APPLIED INFORMATION TECHNOLOGY AND COMPUTER SCIENCE
e-ISSN: 2773-5141

AITCS

Vol. 6 No. 2 (2025) 190-207
https://publisher.uthm.edu.my/periodicals/index.php/aitcs

Smart Irrigation Solutions: An IoT-Based Mobile Application for Optimized Watering in Modern Agriculture

Ng Wei Kang [1], Rosmamalmi Mat Nawi[1]*

[1] *Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, 86400 Parit Raja, Batu Pahat, Johor, Malaysia*

*Corresponding Author: rosmamalmi@uthm.edu.my
DOI: https://doi.org/10.30880/aitcs.2025.06.02.011

Article Info

Received: 11 June 2025
Accepted: 3 November 2025
Available online: 30 November 2025

Keywords

Internet of Things, Smart Irrigation, Sensor, Mobile Application, Modern Agriculture

Abstract

The Smart Irrigation Solutions is an IoT-based application developed using Android Studio, Arduino IDE, and Firebase. It aims to automates irrigation by integrating environmental sensors and weather data to reduce manual effort and prevent over- or under-irrigation. The system features seven core modules including User Management for account handling, Device Management for IoT integration, Environmental Monitoring & Irrigation Control for real-time data and automation irrigation, Weather Integration for forecast-based decisions, Plant Management for plant profiles, Notifications, and Reporting & Logs for historical analysis. Real-time monitoring and automated responses are enabled through sensor integration, while Firebase handles authentication and cloud data storage. The application also uses OpenWeatherAPI for accurate forecasting. This final version of the project demonstrates successful outcomes, including optimized water usage, increased agricultural efficiency, and reduced labor dependency. It promotes sustainable farming by ensuring crops receive the ideal amount of water, supporting both productivity and environmental conservation.

1. Introduction

The Smart Irrigation Solutions utilizes Internet of Things (IoT) technology to optimize irrigation by monitoring soil moisture and weather conditions in real-time [1]. Water management is paramount in countries with water scarcity, as agriculture dedicates a large amount of water to irrigation purposes [2]. By integrating sensors and data analytics, it automates watering to deliver the exact amount of water each plant needs, improving efficiency and reducing water consumption [3]. Automated irrigation systems are essential for conservation of water, and this improvement could have a vital role in minimizing water usage [4]. Traditional irrigation relies on fixed schedules and manual labour, often resulting in over or under-irrigation that harms crop and wastes resources [5]. These methods do not consider real-time environmental changes or individual plant needs [3]. Our proposed project addresses this issue by dynamically adjusting irrigation based on sensor data and weather forecasts, thereby enhancing water management and crop health for small gardens and farms. Results show significant improvement in irrigation efficiency, with more accurate use of water resources based on actual soil and atmospheric conditions [6]. The proposed application is designed especially for home gardens, targeting potted plants, ensuring precise irrigation tailored to each plant's requirements. This approach helps prevent both over and under-irrigation, promoting healthier plants and reducing water consumption [1].

2. Literature Review

This section will discuss about the study been made and the comparison between existing systems.

2.1 Domain Background

The global population is projected to reach 9.7 billion by 2050, increasing food demand, while the agricultural workforce declines, dropping from 44% in 1991 to 26.09% in 2023, threatening food security [7], [8]. Urbanization in Southeast Asia is converting farmland into urban areas, with waste volume projected to double from 150 million tonnes in 2016 to over 300 million tonnes by 2030, further straining land use [9]. The Asia-Pacific waste management market is expected to grow from USD 118.53 billion in 2025 to USD 158.92 billion by 2030, reflecting rapid urban expansion [10]. These trends challenge agricultural sustainability and food production as land availability decreases.

2.2 IoT in Agriculture Sector

Internet of Things (IoT) technology is revolutionizing farming in Southeast Asia, addressing challenges like climate change, soil degradation, and water shortage in countries such as Indonesia, Malaysia, and Thailand. Soil moisture sensors enable precise irrigation, conserving water and maintaining crop health during droughts, while IoT-powered weather stations provide localized forecasts and early warnings for extreme weather, protecting crops from floods and heatwaves [11], [12]. Amid urbanization-driven farmland loss, IoT systems optimize water use, monitor soil health, and track weather changes, helping small-scale farmers maximize yields. For instance, IoT-based irrigation reduces water waste by up to 45% while enhancing crop quality [13]. These technologies are vital for sustainable food production as population growth and urban expansion strain agricultural resources [14].

2.3 Comparison Between Existing System

There is a total of three existing systems and having comparison with our proposed application. Three existing systems which are Rachio 3 Smart Sprinkle Controller [15], Orbit B-Hyve XR [16], and Hunter Hydrawise [17]. The three existing systems offer app control, sensor compatibility, multi-controller support, and weather integration, but none provide water adjustment for plants. Our proposed system introduces this feature by adjusting irrigation levels based on each plant's unique moisture requirements. Table 1 shows the comparison features between three existing systems and our proposed application.

Table 1 System Comparison

No	Features	Rachio 3 Smart Sprinkler Controller	Orbit B-Hyve XR	Hunter Hydrawise	Smart Irrigation Solution Application
1	Smartphone App Control	Yes	Yes	Yes	Yes
2	Adjust and recommended water needed for plants	No	No	No	Yes
3	Activate or stop the irrigation system through app	Yes	Yes	Yes	Yes
4	Sensor Compatibility	Yes	Yes	Yes	Yes
5	Multi-controller support	Yes	Yes	Yes	Yes
6	Weather-Data Integration	Yes	Yes	Yes	Yes
7	Installation Difficulty	Moderate	Moderate	High	Low
8	Cost	RM851	RM723	RM1,064++	RM50

3. Methodology

The Smart Irrigation Solution Application is developed using the prototype model. It is a software development approach that creates a prototype to gather feedback and refine system requirements before full development begins [18]. It involves four phases including planning, analysis, design, and implementation phase. The prototype model repeated in cycles to improve the prototype until requirements are met. Once the prototype meets the desired criteria, the final product will be developed.

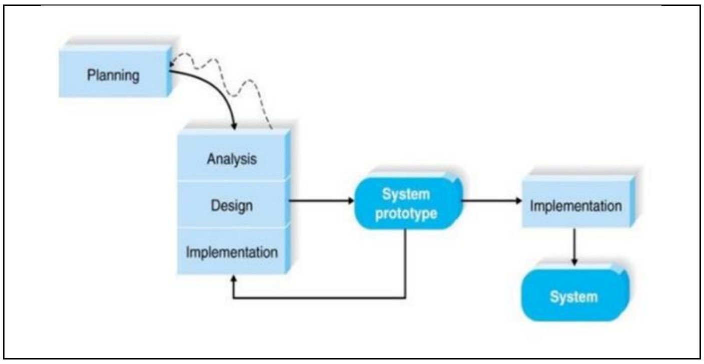


Fig. 1 Prototyping Model [19]

The planning phase involves outlining the project goals, scope, timeline, and resources needed to ensure smooth development. During the analysis phase, requirements are gathered and studied through stakeholder interviews and a literature review to understand existing solutions and industry trends. In the design phase, a detailed wireframe is developed as a blueprint for the application's interface, reviewed by stakeholders to address feedback and potential design issues. UML diagrams and database designs are also prepared to define system functionality and structure. Finally, the implementation phase begins with connecting and testing sensors via the Arduino IDE. Once verified, the development of a Flutter-based mobile application is initiated, ensuring successful integration with Firebase and the weather API. The IoT devices send data to Firebase, which the mobile application retrieves and displays for user monitoring.

Table 2 will show the hardware requirements for the IoT device and Table 3 will show the hardware requirements for laptop. Table 4 will show the software requirements.

Table 2 Hardware Requirement (IoT Device)

No	Hardware	Specification
1	Moisture Sensor	YL-69 Hygrometer w/ HC-38 Module Board LM393
2	Arduino	ESP 32
3	Relay	Relay-5V-4 Ways
4	Water Pump	DC 5V Mini Submersible Water Pump
5	Battery	GP 9V Battery

Table 3 Hardware Requirement (Laptop)

No	Hardware	Specification
1	Laptop Model	Acer Nitro 5
2	Central Processing Unit	Intel Core I i5-8300H CPU @ 2.30GHz 2.30GHz
3	Graphic Processing Unit	Nvidia GeForce GTX 1050
4	Random Access Memory	12 GB
5	Solid State Drive	1.5 TB

Table 4 Software Requirement

No	Hardware	Specification
1	Arduino IDE	To code and compile the coding for Arduino.
2	Visual Studio Code	To code for Mobile Application.
3	Firebase	To store and retrieve data.
4	Android Studio	To test and run as virtual device.

3.1 Application Development Workflow

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 190-207

The application development workflow outlines all the tasks performed during each phase of development. The prototyping model consists of four key phases: Planning, Analysis, Design, and Implementation.

The project follows four main phases, each with their key outputs. The Planning Phase defines the project background, objectives, scope, timeline, and stakeholders, resulting in a project proposal and Gantt chart. The Analysis Phase involves researching existing systems, interviewing stakeholders, and identifying hardware, software, and system requirements, resulting a literature review and detailed requirements. In the Design Phase, wireframes, database schemas, and UML diagrams are created to plan the system's interface and structure. Finally, the Implementation Phase sets up IoT components, develops the mobile app with Firebase integration, and performs testing and debugging, delivering the completed Smart Irrigation Solution Application.

4. Analysis and Design of Smart Irrigation Solution

This section explains the analysis and design of the proposed application.

4.1 System Architecture

System architecture defines the components of a system, their characteristics, and interconnections, forming the foundation for achieving key quality attributes, with performance often being the primary consideration [20]. It illustrates the interactions among the user, mobile application, IoT devices, databases, and external APIs, explain the flow of data in the proposed application. Figure 2 illustrates the system architecture diagram and explains the high-level structure of the application.

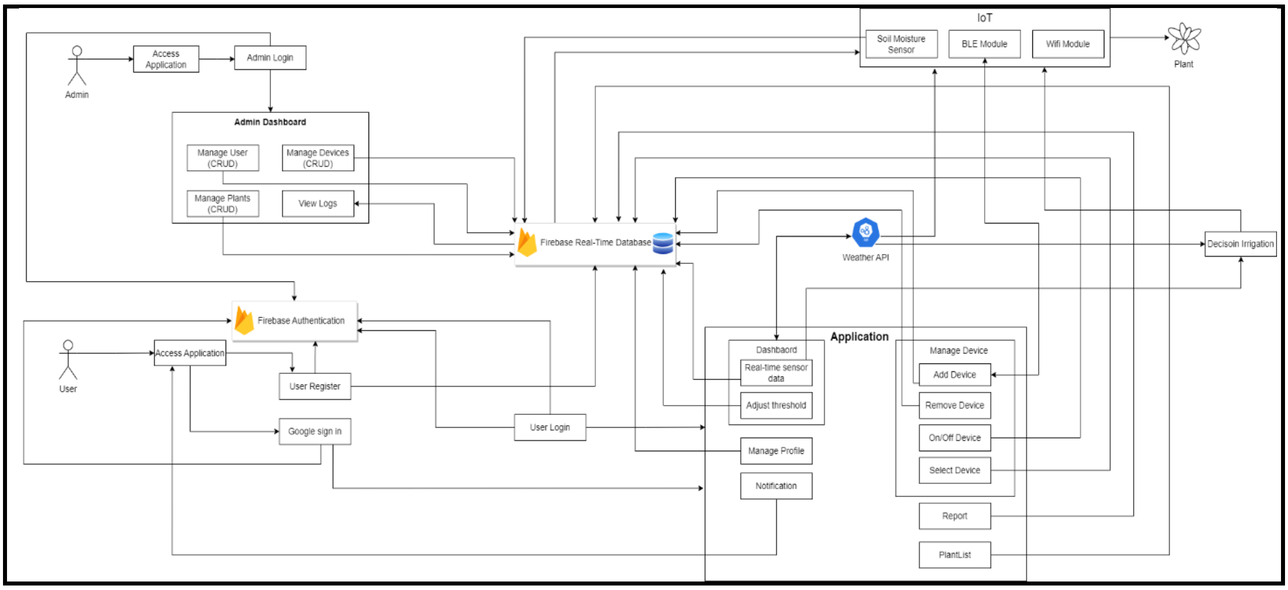


Fig. 2 System Architecture Diagram

4.2 Requirement Analysis

Requirement analysis is a critical step in ensuring the success of any system or software project. Requirements are divided into two types: functional and non-functional [21]. Functional requirements define the tasks the system must perform. They focus on the system's core features such as automatic irrigation where the application capable to study both sensor readings and weather forecasts to controls the irrigation pump automatically, ensure an efficient and ideal water usage. Non-functional requirements focus on how the system operates, covering aspects like operation, performance, security, and usability. Table 5 and Table 6 will display the user and admin functional requirements of the proposed application, while Table 7 will explain about the non-functional requirements for the proposed application.

Table 5 User Functional Requirements

`	Functions	Functionality
1	Register	• Users can create an account, verify via email, use Google Sign-In, and are prevented from duplicate email registrations.
2	Login	• Users log in with Firebase-stored credentials or Google Sign-In, with password reset via email and secure logout.
3	Manage Profile	• Users can view/edit profile details, with changes saved to Firebase Real-time Database.
4	Add Device	• User's pair IoT devices via BLE, input WiFi credentials, assign unique names, and save details to Firebase, with error handling for unsupported/duplicates.
5	Remove Device	• Users can remove IoT devices, clear WiFi credentials via Firebase, and delete device data from Firebase.
6	Select Device	• Users select registered IoT devices for monitoring and control.
7	Weather Integration	• Fetches and displays weather data (temperature, humidity, conditions) from OpenWeatherAPI to inform irrigation decisions.
8	Monitor Real-Life Sensor Data	• Displays real-time soil moisture data from IoT devices via Firebase, with automatic updates.
9	Automatic Irrigation	• Triggers irrigation based on user-defined soil moisture thresholds soil moisture level and weather data, controlling the pump for efficient water use.
10	Plant List	• Displays plant details and updates IoT device thresholds based on selected plants.
11	Adjust Threshold	• Users manually set soil moisture thresholds, synced to Firebase and IoT devices.
12	Notifications Alert	• Sends push notifications for low moisture, device disconnection, or irrigation events.
13	Reporting	• The application should allow users to generate reports based on historical sensor data.

Table 6 Admin Functional Requirements

No	Functions	Functionality
1	Login	• Admins log in with unique Firebase-stored credentials, with role verification for admin-only access and error messages for incorrect credentials.
2	Manage User	• Admins can view, create, edit, and delete user accounts, with details managed in Firebase Real-time Database.
3	Manage IoT Devices	• Admins can view, edit, and remove IoT devices hold by the users.
4	Manage Plants	• Admins can view, add, edit, and delete plant saved in Firebase Real-time Database.
5	View Logs	• Admins can access system logs for user activities, device statuses, and irrigation events from Firebase Real-time Database.

Table 7 Non-Functional Requirements

No	Requirement	Description
1	Operation	• The application should operate reliably with a stable internet connection. • The application should maintain communication with IoT devices via Bluetooth Low Energy for pairing and WiFi for data transmission.
2	Performance	• The application should respond to user actions within 2-3 seconds under normal conditions • The application should handle real-time data from more than one IoT devices simultaneously without performance degradation, updating sensor data every 5 seconds. • The application should process weather data from OpenWeatherAPI and irrigation decisions within 3 seconds.
3	Security	• The application should restrict access to authorized users by requiring valid credentials (email/password or Google Sign-In) validated through Firebase Authentication.
4	Usability	• The application should provide an intuitive user interface and guide for users with limited technical expertise. • The application interface should display in a clear, accessible format on screens

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 190-207

4.3 UML Diagram

Unified Modelling Language (UML) is a used in software engineering to show how a system works and is structured. It works well with iterative development and includes four main diagrams: Use-case, Sequence, Activity, and Class diagrams.

4.3.1 Use-Case Diagram

A use case diagram is a UML diagram that visually shows how an application interacts with actors, highlighting the system's functions through specific actions or services it performs [22]. Figure 3 illustrates the use case diagram of the project.

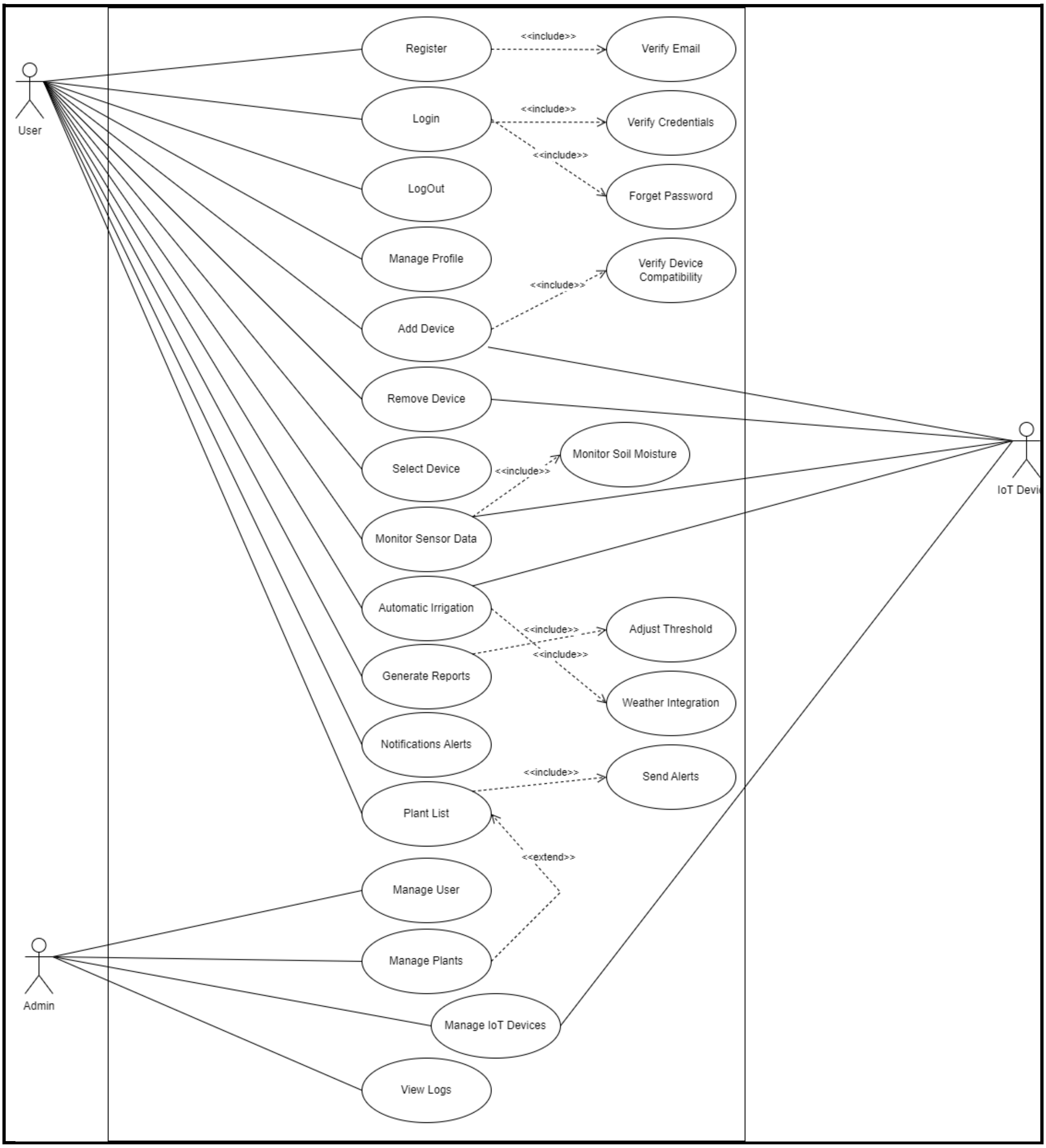


Fig. 3 Use Case Diagram

4.3.2 Activity Diagram

Activity diagram used to represent workflows, processes, or the flow of control in a system. It shows the sequence of activities or tasks involved in a process, showing how different actions are connect with each other.

For the proposed application, once user access the application, user will reach the login page first. If existing user, user will be able to login using existing account. Else if new user, user able to direct to the register page to register an account. In the dashboard page, user will able to monitor the real-life sensor data. User will also able to navigate to profile page, manage device page, plant list page and report page. In profile page, user able to manage or edit their profile. Manage device page will let user to connect or remove the IoT devices. In plant list page, user able to select the plant for best irrigation solutions. The report page will let user view the report. The user will able to logout from their account. Figure 4 illustrates the activity diagram of the proposed application.

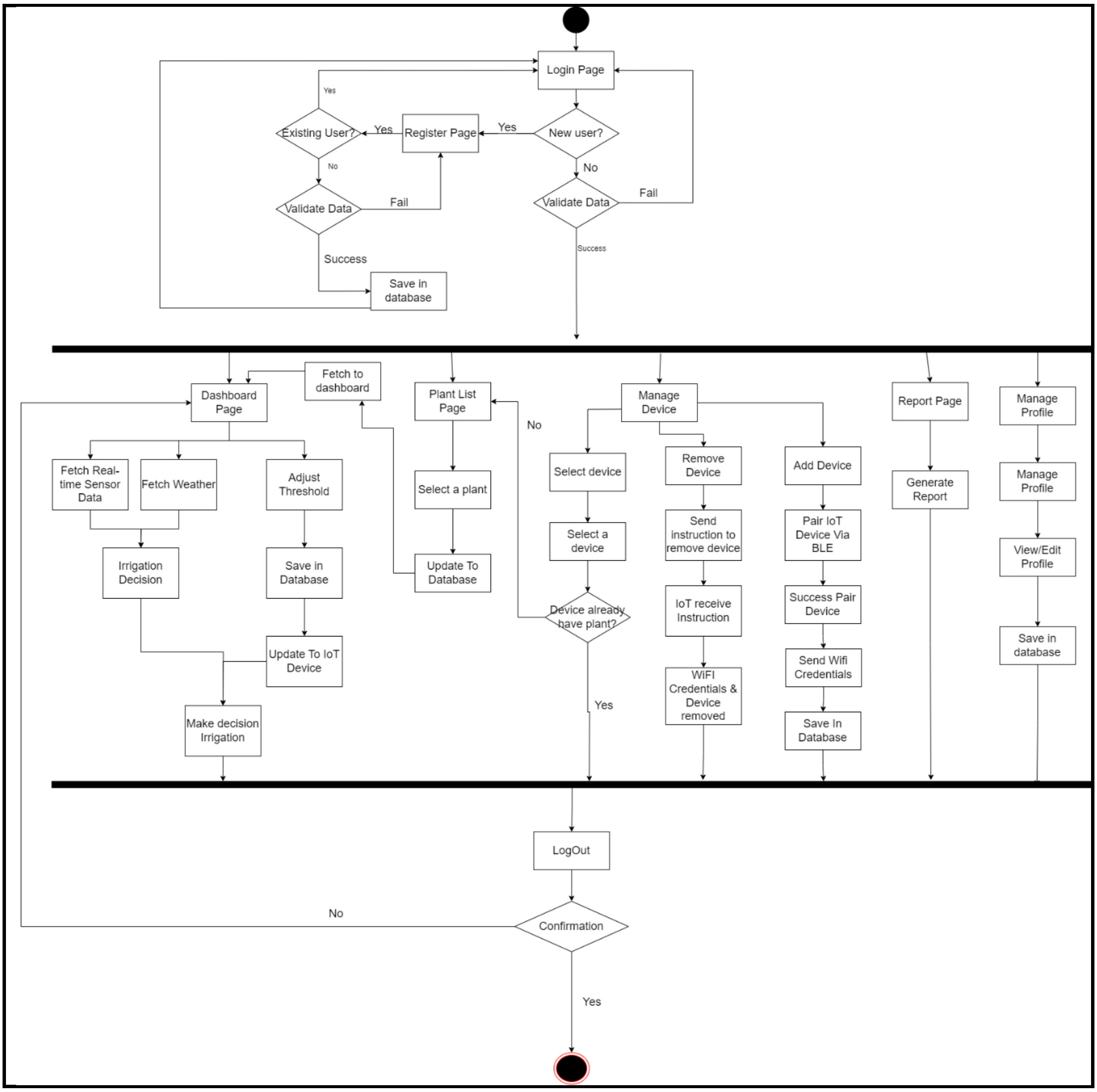


Fig. 4 Activity Diagram

4.3.3 Class Diagram

Class diagram is commonly used to represents the structure of a system by showing its classes, attributes, methods, and the relationships. The proposed application has total of eight classes, including User, Admin, IoT Device, Plant, log, Report, Weather and Notification. Each class requires different attributes and methods. Figure 5 illustrates the class diagram of the proposed application.

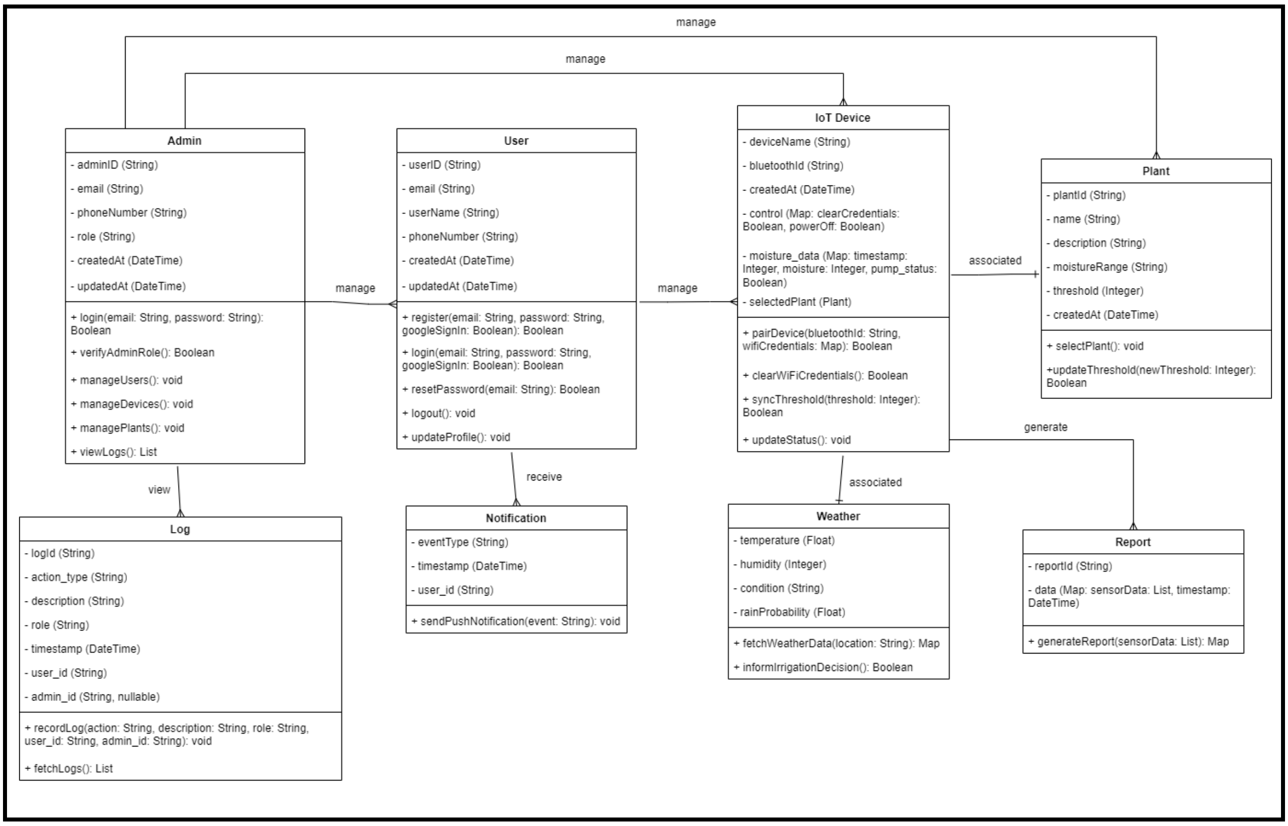


Fig. 5 Class Diagram

4.4 Prototype

Figure 6 shows the prototype of the IoT device for the Smart Irrigation Solutions Application. It uses an ESP32 microcontroller as the main component of the system for automatic watering. The device connects to a soil moisture sensor to monitor the soil moisture level, ensuring that plants receive the right amount of water. A relay is used to activate a small water pump for irrigation. This setup conserves water, prevents over- or under-watering, and helps maintain plant health.

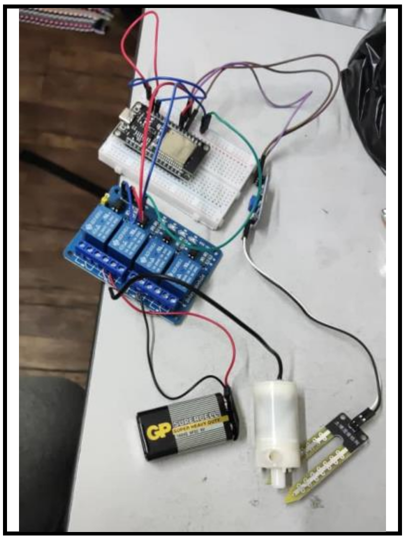


Fig. 6 IoT Device Prototype

5. Results and Discussion

The proposed project includes two main components which are a mobile application and IoT devices. The mobile application, built using the Flutter framework and Dart programming language, provides a user-friendly interface for managing irrigation and other tasks. Firebase serves as the backend-as-a-service, handling authentication, email notifications, and push notifications, while its real-time database securely stores essential data. The IoT devices, programmed with Arduino IDE using C++ language, ensure seamless operation of the system's modules, including soil moisture sensing, pump control, and data transmission. This integration enables efficient, automated irrigation tailored to plant needs, enhancing water conservation and crop health.

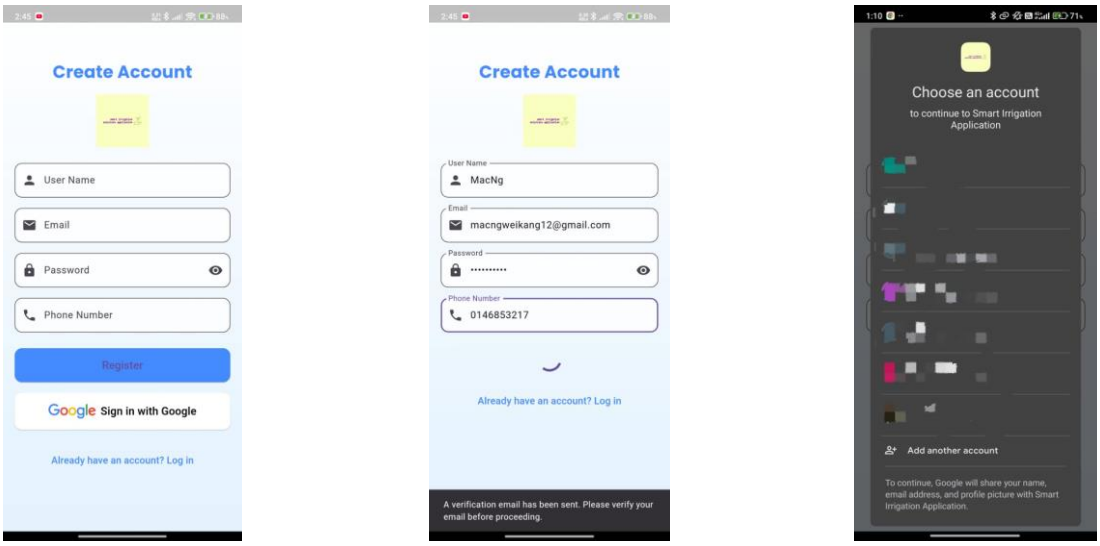
5.1 System Implementation

Figures 7 illustrate a proposed project prototype integrated into a potted plant setup. The system features an ESP32 microcontroller, a soil moisture sensor, a relay, and a mini water pump, with soil moisture sensor attached to the pot plants. The sensor monitors soil moisture levels, while the relay controls the pump to deliver precise water amounts. This setup enables efficient irrigation by tracking and reporting soil conditions, ensuring plant health and water consumption.

Fig. 7 IoT Device attached to pot plant

5.1.1 User Register

Users can create their own accounts to get started with the Smart Irrigation Solutions app. It's designed to be simple and secure, as users cannot sign up with the same email address more than once, ensuring each account is unique. After signing up, users receive a verification email to confirm their account and maintain security. For added convenience, users can also register quickly using Google Sign-In. Figure 8 shows the user register interface.



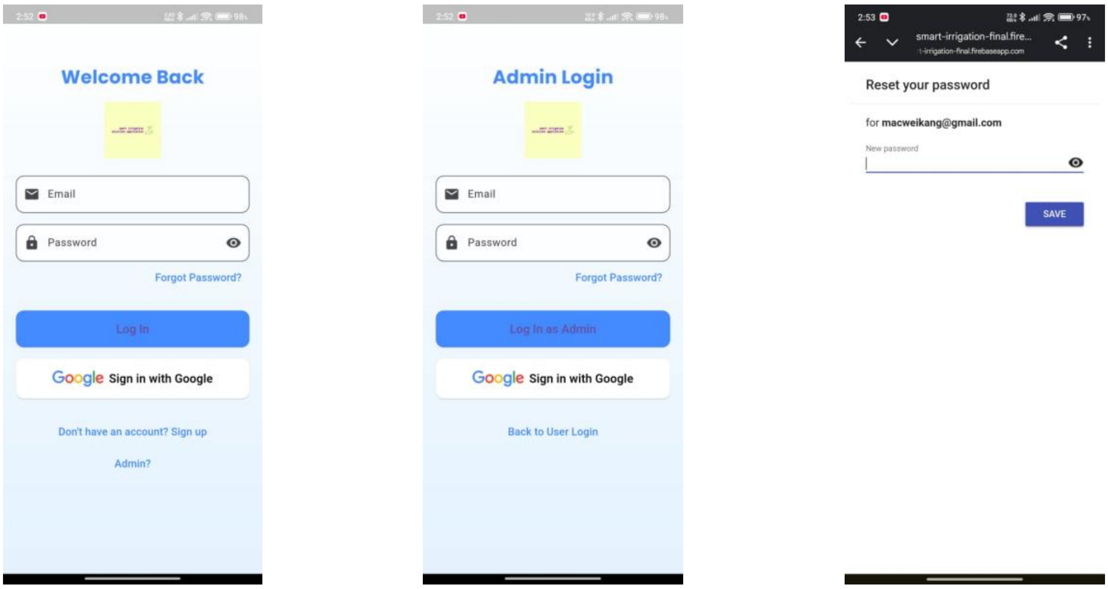
(a) (b) (c)

Fig. 8 (a) User Register Screen (b) Email Verification Send (c) Google Sign-In Function

5.1.2 User Login

Users can login into the Smart Irrigation Solutions app using an email and password stored securely in Firebase Authentication. If the credentials don't match, an error message pops up to guide the user. Users also can reset their password through the forgot password features as it will send an email link for password reset. For a quicker option, signing in with Google is available. Admins have a dedicated login using unique credentials, also stored in

Firebase. The system checks for admin status to unlock exclusive features, ensuring only authorized access. Incorrect admin credentials won't be able to login to admin page.

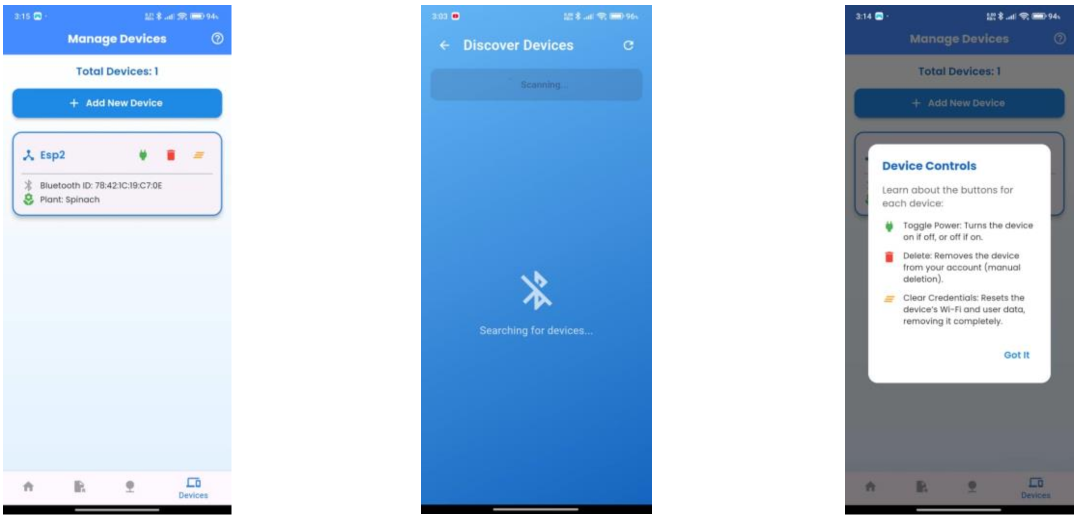


(a) (b) (c)

Fig. 9 (a) User Login Screen (b) Admin Login Screen (c) Reset Password Function

5.1.3 User Manage Devices

Users can effortlessly add, remove, and select IoT devices. Adding a new device happens through Bluetooth Low Energy (BLE) pairing, where WiFi credentials (SSID and password) and a unique device name are entered. To maintain system integrity, attempts to add unsupported or duplicate devices are blocked. Once added, device details are saved to the Firebase Real-time Database and shown on the Manage Devices page for easy access. Users can also delete a device from the app, triggering a command via Firebase to clear its stored WiFi credentials. Upon confirmation, the device is removed from the Firebase Real-time Database. For monitoring or controlling irrigation, users can select a registered device from a list of added devices, enabling seamless interaction with the IoT system. Users can also make instructions to turn on or off the IoT devices and make the IoT devices in deep sleep mode. In deep sleep mode, IoT devices will wake up every 30 seconds to check if the instructions made have changed. Figure 10 shows the user manage devices page while Figure 11 shows the algorithm of the manage devices page including the add device, select device and remove device.



(a) (b) (c)

Fig. 10 (a) Manage Device Screen (b) Bluetooth Connection Screen (c) Manage device Guide

Algorithm: User Manage Device

Flutter App:
Add Device:

1. Initialize controllers for SSID, password, and device name.
2. Validate input fields (SSID, password, device name) are non-empty.
3. Retrieve user ID from Firebase Auth.
    - Code:
```
final userId = FirebaseAuth.instance.currentUser?.uid;
if (userId == null) {
  throw Exception("User not authenticated");
}
```
4. Check if device name already exists in Firebase Realtime Database under user's devices.
5. Connect to ESP32 via BLE using FlutterReactiveBle with device ID and 10-second timeout.
6. On successful connection, send credentials (SSID:password:userId:deviceId) to ESP32's characteristic.
    - Code:
```
final characteristic = QualifiedCharacteristic(
    serviceId: Uuid.parse("4fafc201-1fb5-459e-8fcc-c5c9c331914b"),
    characteristicId: Uuid.parse("beb5483e-36e1-4688-b7f5-
ea07361b26a8"),
    deviceId: device.id,
  );
```
7. Save device metadata (name, Bluetooth ID, creation timestamp, control flags) to Firebase under user's devices.
    - Code:
```
await deviceRef.set({
  'deviceName': nameController.text,
  'bluetoothId': device.id,
  'createdAt': DateTime.now().toIso8601String(),
  'control': {'powerOff': false, 'clearCredentials': false},
});
```

Manage Device:

1. Initialize Firebase Auth, Realtime Database, and SharedPreferences.
2. Load Selected Device:
    - Retrieve 'selectedDeviceId' from SharedPreferences.

      Code:
```
Future<void> _loadSelectedDevice() async {
    final prefs = await SharedPreferences.getInstance();
    setState(() {
      selectedDeviceId = prefs.getString('selectedDeviceId');
    });
  }
```
3. Fetch Devices:
    - Get user ID from Firebase Auth.
    - Query Realtime Database for user's devices.
    - For each device, fetch control data (powerOff, clearCredentials).

      Code:
```
bool isPoweredOff = false;
        bool isClearRequested = false;
        if (controlSnapshot.exists && controlSnapshot.value is Map) {
          final controlData = controlSnapshot.value as Map;
          isPoweredOff = controlData['powerOff'] ?? false;
          isClearRequested = controlData['clearCredentials'] ?? false;
        }
```
    - Store device details (ID, name, Bluetooth ID, creation time, plant, power state, clear request) in a list.
4. Toggle Power Command:
    - Validate user authentication.
    - Update 'powerOff' in Realtime Database for the device.

      Code:
```
setState(() {
        final device = devices.firstWhere((device) =>
device['deviceId'] == deviceId);
        device['isPoweredOff'] = newPowerState;
      });
```
    - Update local device list and display success/error message.
5. Delete Device:
    - Remove device from Realtime Database.
    - Update local device list and clear selected device if deleted.
    - Display success/error message.
6. Clear Credentials Command:
    - Validate user authentication.
    - Set 'clearCredentials' to true in Realtime Database.
    - Update local device list and display success/error message.

      Code:
```
setState(() {
        devices.firstWhere((device) => device['deviceId'] ==
deviceId)['isClearRequested'] = true;
      });
```
7. Select Device:
    - Save device ID and name to SharedPreferences.
    - Update selectedDeviceId and navigate to dashboard.

      Code:
```
setState(() {
      selectedDeviceId = deviceId;
    });
```

Fig. 11 User Manage Devices Algorithm

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 190-207

5.1.4 Plant List

List of plants is displayed, showcasing details like name, description, recommended moisture range, and threshold. Selecting a plant from this list automatically updates the soil moisture threshold for the connected IoT device, ensuring irrigation aligns with the plant's specific needs. Figures 12 shows the plant list page.

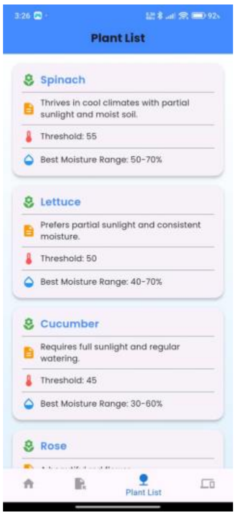


Fig. 12 Plant List Screen

5.1.5 Environmental Monitoring and Irrigation Control

The Dashboard integrates weather data from OpenWeatherAPI, displaying temperature, humidity, and conditions to guide irrigation decisions. Real-time soil moisture data retrieved from IoT devices, stored in the Firebase Real-time Database, is shown and updated automatically as new readings arrive, ensuring accurate monitoring. Automatic irrigation activates when soil moisture falls below a threshold, with the IoT device controlling the pump for efficient water use. Weather data further refines this process by pausing irrigation during rain or cloudy conditions. Users can also manually adjust the soil moisture threshold for each IoT device based on preferences, with changes threshold saved to the Firebase Real-time Database and synced with the IoT device. Figure 13 shows the dashboard page of the application while Figure 14 shows the algorithm of the environmental monitoring and irrigation control.

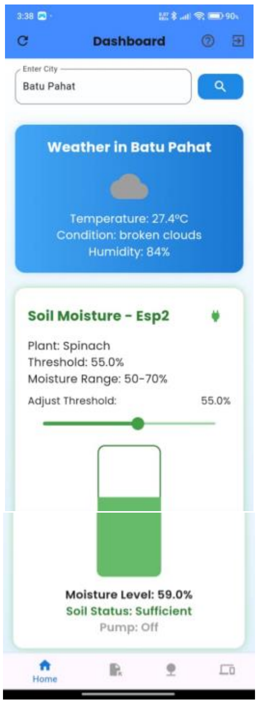


Fig. 13 Environmental Monitoring and Irrigation Control

Algorithm: Environmental Monitoring and Irrigation Control

:

1. Initialize Components
   - Set up Firebase Auth, Realtime Database, SharedPreferences, and WeatherService.
   - Initialize controllers for city input and variables for weather, moisture, and device data.
2. Load Saved Device
   - Retrieve selected device ID and name from SharedPreferences.
   - Fetch device data (name, plant, control state) from Firebase if device ID exists.
   - Update UI with device details or display "No device selected" if none.

   Code:
   ```
   if (savedDeviceId != null) {
       setState(() { deviceId = savedDeviceId; deviceName =
   savedDeviceName; _isLoading = true; });
       await _fetchDeviceData();
       setState(() { _isLoading = false; });
     } else {
       setState(() { moistureLevel = 'Please select a device'; soilStatus
   = 'N/A'; _isLoading = false; });
   ```
3. Fetch Weather Data
   - Retrieve weather data (temperature, humidity, condition) from OpenWeatherAPI for the specified city.
   - Save weather data to Firebase under user's device every 20 minutes.
   - Update UI with weather details and handle errors.

   Code:
   ```
   setState(() {
         _currentCity = cityToFetch;
         _weatherDescription = weatherData['weather'][0]['description'];
         _temperature = weatherData['main']['temp'];
         _humidity = weatherData['main']['humidity'].toDouble();
         _isLoading = false;
         _isRefreshing = false;
       });
   ```
4. Fetch and Display Real-Time Soil Moisture Data
   - Stream moisture data from Firebase path /users/$userId/devices/$deviceId/moisture_data, ordered by timestamp, limited to latest entry.
   - Parse moisture level and pump status, updating UI with formatted moisture percentage and soil status (Dry/Sufficient based on threshold).
   - Handle errors or invalid data formats with appropriate messages.

   Code:
   ```
   setState(() {
           moistureLevel = moisture.toStringAsFixed(1);
           soilStatus = moisture < plantThreshold ? 'Dry' : 'Sufficient';
         });
   ```
5. Automatic Irrigation Control
   - IoT device (ESP32) monitors moisture level against threshold stored in Firebase.
   - If moisture is below threshold and weather is not rainy/cloudy (checked via Firebase weather data), activate pump.

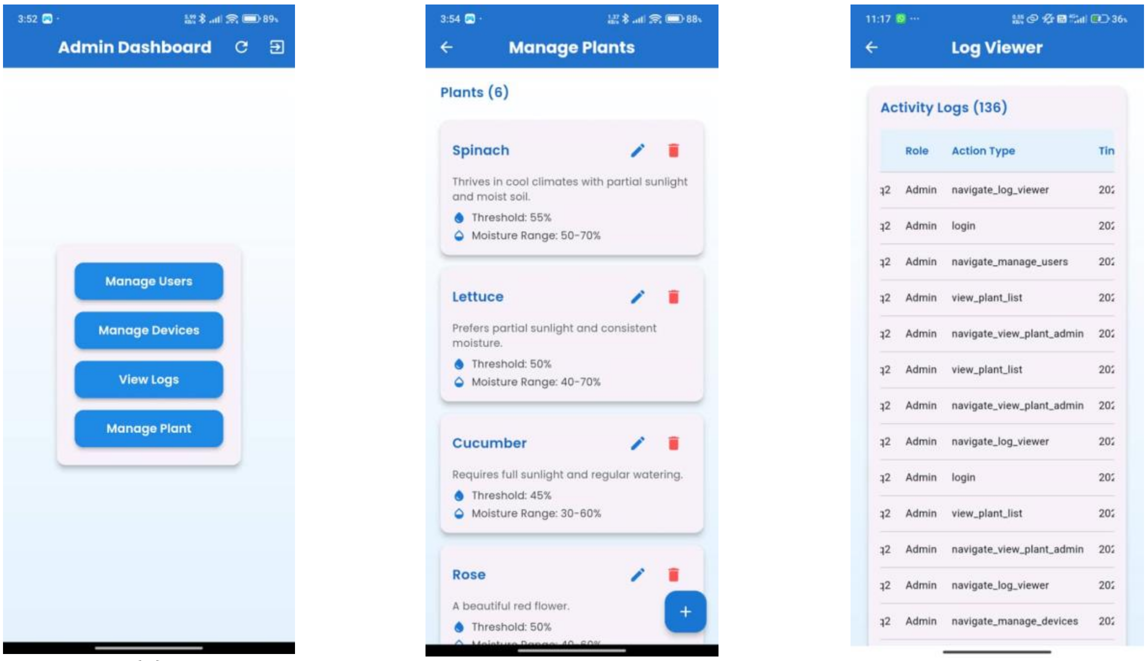
   IoT Code:
   ```
   if (moisturePercent < MOISTURE_THRESHOLD) {
       if (isWeatherSuitableForIrrigation()) {
         digitalWrite(RELAY_PIN, LOW);
         Serial.println("Pump ON: Moisture below threshold (" +
   String(moisturePercent) + "% < " + String(MOISTURE_THRESHOLD) + "%) and
   suitable weather");
       } else {
         digitalWrite(RELAY_PIN, HIGH);
         Serial.println("Pump OFF: Moisture below threshold (" +
   String(moisturePercent) + "% < " + String(MOISTURE_THRESHOLD) + "%) but
   paused due to weather");
       }
   ```
6. Adjust Soil Moisture Threshold
   - Allow user to adjust threshold via slider, disabled if device is powered off.
   - Save new threshold to Firebase under users/$userId/devices/$deviceId/selectedPlant.
   - Update local state and notify user of success.

   Code:
   ```
   Map<dynamic, dynamic> plantData = snapshot.exists ?
   Map.from(snapshot.value as Map) : {};
     plantData['threshold'] = newThreshold;
     await _dbRef
       .child('users/$userId/devices/$deviceId/selectedPlant')
       .set(plantData);
     setState(() {
       plantThreshold = newThreshold;
     });
   ```

Fig. 14 Environmental Monitoring and Irrigation Control Algorithm

5.1.6 Admin Management & Monitoring

A complete list of registered users is available from the Firebase Real-time Database. New user accounts can be added, existing user details can be updated, and accounts can be deleted along with all related data. All registered IoT devices are shown with their details and assigned users, also stored in Firebase. For admin manage devices, assignments to user can be changed, and devices can be removed, which also clears their WiFi settings and deletes their records from the database. Plant profiles can be viewed, created, edited, or deleted. Each profile includes specific settings, saved in Firebase, to support proper irrigation control. System logs from Firebase show records of user actions, device conditions, and irrigation activities, helping to monitor and manage the system effectively.



(a) (b) (c)

Fig. 15 (a)Admin Dashboard Screen (b) Admin Manage Screen (c) Admin View Logs

5.2 System Testing

A testing is conducted before the application and IoT device are published. The testing is aimed to ensure all the requirements were met including functional and non-functional requirements.

5.2.1 Test Plan

A test plan was made to define the overall approach to system testing, including objectives, schedule, deliverables, resource allocation, and effort estimates. It guides the testing activities and explains the testing procedures. Table 8 until Table 9 will list the test plan for the proposed project.

Table 8 User Functional Requirements Testing

Test	Test Cases	Results
Register	• Verify successful account creation with valid email and password.	Pass
	• Confirm error message for duplicate email registration.	Pass
	• Validate email verification sent.	Pass
	• Test successful registration using Google Sign-In.	Pass
Login	• Test login with valid email/password.	Pass
	• Show error message for incorrect credentials.	Pass
	• Check password reset email was send to user email and set a new password for forgotten password.	Pass
	• Test login via Google Sign-In.	Pass
Logout	• Make sure logout works safely and securely.	Pass

Table 8 (Cont.)

Manage Profile	• Check that profile details can be viewed and edited. • Make sure changes are saved to Firebase.	Pass Pass
Add Device	• Test adding a device through BLE pairing. • Check that WiFi details and device name are entered. • Make sure device info is saved to Firebase and shown on the Manage Devices page.	Pass Pass Pass
Remove Device	• Test removing a device from the app. • Check that a command is sent to clear WiFi settings via Firebase. • The device is deleted from Firebase after confirmation.	Pass Pass Pass
Select Device	• Users can select a device from the device list to monitor and control.	Pass
Weather Integration	• Test fetching temperature, humidity, and conditions from OpenWeatherAPI. • Make sure weather info shows on the Dashboard. • Confirm irrigation is paused when it's rainy or cloudy.	Pass Pass Pass
Monitor Real-Life Sensor Data	• Real-time soil moisture data displayed from Firebase. • Sensor data was update on mobile application.	Pass Pass
Automatic Irrigation	• Irrigation trigger when soil moisture falls below threshold. • Weather data integration pauses irrigation during rain/cloudy conditions. • IoT device controls pump based on sensor/weather data.	Pass Pass Pass
Plant List	• Selecting a plant from list to updates IoT device threshold.	Pass
Adjust Threshold	• Manual adjustment of soil moisture threshold per IoT device. • Threshold updates saved to Firebase and synced with device.	Pass Pass
Notifications Alert	• Push notifications for low soil moisture, device disconnection, irrigation events.	Pass
Reporting	• Report generation from historical sensor data.	Pass

Table 9 Admin Functional Requirements Testing

Test	Test Cases	Results
Login	• Test admin login. • Make sure only the admin can access admin-only features. • Show error message when login details are wrong.	Pass Pass Pass
Manage User	• Check that all users are listed from Firebase. • Test creation of new user accounts. • Test edit existing user accounts. • Test deleting user accounts.	Pass Pass Pass Pass
Manage IoT devices	• Show list of all devices with user info from Firebase. • Test editing device info for a selected user. • Test deleting a device, clearing its WiFi credentials, and updating Firebase.	Pass Pass Pass
Manage Plants	• Check that all plant profiles are shown from Firebase • Test adding new plant profiles with their settings. • Test editing of existing plant profiles. • Test deleting plant profiles.	Pass Pass Pass Pass
View Logs	• Check system logs from Firebase for user actions, device status, and irrigation events.	Pass

5.2.2 User Acceptance Testing

User Acceptance Testing (UAT) checks if a system works as expected for users. It involves real users testing the app to ensure it meets their needs and is easy to use. UAT helps confirm that features, such as registering, logging in, managing devices, or viewing reports, function correctly in real-world scenarios. Table 10 until Table 11 will list the User Acceptance Testing results of user and admin.

Applied Information Technology and Computer Science Vol. 6 No. 2 (2025) p. 190-207

Table 10 User Acceptance Testing Form for User

No	Description	Scale 1	2	3	4	5	Total
1	Can user easily create an account using email or Google?	-	-	3	5	7	15
2	Is it easy to log in using email/password or Google, and reset the password if needed?	-	-	2	5	8	15
3	Can the user view and change profile details without problems?	-	-	2	5	8	15
4	Is adding a device through Bluetooth (BLE) simple?	-	-	4	6	5	15
5	Can a device be removed from the app without issues?	-	-	4	6	5	15
6	Is choosing a device for monitoring and control easy to do?	-	-	3	7	5	15
7	Does the dashboard clearly show weather info to help with irrigation decisions?	-	-	3	6	6	15
8	Is soil moisture data shown accurately and updated in real-time?	-	-	2	6	7	15
9	Does the system start and stop watering correctly based on soil and weather data?	-	-	3	6	6	15
10	Is choosing a plant to set device thresholds easy and clear?	-	-	2	5	8	15
11	Can the user easily change soil moisture thresholds, and are changes saved correctly?	-	-	2	6	7	15
12	Are alerts about system events (like low moisture or disconnection) sent quickly and clearly?	-	-	3	6	6	15
13	Are reports based on past sensor data helpful?	-	-	4	6	5	15

Table 11 User Acceptance Testing Form for Admin

No	Description	Scale 1	2	3	4	5	Total
1	Can the admin login secure and only gives access to admin features?	-	-	1	5	9	15
2	Can the admin manage users (view, add, edit, and delete) without problems?	-	-	3	5	7	15
3	Can the admin manage devices (view, change, remove) without problems?	-	-	3	6	6	15
4	Can the admin manage plants (view, add, edit, or delete) plant profiles without problems?	-	-	1	6	8	15
5	Can the admin easily check logs showing user actions and system events?	-	-	1	5	9	15

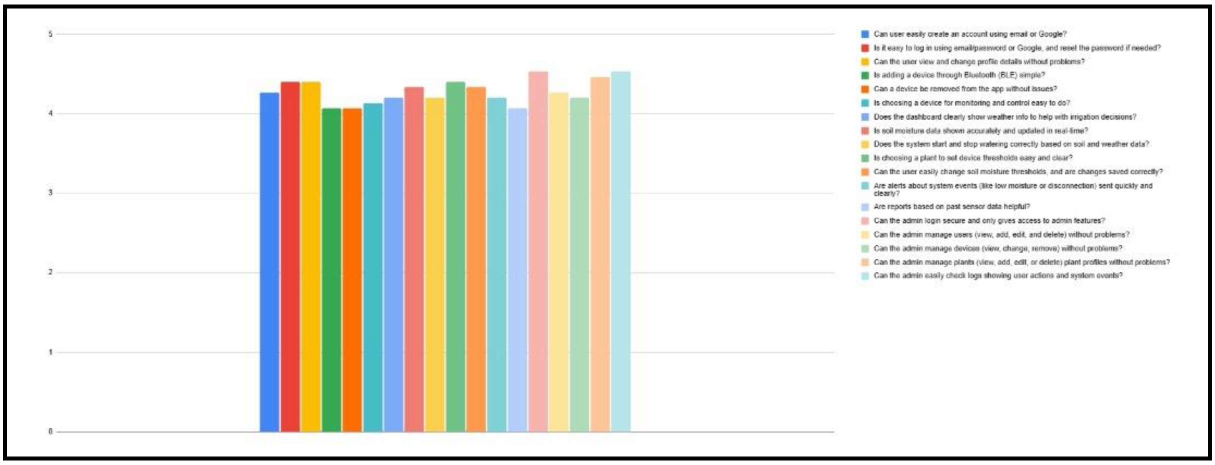


Fig. 16 User Acceptance Testing (UAT) Result

6. Conclusion

The Smart Irrigation Solution used the latest technology in trends, Internet of Things (IoT) to provide a more efficient and convenient solution for home gardeners. Our system achieves its primary objectives by automating the watering process, significantly reducing manual intervention and water wastage. Through the application, users can monitor real-time environmental data, control irrigation schedules remotely, and manage multiple IoT devices seamlessly, leading to improved convenience and more sustainable water usage. However, the system does have some limitations. Its effectiveness depends on reliable internet connectivity, which may restrict use in areas with poor network coverage. The weather integration depends only on data from OpenWeatherAPI, which may not always capture small-scale weather conditions accurately. Despite these limitations, the Smart Irrigation Solution represents a meaningful step toward smarter, more sustainable home gardening practices. Overall, Smart Irrigation Solution aims to simplify daily life while contributing to a more sustainable environment.

Acknowledgement

The author would like to thank the Faculty of Computer Science and Information Technology, University Tun Hussein Onn Malaysia for its support.

Conflict of Interest

Authors declare that there is no conflict of interests regarding the publication of the paper.

Author Contribution

This journal requires that all authors take public responsibility for the content of the work submitted for review. The contributions of all authors must be described in the following manner:

The authors confirm contribution to the paper as follows: ***study conception and design:*** *Ng Wei Kang, Rosmamalmi Mat Nawi;* ***data collection:*** *Ng Wei Kang, Rosmamalmi Mat Nawi;* ***analysis and interpretation of results:*** *Ng Wei Kang, Rosmamalmi Mat Nawi;* ***draft manuscript preparation:*** *Ng Wei Kang, Rosmamalmi Mat Nawi. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] R. Al-Yahyai et al., "An overview of smart irrigation systems using IoT," ScienceDirect, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772427122000791

[2] R. Q. Grafton et al., "Weak and Strong Sustainability of Irrigation: A Framework for Irrigation Practices Under Limited Water Availability," Frontiers in Sustainable Food Systems, vol. 4, p. 17, 2020. doi: 10.3389/fsufs.2020.00017

[3] P. Carrión et al., "IoT-Based Smart Irrigation Systems: An Overview on the Recent Trends on Sensors and IoT Systems for Irrigation in Precision Agriculture," Sensors, vol. 20, no. 4, p. 1042, 2020. doi: 10.3390/s20041042

[4] HashStudioz, "IoT Smart Irrigation Systems for Enhancing Agriculture Efficiency," 2024. [Online]. Available: https://www.hashstudioz.com/blog/enhancing-agriculture-with-iot-based-smart-irrigation-systems/

[5] DripWorks, "Drip Irrigation Vs. Traditional Watering Methods: A Comparative Study." May. 16, 2024 [Online]. Available: https://www.dripworks.com/blogdrip-irrigation-vs-traditional-watering-methods-a-comparative-study/

[6] A. Morchid et al., “IoT-enabled smart agriculture for improving water management: A smart irrigation control using embedded systems and Server-Sent Events,” Scientific African, vol. 27, p. e02527, Dec. 2024, doi: https://doi.org/10.1016/j.sciaf.2024.e02527.

[7] United Nations, "Growing at a slower pace, world population is expected to reach 9.7 billion in 2050 and could peak at nearly 11 billion around 2100 | United Nations Department of Economic and Social Affairs," Jun. 17, 2019. [Online]. Available: https://www.un.org/development/desa/en/news/population/world-population-prospects-2019.html

[8] W. Bank, "Employment in Agriculture (% of Total employment) (modeled ILO estimate)," World Bank Group, Jan. 7, 2025. [Online]. Available: https://data.worldbank.org/indicator/SL.AGR.EMPL.ZS

[9] S. Tan and L. J. Hong, "Trashing It Out: Waste Management in Asia," Jul. 1, 2020. [Online]. Available: https://connectivity.asean.org/resource/trashing-it-out-waste-management-in-asia/

[10] Mordor Intelligence, "APAC Waste Management Market Size & Share Analysis - Growth Trends & Forecasts (2025–2030)," 2025. [Online]. Available: https://www.mordorintelligence.com/industry-reports/apac-waste-management-market

[11] D. Taniushkina, A. Lukashevich, V. Shevchenko, I. Belalov, N. Sotiriadi, V. Narozhnaia, K. Kovalev, A. Krenke, N. Lazarichev, A. Bulkin, and Y. Maximov, "Case study on climate change effects and food security in Southeast Asia," Scientific Reports, vol. 14, Jul. 12, 2024. [Online]. Available: https://doi.org/10.1038/s41598-024-65140-y

[12] A. Kanimozhi and R. Vadivel, "Optimized water management for precision agriculture using IoT-based smart irrigation system," World Journal of Advanced Research and Reviews, vol. 21, no. 3, pp. 802–811, Mar. 2024. [Online]. Available: https://doi.org/10.30574/wjarr.2024.21.3.0682

[13] E. S. C. Pramanik, C. Gonesh, G. Saha, M. Billah, H. Khan, R. Che Mat, M. Hossain, R. Hoque, E. Satta, and H. Saha, IoT Based Smart Agricultural Crop Monitoring in Terms of Temperature and Moisture, Jan. 12, 2024. [Online]. Available: https://doi.org/10.13140/RG.2.2.25726.15684

[14] S. Balasundram, R. Shamshiri, S. Sridhara, and N. Rizan, "The role of digital agriculture in mitigating climate change and ensuring food security: An overview," Sustainability, vol. 15, no. 6, p. 5325, Mar. 2023. [Online]. Available: https://doi.org/10.3390/su15065325

[15] Rachio, "Rachio 3 Smart Sprinkler Controller," 2024. [Online]. Available: https://rachio.com/products/rachio-3/

[16] Orbit, "R Smart Irrigation & Leak Detection Bundle," 2024. [Online]. Available: https://www.orbitonline.com/products/xr-smart-irrigation-leak-detection-bundle

[17] Hydrawise, "Hydrawise Smart Controller and Water Management Software," 2024. [Online]. Available: https://www.hydrawise.com/content/hydrawise-smart-controller-and-water-management-software

[18] D. Sileshi, "Software Prototyping," Baeldung on Computer Science, Mar. 18, 2023. [Online]. Available: https://www.baeldung.com/cs/software-prototyping

[19] M. H. Bin Hasnan and N. Wahid, Kinect Integration into Computer Games for the Rehabilitation of Stroke Patients, Sep. 29, 2022.

[20] Y. Zhao, L. Xiao, C. Wei, R. Kazman, and Y. Yang, A Systematic Mapping Study on Architectural Approaches to Software Performance Analysis, arXiv preprint arXiv:2410.17372, Oct. 22, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2410.17372

[21] C. Singla, "Functional vs Non Functional Requirements," GeeksforGeeks, Apr. 28, 2020. [Online]. Available: https://www.geeksforgeeks.org/functional-vs-non-functional-requirements

[22] Y. Waykar, "Role of use case diagram in software development," International Journal of Management and Economics, Jan. 23, 2015.