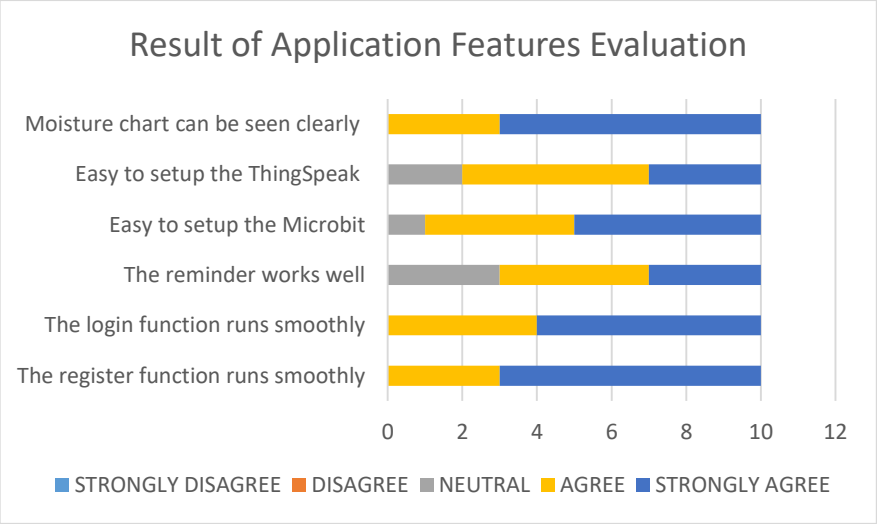

Figure 18: Result of Application Features Evaluation

Based on the analysis of the data presented in Figure 17 and Figure 18, it can be concluded that the majority of users expressed satisfaction with the user interface of the application. They found it to be easily navigable, comprehensible, and straightforward also showed a willingness to recommend it to others. No users reported dissatisfaction or significant dissatisfaction with the design of the interface or the functionality of the proposed application. Regarding the features, most of them worked effectively and smoothly. However, there is room for improvement in the reminder feature to ensure timely reminders. Initially, setting up ThingSpeak and Microbit may require some time and understanding, but it becomes relatively easy with time.

5. Conclusion

The project focuses on using an IoT device to collect soil moisture data and transmit it to the ThingSpeak platform. An application is developed to display moisture charts based on the collected data, providing users with visual insights. Included in the application is a feature that sends reminders

to users when the sensor detects a moisture level below 800, indicating the requirement for watering. The project successfully achieves its objective of aiding individuals in arid regions to monitor their plants' water requirements. However, there are limitations to address, such as the absence of comprehensive plant monitoring features, the lack of picture integration, occasional delays in reminders, and opportunities to improve the user interface. Future development can expand monitoring capabilities, integrate image support, enhance the reminder system, consider smart irrigation integration, and refine the user experience. These improvements will further enhance the application's functionality, usability, and effectiveness in plant care.

Acknowledgment

The authors would like to thank the Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia for its support throughout the process of conducting this study.

Penulis ingin mengucapkan terima kasih kepada Fakulti Sains Komputer dan Teknologi Maklumat, Universiti Tun Hussein Onn Malaysia atas sokongan dan dorongan sepanjang proses menjalankan tugas ini.

References

[1] S. Vaishali, S. Suraj, G. Vignesh, S. Dhivya and S. Udhayakumar, "Mobile integrated smart irrigation management and monitoring system using IOT," 2017, International Conference on Communication and Signal Processing (ICCSP), April 2017, pp. 2164-2167.

[2] K. Kodali, B. S. Sarjerao, "A low-cost smart irrigation system using MQTT protocol", 2017 IEEE Region 10 Symposium (TENSYMP), Cochin, India. July 2017, pp. 1-5.

[3] Huang, Y., Girdhar, Y., & Baldocchi, D. D, "A conceptual model of a deep learning network for estimating gross primary production of maize and soybean crops with fused multi-sensor remote sensing data.", Agricultural and Forest Meteorology, issue 264, pp. 188-201, 264, 2019.

[4] Chen, J. M., & Cihlar, J., "Plant canopy gap-size analysis theory for improving optical measurements of leaf-area index", Applied Optics, vol. 34, issue 27, pp. 6211-6222, 1995, doi: https://doi.org/10.1364/AO.34.006211

[5] Wahabzada, M., Mahlein, A. K., Bauckhage, C., & Kersting, K., "Plant phenotyping using probabilistic topic models: Uncovering the hyperspectral language of plants. Scientific Reports", no. 5, 15786.1025, 2016, doi: https://doi.org/10.1038/srep22482.

[6] Schirrmann, M., & Rao, S., "Wireless sensor networks for soil moisture monitoring in precision agriculture: A review. Computers and Electronics in Agriculture". no. 165, 104964, 2019, doi: https://doi.org/10.3390/s21217243

[7] (2021). Plantnote: Plant Diary & Water [Mobile App]. Retrieved from Google Play Store. https://play.google.com/store/apps/details?id=com.dyhwang.plantnote&hl=en&gl=US

[8] (2020). Blossom: Plant Identifier [Mobile App]. Retrieved from Google Play Store. https://play.google.com/store/apps/details?id=com.conceptivapps.blossom&hl=en&gl=USJ

[9] (2019). Plant Parent: Plant Care Guide [Mobile App]. Retrieved from Google Play Store. https://play.google.com/store/apps/details?id=com.plantparentai.app&hl=en&gl=US

[10] Pickett, & Joseph. The American Heritage Dictionary of the English language, 4th ed., p. 2074. Publisher: Houghton Mifflin. 2000. [Ebook] Available: Google Books.

[11] Hyman, B. I. Fundamentals of Engineering Design. Prentice Hall/Pearson Education. 2003. [Ebook] Available: Google Books.

[12] Dieter, G. E. (n.d.). Engineering design: A materials and processing approach. 2022. [E-book] Available: Google Books.

[13] RanaKuldeep , K., “Prototype model - phases, types, Advantages & Disadvantages”, April 29, 2021. ArtOfTesting. [Accessed December 4, 2022], Available: https://artoftesting.com/prototype-model