Department of Computer Engineering

CENG350 Software Engineering

Software Architecture Description (SAD) for FarmBot

Group 80

By

Batuhan Akçan, 2580181

Mert Sönmezer, 2516920

Tuesday 14th May, 2024

Contents

List of Figures

List of Tables

Revision History

Date	Description	Version
06.05.2024	SAD Part-1 Completion	1.0.0
12.05.2024	SAD Final Completion	1.1.0

Table 1: Revision History

1. Introduction

This document is the Software Architecture Description (SAD) of an open source system called FarmBot [1]. The system has a website [2] and a GitHub repository [3].

1.1 Purpose and objectives of FarmBot

FarmBot harmonizes the efficiencies of monocrop and polycrop systems, aiming to create a sustainable and productive agricultural model. While monocropping makes use of machinery and reduces labor through uniform crops, it often leads to higher resource use and is harmful to the environment. Polycropping's diverse ecosystem reduces these inputs but typically lacks machinery support, requiring more labor. FarmBot's scalable technology bridges this gap, offering machine efficiency to diverse crops with minimal labor, similar to monocropping. The core purpose of FarmBot is to cultivate farms that combine abundance and resistivity with sustainability and efficiency. Objectives include:

- Enhancing the biological efficiency of polycrops while maintaining mechanical efficiency.
- Reducing dependency on external inputs like fertilizers and pesticides.
- Minimizing environmental impacts such as soil depletion and water pollution.
- Promoting sustainable farming through technological innovation.
- Achieving large-scale operations with minimal human labor.

1.2 Scope

FarmBot aims to modernize agriculture by automation, and it helps to apply both individual and commercial farming operations requiring efficiency, sustainability, and scalability. The project will deliver FarmBot's integrated software package, an open-source cloud-based Software as a Service (SaaS) solution.

FarmBot offers a bunch of key benefits aimed at transforming modern agriculture: it boosts crop yields, minimizes waste of resources, and improves operational efficiency. This software is designed to generalize precision farming, making it accessible to a wider audience by harnessing technology for sustainable agricultural practices. The goals of the FarmBot project are to make advanced farming technology more universally applicable, support data-driven agricultural methods, and inspire society to adopt cutting-edge agricultural systems. The development and features of the FarmBot software are rigorously crafted to support these objectives, which ensure a direct contribution to enhancing agricultural productivity and sustainability through technological advancement. Some components of the software are:

- A user-friendly web frontend which enable users to design and plan farm layouts.
- Decision support system which includes data mapping and analysis tools that helps the user to store, access, and analyze farming data, thus, utilize analytics for optimizing farming operations.
- User, farm, and equipment profile management systems. By means of user profiles, work and data of the user can be saved and later accessed. Also, user profiles provide authentication against the hacking purposes. Farm profiles include the data associated with the farm including plant layouts, scheduled operations, statistics, and history of the farm. Via the equipment profiles, information about equipments is stored and easily modified as equipment is upgraded or become inactive for maintenance.

- Microcontroller software for direct hardware interaction that helps the user to control FarmBot hardware.
- Open data repositories for community-driven knowledge sharing.

1.3 Stakeholders and their concerns

The FarmBot project comprises a large spectrum of stakeholders, each with varying levels of technical expertise and operational needs. The stakeholders include:

- **Hobbyist Gardeners:** Their main concern is the ease of use and reliability of FarmBot to cultivate their home gardens. They require a system that is intuitive and doesn't necessitate advanced technical knowledge.
- **Professional Farmers:** They prioritize FarmBot's efficiency and scalability to ensure it integrates smoothly into larger-scale farming operations, often focusing on the system's capability to increase crop yield and reduce manual labor.
- **Educators:** They utilize FarmBot as an educational device, hence they are interested in the system's functionality and its potential as a teaching aid to demonstrate modern farming techniques.
- **Researchers:** They utilize FarmBot for experimental farming methods and crop studies. Their concerns include the adaptability of the system to different research settings and the accuracy of data collection for sustainable agriculture practices.
- **Software Developers:** A group primarily concerned with the system's programmability and open-source nature, ensuring that enhancements and community contributions can be made effectively.
- **System Administrators:** Focus on maintaining the FarmBot system, particularly its operational integrity and data security. They need comprehensive control over the software and hardware for updates and troubleshooting.

Each stakeholder group contributes to the FarmBot ecosystem, driving the project towards a future where technology and agriculture unite to create efficient and sustainable farming practices.

2. References

[1] R. L. Aronson, "Farmbot: Humanity's open-source automated precision farming machine." https://farm.bot/pages/whitepaper, 2013.

[2] https://farm.bot.

[3] https://github.com/FarmBot.

3. Glossary

AI Artificial Intelligence.

AMQP Advanced Message Queuing Protocol.

API Application Programming Interface.

CRUD Create Read Update Delete.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

IoT Internet of Things.

MQTT Message Queue Telemetry Transport.

OS Operating System.

SaaS Software as a Service.

SAD Software Architecture Description.

SQL Structured Query Language.

SRS Software Requirements Specification.

URL Uniform Resource Locator.

UV Ultraviolet.

WiFi Wireless Fidelity.

4. Architectural Views

4.1 Context View

4.1.1 Stakeholders' uses of this view

The Context View is utilized by various stakeholders to understand how FarmBot interfaces with its environment and other systems:

- **Hobbyist Gardeners and Professional Farmers:** They might consult the Context View to grasp how FarmBot integrates with weather services, plant databases, and other agricultural tools, enhancing their ability to use FarmBot effectively within their existing workflows.
- **Educators:** Educators leverage the Context View to illustrate to students the broader ecosystem of FarmBot, including its connectivity to cloud services and the exchange of information between FarmBot and external agricultural knowledge bases.
- **Researchers:** Researchers examine the Context View to ascertain how their experimental designs will interact with the various components of FarmBot, and to ensure the integrity of data exchange for their studies.
- **Software Developers:** Developers rely on the Context View to conceptualize how the different pieces of the FarmBot software interact with the hardware components, the Web Application, and third-party APIs like OpenFarm.cc.

- **System Administrators:** Admins use the Context View to monitor and maintain the health of the system, ensuring the integration points between FarmBot and external services are functioning as expected.
- **Open-Source Contributors and Community Members:** They might use the Context View to better understand where their contributions can be most effective and how to interface with FarmBot's various system components.

Each stakeholder interacts with the Context View based on their unique concerns and needs, ensuring they can effectively engage with and support the FarmBot system.

4.1.2 Context Diagram

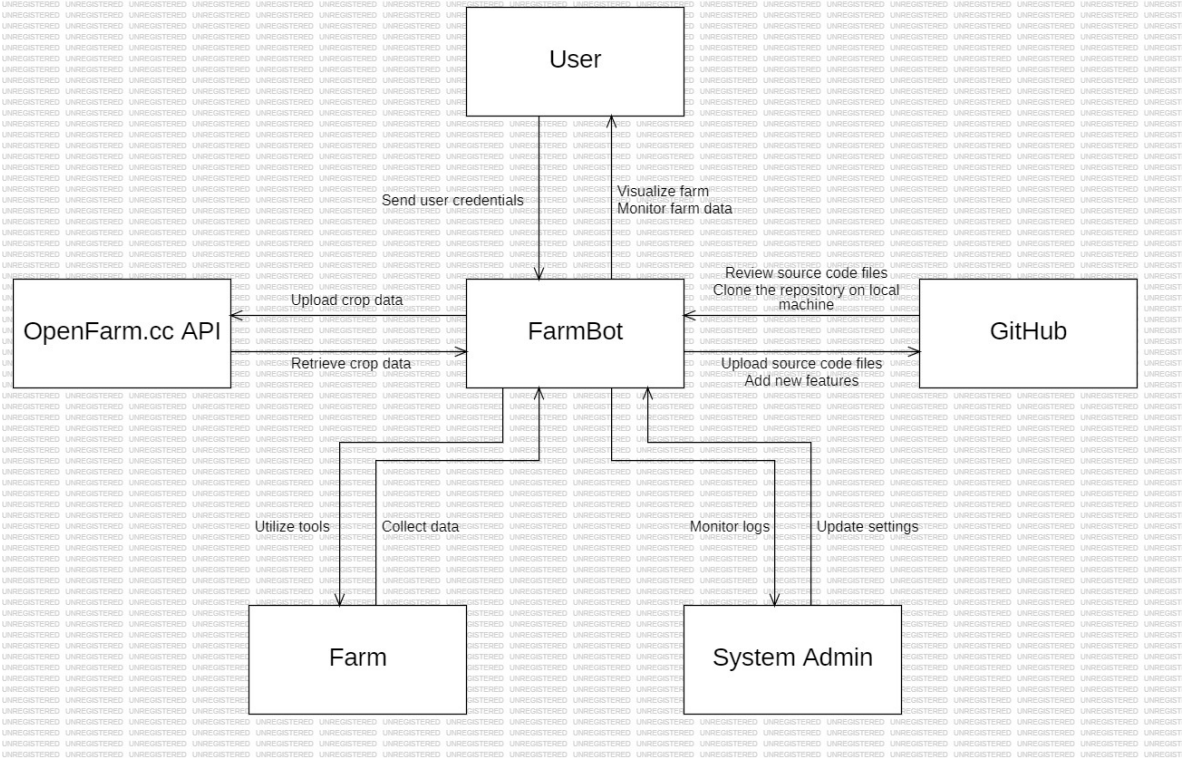

Figure 4.1: System Context Diagram

The FarmBot system operates as an independent entity, interfacing with various external components to enhance its autonomous agricultural capabilities. Key interactions include:

- **OpenFarm.cc API:** FarmBot utilizes this service to enrich its database with extensive plant cultivation data, enabling users to make informed decisions about crop management and optimizing agricultural practices.
- **User:** Individuals engage directly with FarmBot, providing their credentials to gain personalized access, and in return, they receive visual and data-driven insights into the farm's operations, contributing to an interactive farming experience.
- **System Admin:** This role is critical to the health of the FarmBot system, involving the monitoring of logs, the collection of data, and the adjustment of settings to ensure optimal performance.
- **GitHub:** Serves as a collaborative platform for the FarmBot community, where developers and contributors manage the codebase, addressing system enhancements, bug fixes, and feature requests.
- **Farm:** The physical environment where FarmBot's tools are deployed, and data is collected to perform a variety of tasks such as planting, watering, and monitoring environmental conditions.

The context diagram emphasizes the balance between user interaction, automated farming processes, and data exchange, ensuring that FarmBot functions as a cohesive system for sustainable and technologically advanced agriculture.

4.1.3 External Interfaces

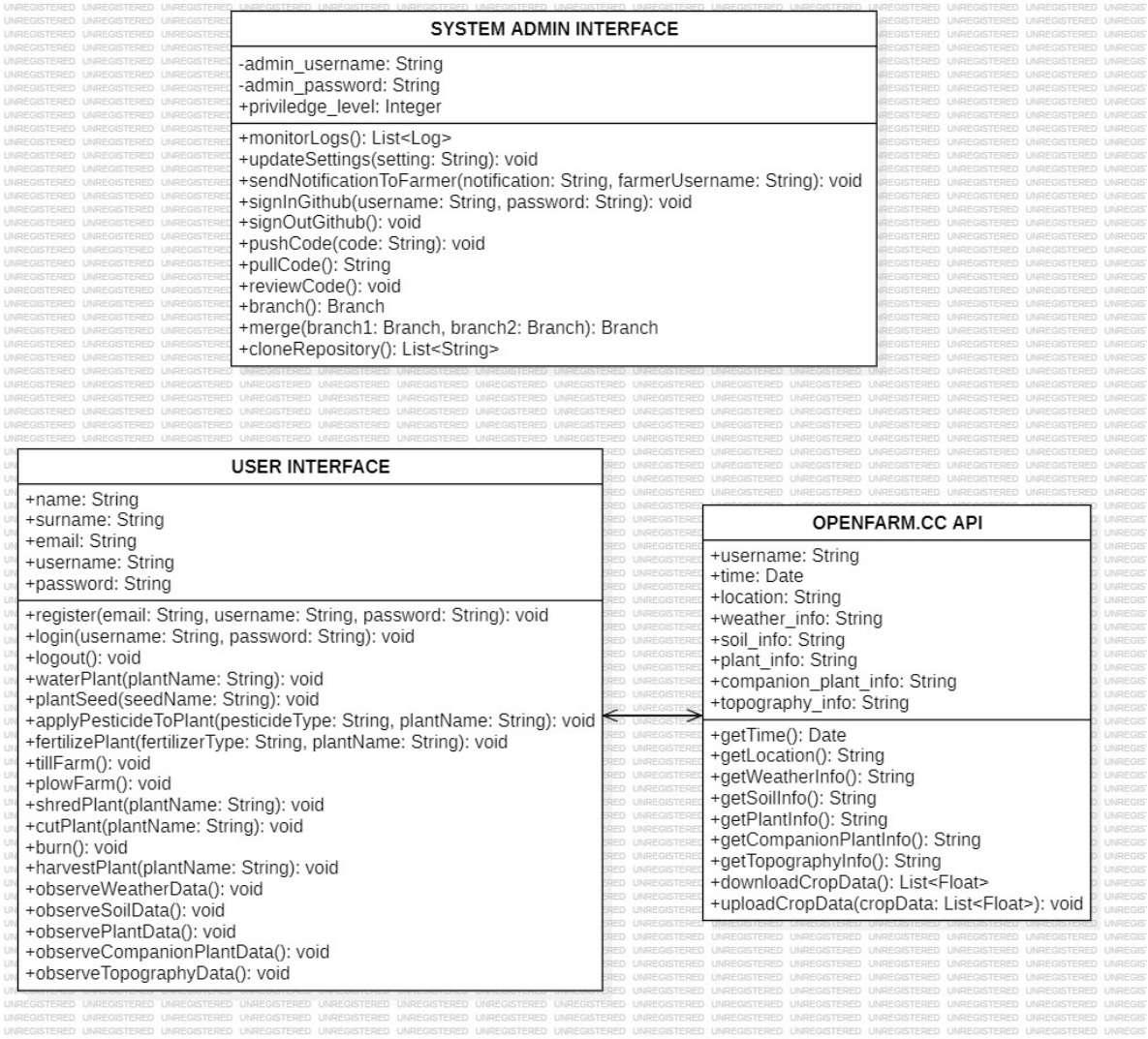

Figure 4.2: External Interfaces Class Diagram

In the architecture of the FarmBot system, external interfaces play a pivotal role in seamless interaction between users, administrators, and the broader online community. Here is an overview of each interface's core functionalities:

- **User Interface:** Acts as the direct touchpoint for users to manage and interact with FarmBot's diverse agricultural tasks such as planting and harvesting, leveraging a straightforward design that accommodates varying levels of user expertise. Methods and fields of this interface can be described as follows:
  - **name, surname, email:** Personal identification fields for user accounts.
  - **username, password:** Credentials for the user.
  - **register**(), **login**(), **logout**(): Operations for signing in and signing out from the FarmBot application.
  - **waterPlant**(): Command to initiate the watering of plants.
  - **plantSeed**(): Function to start the planting of seeds.
  - **applyPesticideToPlant**(): Function to apply pesticides to plants.
  - **fertilizePlant**(): Command to fertilize plants.
  - **tillFarm**(): Operation for tilling the soil.
  - **plowFarm**(): Function to plow the farm area.
  - **shredPlant**(): Command to shred unwanted plants.
  - **cutPlant**(): Function to cut or prune plants.
  - **burn**(): Command that could relate to a controlled burn function for crop management.
  - **harvestPlant**(): Function to initiate the harvesting of plants.
  - **observeWeatherData**(): Function to display current and forecasted weather data.
  - **observeSoilData**(): Function to display real-time soil data such as moisture levels, pH, and nutrient profiles.

  - **observePlantData**(): Function to display detailed information about the plants.
  - **observeCompanionPlantData**(): Function to display information about companion planting strategies.
  - **observeTopographyData**(): Function to display information about topography data such as latitude, longitute, and altitude.

- **System Admin Interface:** Tailored for administrators to perform system-level operations, including monitoring and updating settings, thus ensuring the robustness and reliability of FarmBot's service. Methods and fields of this interface are:
  - **admin_username, admin_password:** Credentials for the system administrator.
  - **priviledge_level:** Determines the level of access and control the admin has.
  - **monitorLogs**(): Function for monitoring system logs.
  - **updateSettings**(): Command to update system settings.
  - **sendNotificationToFarmer**(): Function to send notifications to farmers.
  - **signInGithub**(), **signOutGithub**(): Operations for signing in to and signing out from the GitHub repository.
  - **pushCode**(), **pullCode**(): Functions for pushing and pulling code, respectively, to and from the repository.
  - **reviewCode**(): Command to review code changes.
  - **branch**(), **merge**(): Operations for branching and merging code for version controlling purposes.
  - **cloneRepository**(): Function to clone the repository for local work or backup.

- **OpenFarm.cc API:** Serves as a conduit for accessing and uploading detailed plant cultivation data, which is crucial for informed farming decisions and enhancing FarmBot's agricultural database. Methods and fields of this interface can be listed as:
  - **username, time, location:** Identifiers for the user and timing/location for the data retrieval or action.
  - **weather_info, soil_info, plant_info, companion_plant_info, topography_info:** Data fields containing environmental and cultivation details.
  - **getTime**(), **getLocation**(), **getWeatherInfo**(), **getSoilInfo**(), **getPlantInfo**(), **getCompanionInfo**(), **getTopographyInfo**(): Functions to retrieve the current time, location details, and environmental conditions as well as specific plant and soil information.
  - **downloadCropData**(): Function to download data about crops.
  - **uploadCropData**(): Function to upload new crop data to OpenFarm.CC.

4.1.4 Interaction scenarios

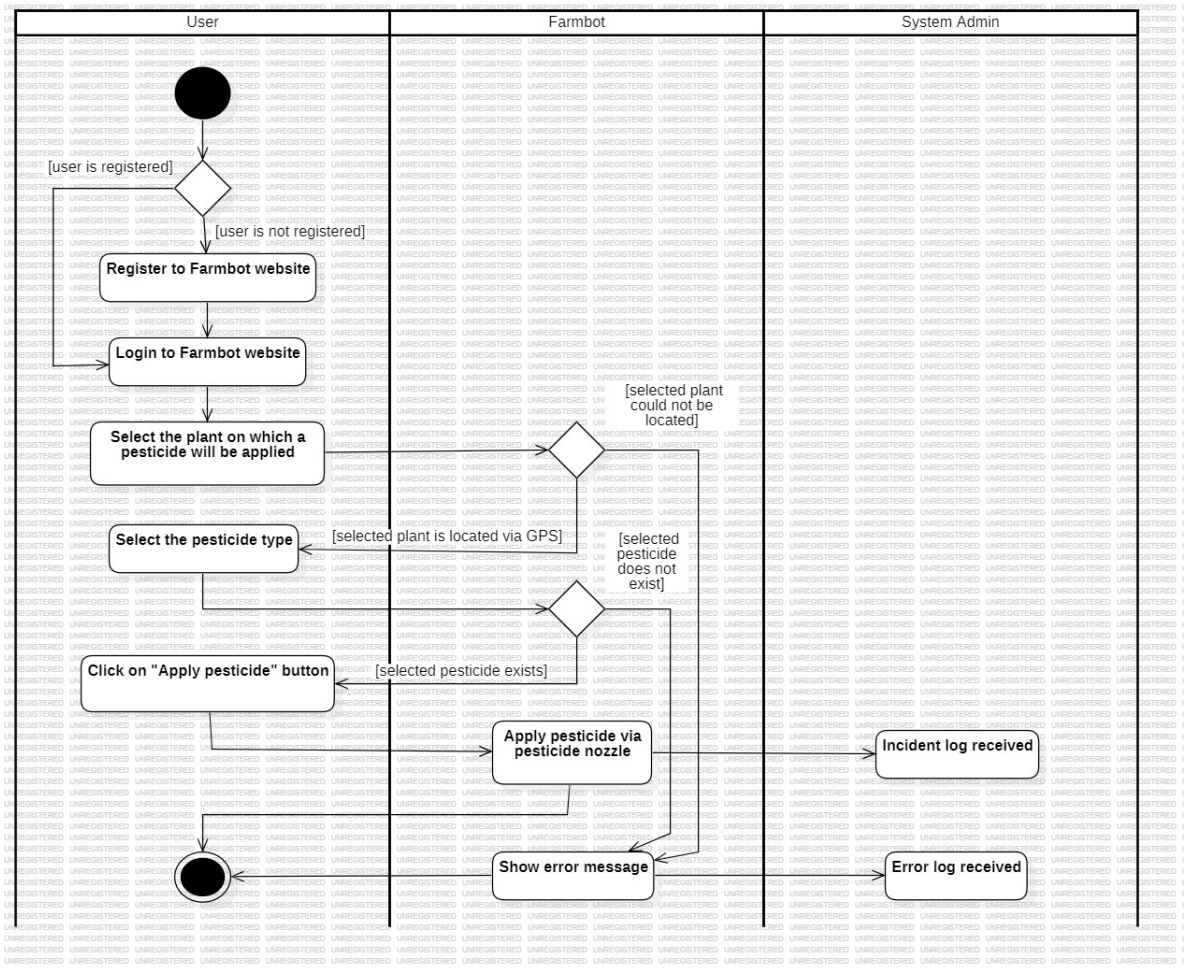

Figure 4.3: Activity Diagram of the "Apply Pesticide to Plant" Interaction Scenario

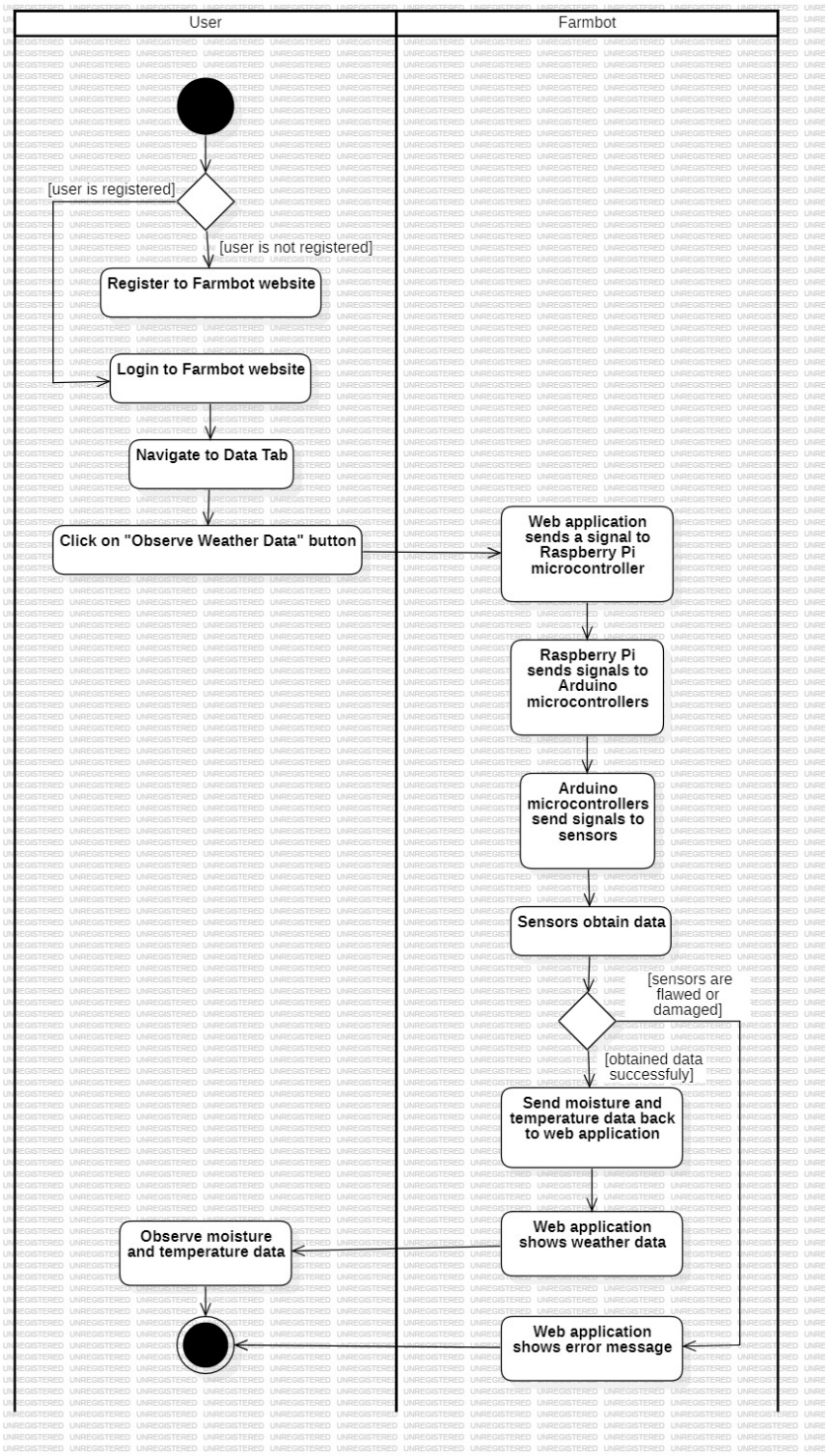

Figure 4.4: Activity Diagram of the "Observe Weather Data" Interaction Scenario

4.2 Functional View

4.2.1 Stakeholders' uses of this view

The Functional View provides critical insights into the system's operations and interactions, with a primary focus on the diverse functionalities offered by FarmBot. The stakeholders of the FarmBot project, ranging from developers and users to administrators and contributors, utilize this view for various purposes:

- **Developers:** Leverage the Functional View as a blueprint for understanding system operations, guiding them in creating functionalities. It also aids in identifying inter-component dependencies, essential for efficient code integration and system testing.
- **Users (Hobbyist Gardeners, Professional Farmers, and Educators):** Access the Functional View to comprehend the capabilities of FarmBot and how they can employ these features for automated farming, educational purposes, or agricultural research.
- **System Admins:** Utilize this view to monitor the efficacy of FarmBot's functions and to anticipate the impact of system updates or configurations.
- **Researchers:** Rely on the Functional View to investigate how FarmBot's features can be employed for experimental setups, data collection, and analysis, supporting their work in sustainable agriculture.
- **Contributors (Open-Source Community Users):** Use this view to see how their contributions fit into the broader FarmBot ecosystem and influence its functionality.

This Functional View acts as a lens through which the entire FarmBot operation can be understood, providing clarity to stakeholders about how their individual interactions with the system contribute to its goals of efficiency, sustainability, and user accessibility.

4.2.2 Component Diagram

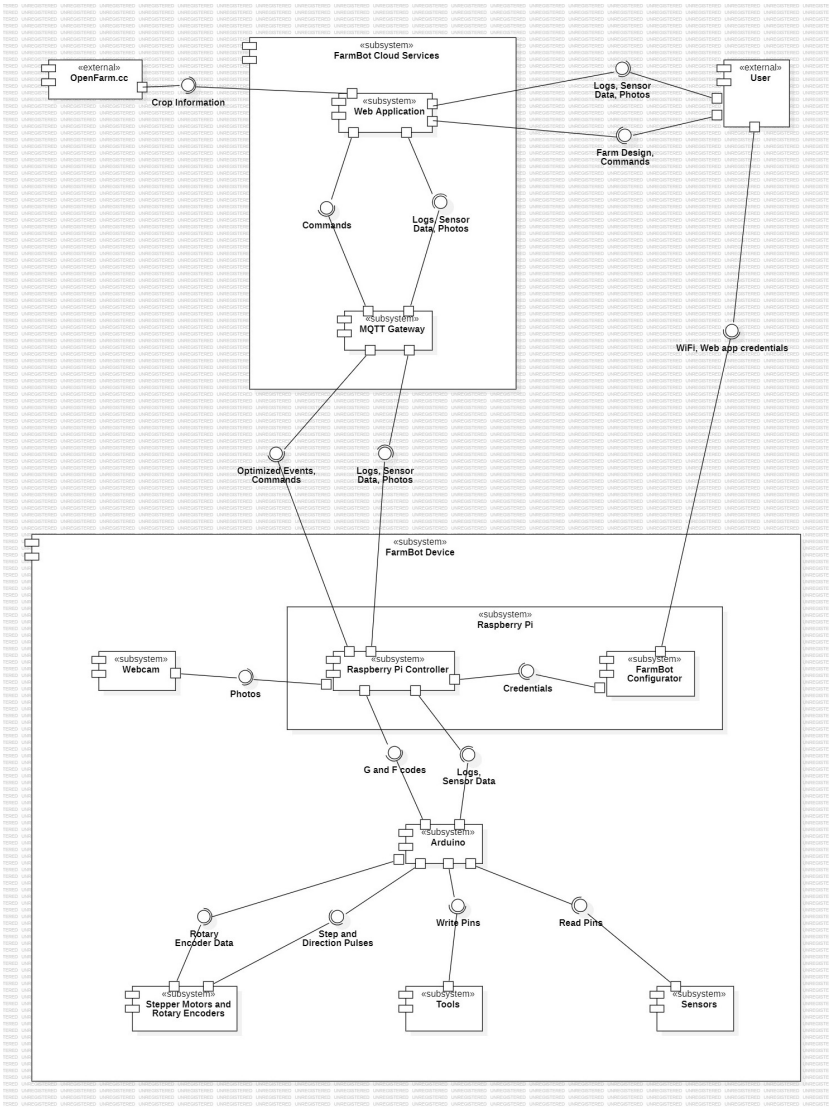

Figure 4.5: Component Diagram

The FarmBot system is composed of several interconnected subsystems: FarmBot Cloud Service, Web Application, MQTT Gateway, FarmBot device, Raspberry Pi, Webcam, FarmBot Configurator, Arduino, Stepper Motors & Rotary Encoders, Tools and Sensors. These subsystems are structured into hierarchies for efficient management and operation. For instance, the FarmBot Cloud Service encompasses the Web Application and MQTT Gateway, while the FarmBot Device integrates components like the Raspberry Pi, Webcam, FarmBot Configurator, Arduino, Stepper Motors & Rotary Encoders, and Tools and Sensors.

Within the FarmBot Cloud Service, the Web Application provides a user-friendly interface for configuring and managing the FarmBot operations. It facilitates the design of farm layouts, scheduling tasks, and viewing operational data. The MQTT Gateway acts as a communication broker that manages the messaging between the FarmBot device and the web application, ensuring timely and secure data transfer.

The FarmBot Device subsystem is central to the operation, containing several critical components:

- **Raspberry Pi:** Acts as the control hub, processing commands from the web application and managing device operations.
- **Webcam:** Captures real-time visual data of the farm, which is accessible via the web application for monitoring and decision-making.
- **FarmBot Configurator:** A setup tool that helps users initialize the device by connecting it to their network and configuring its settings.
- **Arduino:** Interfaces directly with physical components, interpreting G-code instructions to operate the motors, tools, and read sensor data.
- **Stepper Motors & Rotary Encoders:** These components drive the precise movement of the FarmBot along its tracks and provide feedback on position for accuracy.

- **Tools and Sensors:** Include various implements such as seed injectors, soil sensors, and watering nozzles that perform the actual agricultural tasks.

Each of these components works in harmony to ensure that the FarmBot operates efficiently, adhering to the precision farming model. The system's architecture supports scalability and modularity, allowing for future expansions and modifications to enhance capabilities or integrate new technologies as they become available. This structured yet flexible setup ensures that FarmBot can adapt to various farming environments and requirements, promoting innovation and improvement in automated farming technology.

4.2.3 Internal Interfaces

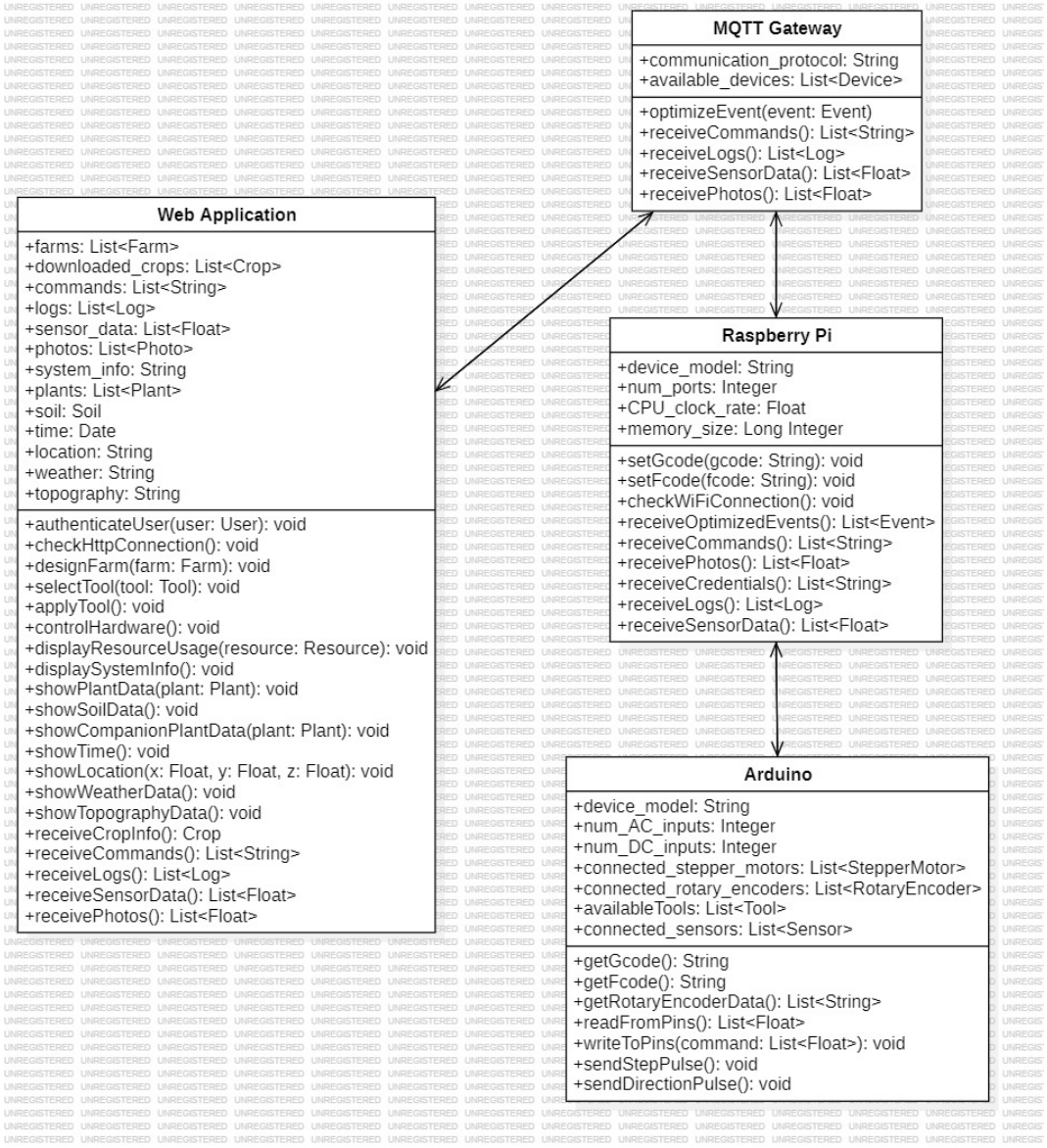

Figure 4.6: Internal Interfaces Class Diagram

FarmBot integrates a sophisticated combination of internal interfaces, each tasked with specific functions vital to the system's operation. These interfaces enable seamless communication and operational coordination among the various components of FarmBot.

- **Web Application Interface:** Acts as the primary user interaction platform, facilitating a variety of functionalities:
  - User Authentication and Connection Checks: authenticateUser, checkHttpConnection
  - Farm Design and Tool Management: designFarm, selectTool, applyTool
  - Hardware Control and Monitoring: controlHardware, displayResourceUsage, displaySystemInfo
  - Data Display and Environmental Monitoring: showPlantData, showSoilData, showCompanionData, showTime, showLocation, showWeatherData, showTopographyData
  - Data Reception: receiveCropInfo, receiveCommands, receiveLogs, receiveSensorData, receivePhotos
- **MQTT Gateway Interface:** Manages real-time communication, data reception, and event optimization:
  - Event Management and Data Reception: optimizeEvent, receiveCommands, receiveLogs, receiveSensorData, receivePhotos
- **Raspberry Pi Interface:** Serves as the central processing unit within the FarmBot device, handling command execution and data reception:
  - Command Setting and WiFi Connection Check: setGcode, setFcode, checkWiFiConnection
  - Optimized Event and Data Reception: receiveOptimizedEvents, receivePhotos, receiveCredentials, receiveLogs, receiveSensorData

- **Arduino Interface:** Directly controls the physical operations of FarmBot, interfacing with the mechanical systems and sensors:
  - Code Retrieval and Sensor Data Handling: getGcode, getFcode, getRotaryEncoderData
  - Pin and Motor Control: readFromPins, writeToPins, sendStepPulse, sendDirectionPulse

Each interface is meticulously designed to ensure that FarmBot operates efficiently, responding accurately to user inputs and effectively managing the automation of farming tasks. This structured interface design promotes clarity in component responsibilities, making the system both maintainable and scalable.

4.2.4 Interaction Patterns

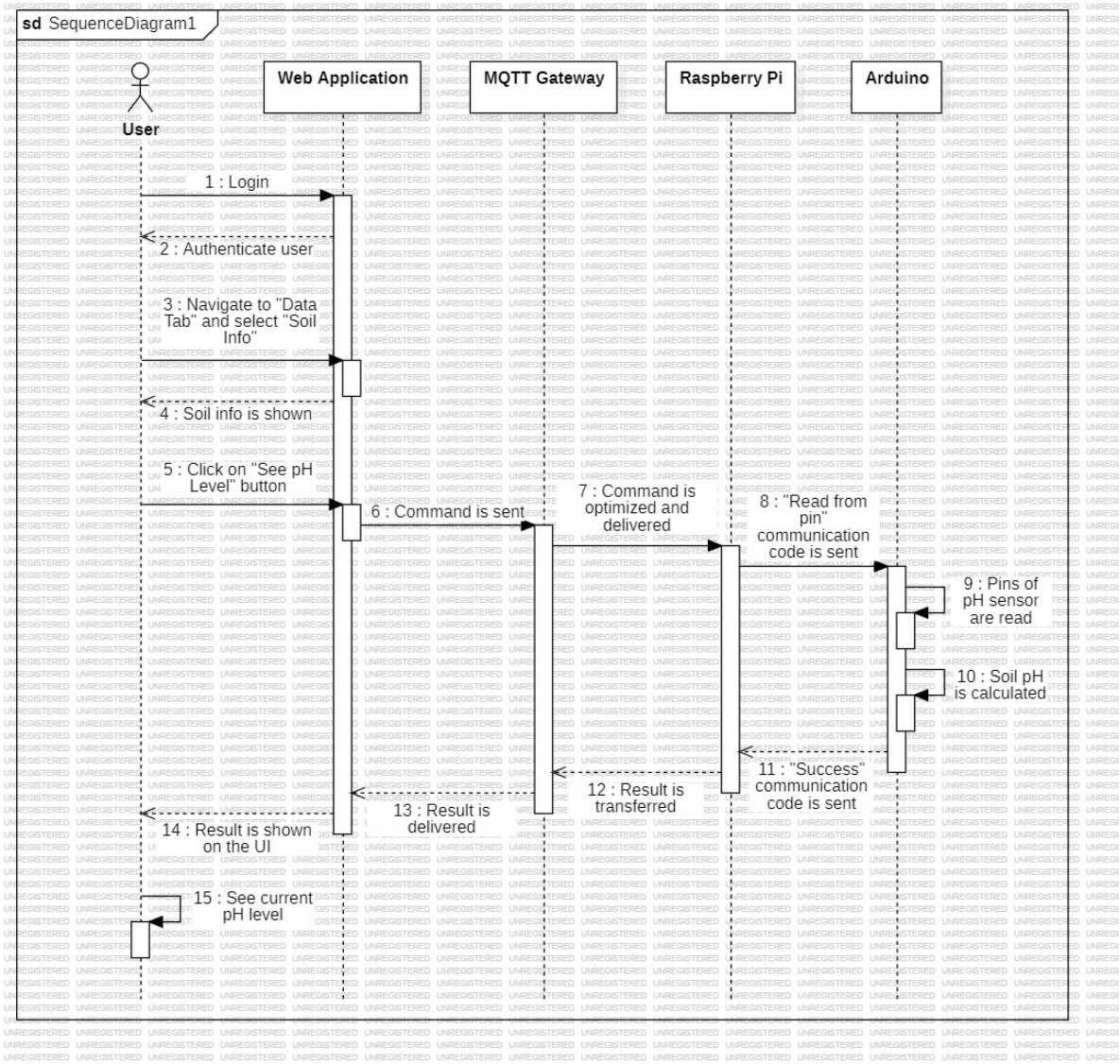

Figure 4.7: Sequence Diagram for "Show Current pH Level" Interaction Pattern

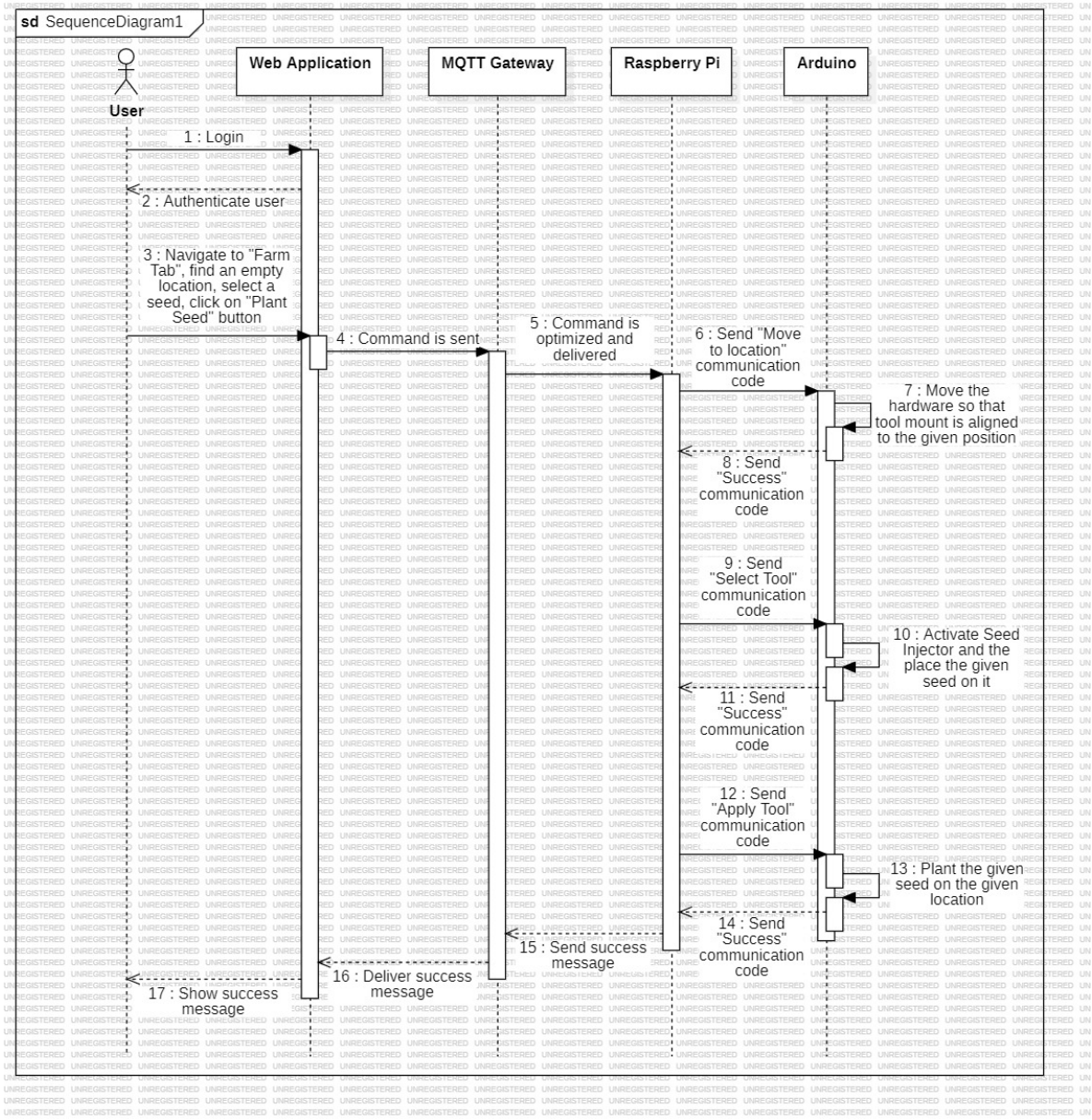

Figure 4.8: Sequence Diagram for "Plant Seed" Interaction Pattern

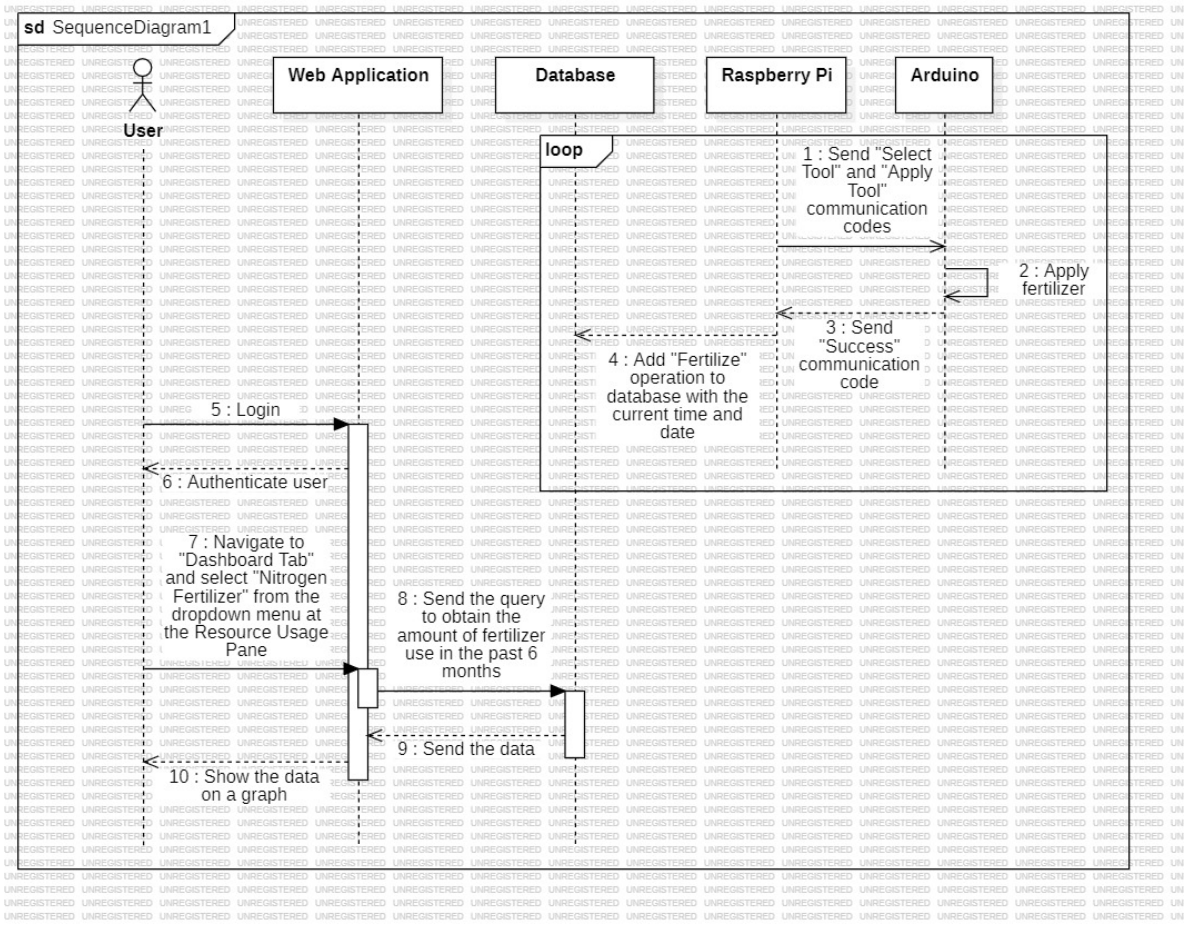

Figure 4.9: Sequence Diagram for "Display Fertilizer Usage" Interaction Pattern

4.3 Information View

4.3.1 Stakeholders' uses of this view

The Information View is instrumental for stakeholders in understanding and managing the flow of data within the system. Here is how various stakeholders might use the Information View:

- **Hobbyist Gardeners and Professional Farmers:** They refer to the Information View to know what kind of data is collected about their farming activities, such as soil conditions, weather data, and crop growth, which helps them in making informed decisions for their farming practices.
- **Educators:** Utilize the Information View to teach students about the types of agricultural data managed by FarmBot, which includes data structures related to plant health, environmental conditions, and automated responses of the FarmBot system.
- **Researchers:** Leverage the Information View to understand the specifics of data collected, stored, and analyzed by FarmBot, aiding in research related to sustainable farming practices, crop yield optimization, and ecosystem impacts.
- **Software Developers:** Use the Information View to comprehend the data architecture of FarmBot, informing their software development to enhance functionalities, create new features, and integrate with third-party data services.
- **System Administrators:** Rely on the Information View for database management, ensuring data integrity, security, and efficient data flow from sensors to user interfaces and vice versa. They also use this view to monitor data-related system performance and to troubleshoot issues.
- **Open-Source Contributors and Community Members:** Refer to the Information View to understand the data models and structures that FarmBot uses,

which helps them contribute effectively to the project by suggesting improvements or developing new data-driven features.

By understanding the Information View, all stakeholders can ensure that data handling within FarmBot aligns with the system's goals of providing an efficient, user-friendly, and educational automated farming experience.

4.3.2 Database Class Diagram

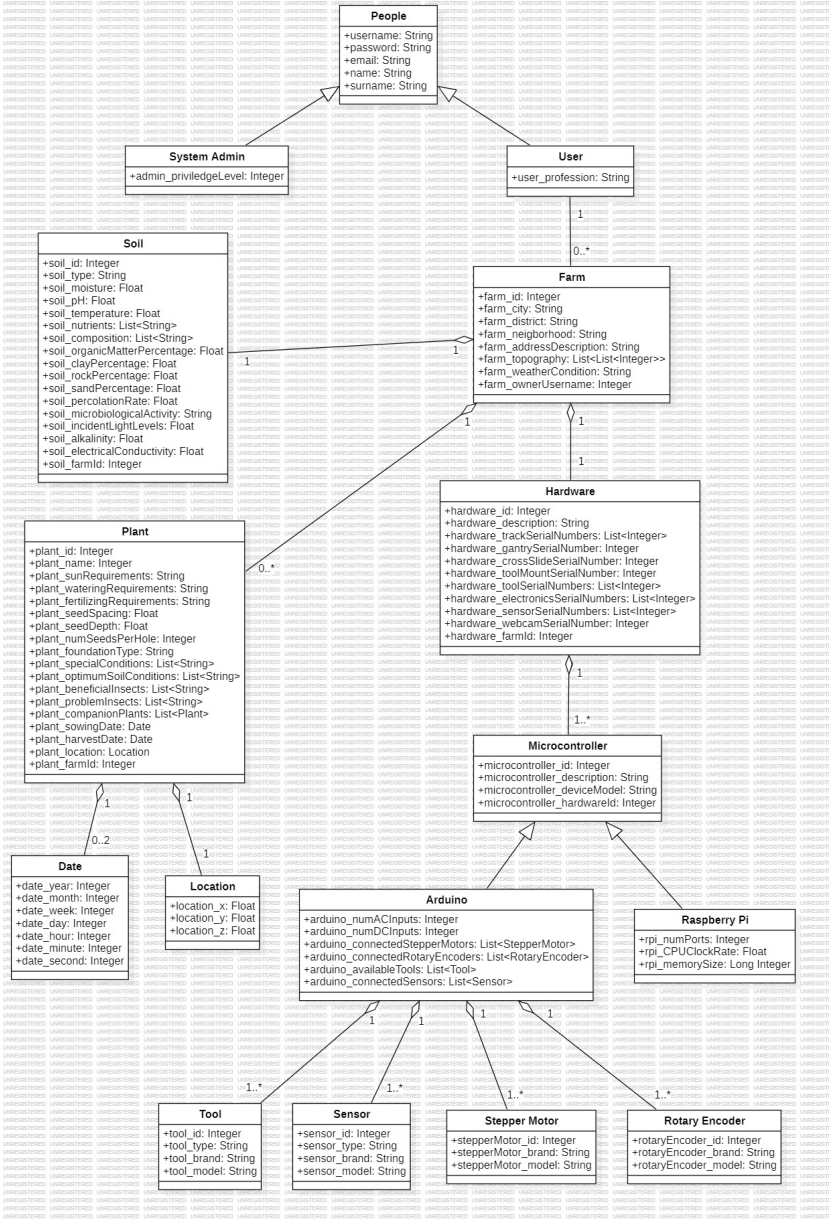

Figure 4.10: Database Class Diagram

4.3.3 Operations on Data

Table 4.1: CRUD Operations

Operation	CRUD Operations
AddUser	Create: Add a new user to the system. Read: - Update: - Delete: -
ChangePriviledgeLevel	Create: - Read: - Update: Change the priviledge level of system administrator. Delete: -
UpdateUserEmail	Create: - Read: - Update: Update a user's email address. Delete: -
RemoveUser	Create: - Read: - Update: - Delete: Delete a user from the system.
RegisterSoilData	Create: Register new soil data into the system. Read: - Update: - Delete: -

Continued on next page

Table 4.1: CRUD Operations (Continued)

UpdateSoilComposition	Create: - Read: - Update: Arrange the composition of soil by changing organic matter, clay, rock, and sand percentages. Delete: -
DeleteSoilData	Create: - Read: - Update: - Delete: Delete specific soil data from the database.
CreatePlant	Create: Add a new plant entry with full details. Read: - Update: - Delete: -
UpdatePlantRequirements	Create: - Read: - Update: Update a plant's sun, watering, and fertilizer requirements. Delete: -
GetSowingDate	Create: - Read: Read the sowing date of a plant. Update: - Delete: -

Continued on next page

Table 4.1: CRUD Operations (Continued)

GetHarvestDate	Create: - Read: Read the harvest date of a plant. Update: - Delete: -
DeletePlant	Create: - Read: - Update: - Delete: Remove a plant entry from the database.
CreateFarm	Create: Create a farm instance with given data. Read: - Update: - Delete: -
UpdateFarmAddress	Create: - Read: - Update: Update the address of a farm by changing city, district, neighborhood, etc. Delete: -
DeleteFarm	Create: - Read: - Update: - Delete: Delete a farm instance from the database.

Continued on next page

Table 4.1: CRUD Operations (Continued)

AddHardware	Create: Register new hardware in the system. Read: - Update: - Delete: -
GetSerialNumber	Create: - Read: Read the serial number of a hardware part. Update: - Delete: -
RemoveHardware	Create: - Read: - Update: - Delete: Delete hardware from the system.
AddMicrocontroller	Create: Add a new microcontroller configuration. Read: - Update: - Delete: -
UpdateMicrocontrollerDescription	Create: - Read: - Update: Update the description of an existing microcontroller. Delete: -

Continued on next page

Table 4.1: CRUD Operations (Continued)

DeleteMicrocontroller	Create: - Read: - Update: - Delete: Remove a microcontroller configuration from the system.
RegisterTool	Create: Add a new tool to the system. Read: - Update: - Delete: -
GetToolBrand	Create: - Read: Read the brand of a tool. Update: - Delete: -
RemoveTool	Create: - Read: - Update: - Delete: Delete a tool from the system.
AddSensor	Create: Add a new sensor to the database. Read: - Update: - Delete: -
GetSensorModel	Create: - Read: Read the model of a sensor. Update: - Delete: -

Continued on next page

Table 4.1: CRUD Operations (Continued)

DeleteSensor	Create: - Read: - Update: - Delete: Remove sensor data from the database.
AddStepperMotor	Create: Add a new stepper motor to the database. Read: - Update: - Delete: -
RemoveStepperMotor	Create: - Read: - Update: - Delete: Remove a stepper motor from the database.
RegisterRotaryEncoder	Create: Register a new rotary encoder to the database. Read: - Update: - Delete: -
DeleteRotaryEncoder	Create: - Read: - Update: - Delete: Delete a rotary encoder from the database.

4.4 Deployment View

4.4.1 Stakeholders' uses of this view

The Deployment View helps different stakeholders understand how software components are distributed across the hardware and network infrastructure:

- **Hobbyist Gardeners and Professional Farmers:** While they may not interact directly with the Deployment View, understanding that the software runs on both a cloud platform and local devices assures them of the system's resilience and continuous operation, even if their own internet connection is unstable.
- **Educators:** They can use the Deployment View to show students how the FarmBot software is distributed and interacts with various hardware components, from the cloud servers to the Raspberry Pi and Arduino controllers on the physical FarmBot units.
- **Researchers:** Interested in how the deployment of the system can affect data collection and the responsiveness of FarmBot to changes in the environment or in its operational commands.
- **Software Developers:** Utilize the Deployment View to ensure their code is optimized for the environments it will run in, whether it's on the lightweight Raspberry Pi or a robust cloud server, and understand the communication between these nodes.
- **System Administrators:** Rely heavily on the Deployment View to manage the physical and virtual resources that the FarmBot system depends on, ensuring that all components are properly maintained and configured for efficient operation.
- **Open-Source Contributors and Community Members:** May refer to the Deployment View to better understand how their contributions fit into the broader system, ensuring compatibility with the existing infrastructure and identifying areas where improvements can be made.

This view is essential for ensuring that the FarmBot system's deployment is aligned with its performance requirements and can scale to meet growing demand while remaining robust and reliable.

4.4.2 Deployment Diagram

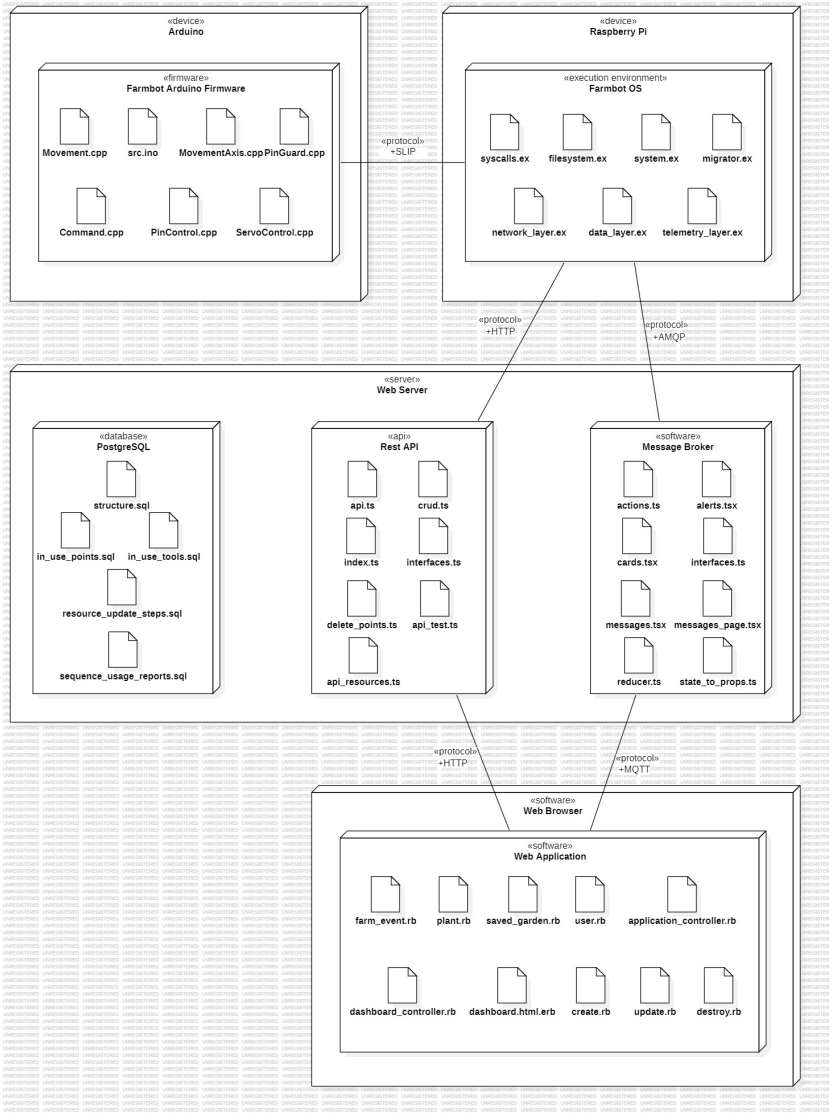

Figure 4.11: Deployment Diagram

The deployment diagram for FarmBot showcases the system's infrastructure and interconnections, depicting the distribution of software components across various hardware devices and the specific protocols used for communication. The diagram is organized into several key areas, each highlighting different aspects of the FarmBot system.

- **Arduino:** Hosts the FarmBot Arduino Firmware, executing various C++ files such as Movement.cpp, Command.cpp, PinControl.cpp, and ServoControl.cpp that manage movement axes, command interpretation, pin control, and servo management, crucial for the physical operations of FarmBot.

- **Raspberry Pi:** Runs FarmBot OS, which includes critical modules like syscall.ex, filesystem.ex, system.ex, and migrator.ex, alongside network_layer.ex, data_layer.ex, and telemetry_layer.ex. These modules handle system calls, file management, data processing, and network communication using HTTP and AMQP protocols to interact with other system components.

- **Web Server:** Divided into three main segments:

  - **Database:** Utilizes PostgreSQL for data storage, handling files like structure.sql and various SQL scripts for updating and managing resources and sequences.

  - **REST API:** Supports operations defined in files like crud.ts, index.ts, and api.ts, which facilitate data handling and interaction with the web application.

  - **Message Broker:** This component functions as the middleware facilitating asynchronous communication across the system. It manages actions, alerts, and message parsing as defined in files like actions.ts, alerts.tsx, and messages.tsx, crucial for efficient state management and event handling across FarmBot's distributed environment.

- **Web Browser:** Represents the client-side of the FarmBot system, hosting the Web Application with Ruby on Rails files like farm_event.rb, plant.rb, user.rb,

which are crucial for rendering the user interface and handling user interactions, data display, and direct management of farming events.

Communication across these components is maintained through various protocols:

- HTTP/HTTPS for secure web interactions,
- AMQP for robust messaging between the Raspberry Pi and the web server,
- MQTT for lightweight messaging particularly useful for IoT devices like FarmBot.

This deployment diagram effectively outlines the operational blueprint of FarmBot, illustrating how various software and hardware components integrate and communicate to create a functional autonomous farming system. This structure ensures that FarmBot can efficiently execute agricultural tasks while providing users with real-time control and monitoring capabilities.

4.5 Design Rationale

Design Rationale for Context View

For the Context View of FarmBot, the decision was made to integrate the system with the OpenFarm.cc API and other external interfaces like GitHub. The rationale was to ensure that FarmBot benefits from a vast, shared knowledge base for plant data and collaborative improvement of its software, which aligns with the open-source nature of the project. Leveraging GitHub fosters an active development community and continuous enhancement of the platform, essential for adapting to the evolving needs of users and technology.

Design Rationale for Functional View

The Functional View of FarmBot divides the system into distinct components, such as the User Interface, System Admin Interface, and interactions with external entities like OpenFarm.cc and GitHub. This separation allows clear representation of responsibilities, improving system coherence and easing both development and future enhancements. Focusing on modular design also supports ease of maintenance and scalability, essential for the diverse applications of FarmBot users, ranging from hobbyists to research institutions.

Design Rationale for Information View

The Information View was designed to organize and manage data related to farming operations in a structured, relational model. This approach facilitates the efficient handling of complex and interrelated agricultural data. By structuring data hierarchically, FarmBot ensures quick access and robust data integrity, making the system reliable and trustworthy for users who rely on it for precision farming.

Design Rationale for Deployment View

In the Deployment View, the decision to deploy key components of the FarmBot system

on scalable servers was driven by the need for high availability and robustness. Such a deployment ensures that FarmBot's services are reliable and continuously available to users worldwide. Moreover, the use of servers enables FarmBot to handle varying loads dynamically, which is critical during peak farming seasons or when integrating with third-party services for extended functionality.

5. Architectural Views for Your Suggestions to Improve the Existing System

5.1 Context View

5.1.1 Stakeholders' uses of this view

The Context View is vital for stakeholders to understand FarmBot's integration with external systems, emphasizing enhanced functionality:

- **Hobbyist Gardeners and Professional Farmers:** These users see how FarmBot's integration with services like the OpenWeather API and the mobile application optimizes farming based on real-time data, facilitating remote management.
- **Educators:** They use the Context View to show students FarmBot's connectivity to technologies such as the OpenWeather API and mobile apps, illustrating practical applications of integrated agricultural technology.
- **Researchers:** Researchers explore how external data sources like the OpenWeather API influence FarmBot's functionality, with the mobile app providing essential real-time control and data access for experiments.
- **Software Developers:** Developers assess how the mobile application and external APIs like OpenWeather interact with FarmBot's core systems, crucial for

developing responsive applications.

- **System Administrators:** Admins monitor FarmBot's performance with external services including the OpenWeather API and mobile app, ensuring reliability and a seamless user experience.
- **Open-Source Contributors and Community Members:** Contributors identify opportunities to enhance FarmBot's capabilities, focusing on the mobile application and third-party service integrations to improve functionality.

Each stakeholder group uses the Context View to effectively support and utilize the FarmBot system, leveraging its connections with external technologies.

5.1.2 Context Diagram

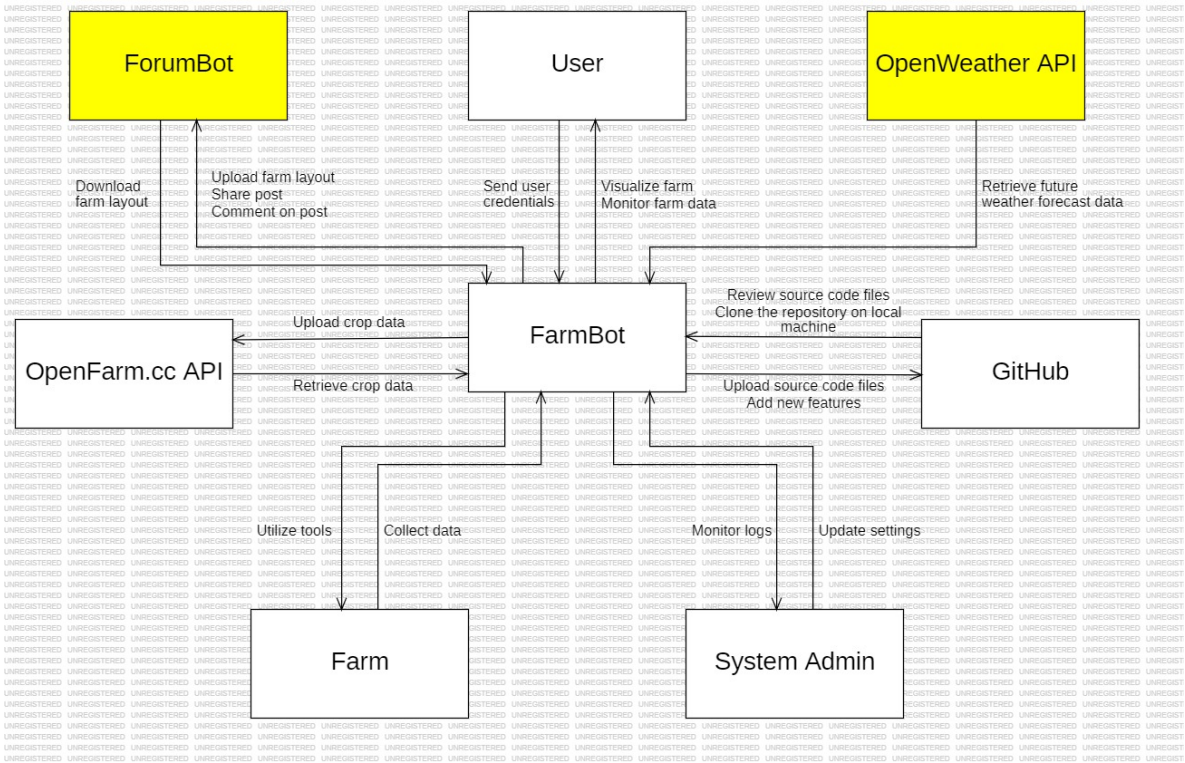

Figure 5.1: Context Diagram for the Improved System

The updated Context Diagram of FarmBot reflects the integration of new features and external interfaces that further enhance its capabilities for autonomous agricultural management. Here's how the system now interacts with new components:

- **ForumBot:** A new addition that facilitates user interaction within the FarmBot community by enabling discussions, knowledge exchange, and the sharing of custom farm layouts and strategies, fostering a collaborative user environment.
- **OpenWeather API:** Newly integrated to provide real-time weather data directly to FarmBot, allowing the system to make immediate adjustments to farming operations based on current weather conditions, enhancing responsiveness to environmental factors.

The revised context diagram illustrates FarmBot's evolution into a more connected, responsive, and user-focused system. These enhancements not only improve FarmBot's operational efficiency but also enrich the user experience and community engagement, ensuring that the system remains at the forefront of technology-driven sustainable agriculture.

5.1.3 External Interfaces

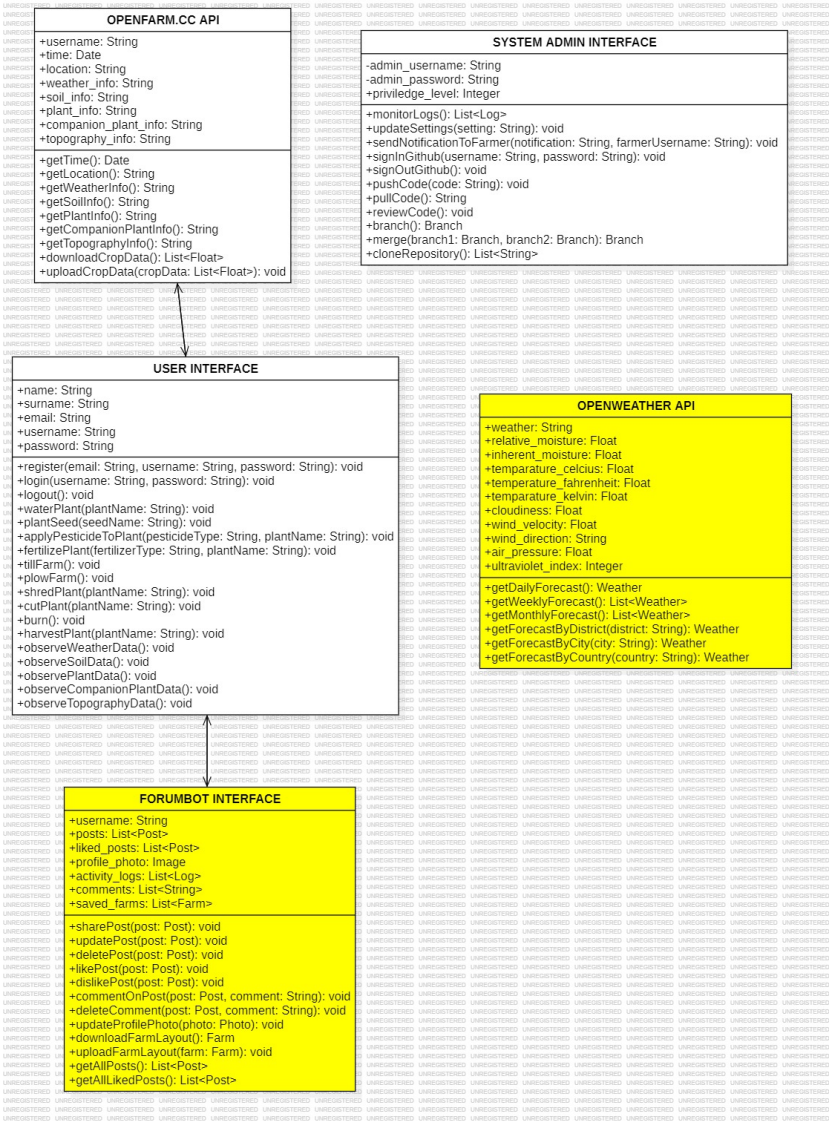

Figure 5.2: External Interfaces Class Diagram for the Improved System

In the enhanced architecture of the FarmBot system, significant developments in the external interfaces include the addition of the OpenWeather API and the ForumBot Interface, which broaden the capabilities of data interaction and community engagement:

- **ForumBot Interface:**
  - **username:** Unique identifier for a user within the ForumBot community.
  - **posts:** A collection of posts made by a user.
  - **liked_posts:** List of posts that the user has liked.
  - **profile_photo:** Image file that represents the user's profile picture.
  - **activity_logs:** Record of user actions and interactions within the community.
  - **comments:** Comments made by the user on various posts.
  - **saved_farms:** Collection of farm layouts or strategies that the user has saved for future reference.
  - **sharePost**(): Allows users to create and share a new post with the community.
  - **updatePost**(): Method to update a post created by the user.
  - **deletePost**(): Deletes a specific post made by the user.
  - **likePost**(): Adds a post to the user's liked posts list.
  - **dislikePost**(): Removes a post from the user's liked posts list.
  - **commentOnPost**(): Adds a comment to a specific post.
  - **deleteComment**(): Removes a specific comment made by the user.
  - **updateProfilePhoto**(): Updates the user's profile photo displayed on the forum.
  - **downloadFarmLayout**(): Enables users to download a specific farm layout from the community.

- **uploadFarmLayout**(): Allows users to upload and share their farm layout with the community.
- **getAllPosts**(): Retrieves all posts made within the ForumBot community.
- **getAllLikedPosts**(): Retrieves all posts that the user has liked.

- **OpenWeather API:**
  - **weather:** General description of the current weather conditions.
  - **relative_moisture:** The moisture content in the air relative to the temperature.
  - **inherent_moisture:** Absolute moisture content in the air regardless of temperature.
  - **temperature_celsius:** Current temperature in degrees Celsius.
  - **temperature_fahrenheit:** Current temperature in degrees Fahrenheit.
  - **temperature_kelvin:** Current temperature in degrees Kelvin.
  - **cloudiness:** Percentage describing the cloud cover.
  - **wind_velocity:** Speed of the wind measured in units per hour.
  - **wind_direction:** Compass direction from which the wind is coming.
  - **air_pressure:** Atmospheric pressure measured in units of pressure (such as pascals or millibars).
  - **ultraviolet_index:** Index indicating the strength of the sun's ultraviolet radiation at the surface.
  - **getDailyForecast**(): Retrieves the weather forecast for the current day.
  - **getWeeklyForecast**(): Retrieves the weather forecast for the current week.
  - **getMonthlyForecast**(): Retrieves the weather forecast for the current month.
  - **getForecastByDistrict**(): Provides weather forecasts specific to a district.

- **getForecastByCity**(): Provides weather forecasts specific to a city.
- **getForecastByCountry**(): Provides weather forecasts specific to a country.

The integration of these interfaces into FarmBot's system enhances the platform's functionality and user engagement. ForumBot enriches the community aspect by allowing more dynamic interactions among users, while the OpenWeather API integrates essential weather data into FarmBot's operational decisions, ensuring that farming strategies are optimized for current and anticipated environmental conditions. These updates not only enrich the user experience but also bolster FarmBot's capability as a comprehensive, data-driven agricultural solution.

5.1.4 Interaction scenarios

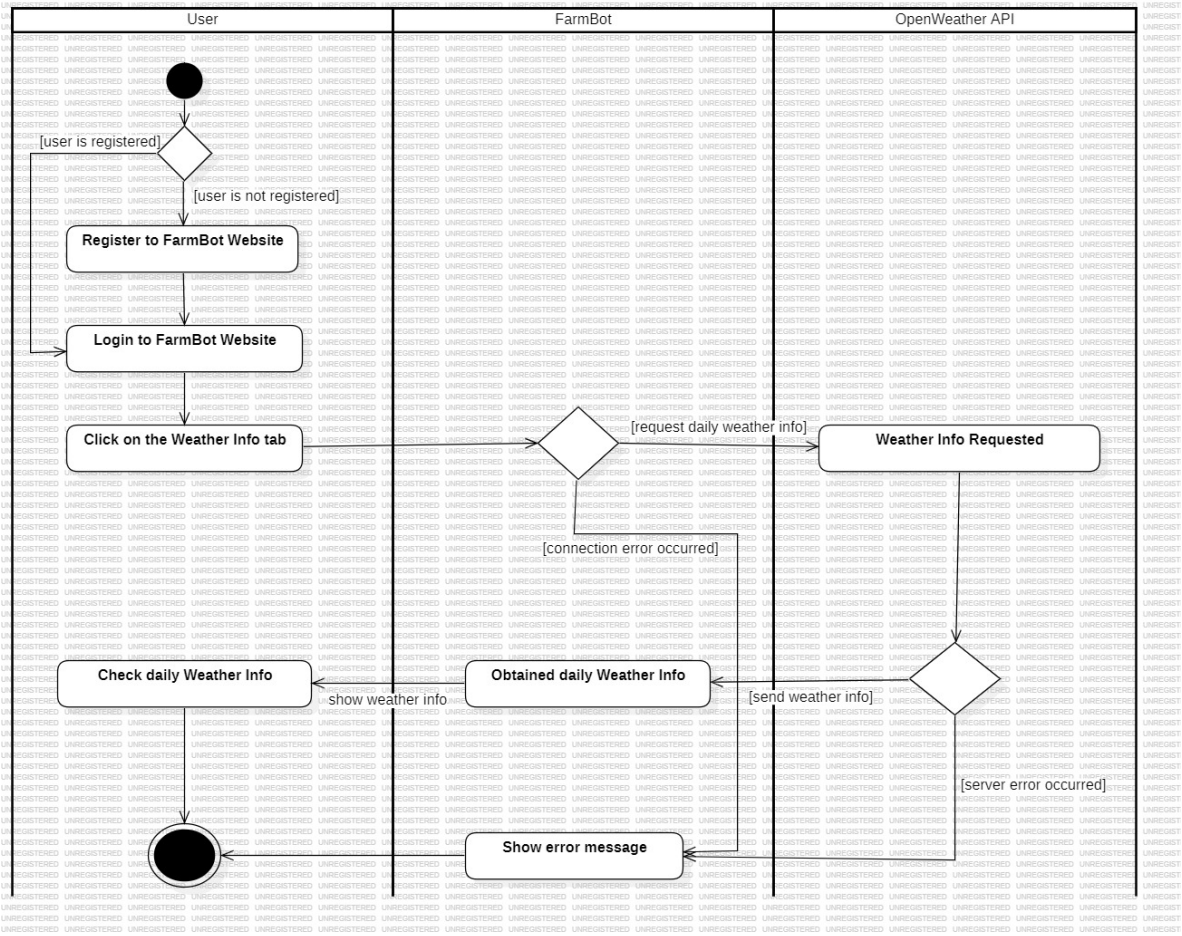

Figure 5.3: Activity Diagram for "Check Daily Wheather Info" Interaction Scenario

5.2 Functional View

5.2.1 Stakeholders' uses of this view

The Functional View offers insights into FarmBot's internal operations, focusing on new features to guide stakeholders in development, usage, and system enhancement:

- **Developers:** Use this view to understand and integrate new features like AI-Driven Weed Identification and the Mobile Application. It helps ensure these functionalities blend seamlessly with FarmBot's existing systems and enhance overall performance.
- **Users (Hobbyist Gardeners, Professional Farmers, Educators):** Access this view to learn how new AI tools improve FarmBot's efficiency and how the mobile application facilitates remote management, enhancing their daily farming and educational activities.
- **System Admins:** Monitor the performance and reliability of new features, particularly the mobile application, to maintain system integrity and address any issues promptly.
- **Researchers:** Explore how AI-driven features can be applied in agricultural research, particularly for precise disease detection and weed management.
- **Contributors (Open-Source Community Users):** Evaluate how their contributions, such as AI algorithms or mobile app enhancements, fit into FarmBot's framework and impact its functionality.

This streamlined Functional View clarifies the roles of new internal features in FarmBot's operations, ensuring stakeholders understand how these advancements contribute to the system's goals of efficiency and user accessibility.

5.2.2 Component Diagram

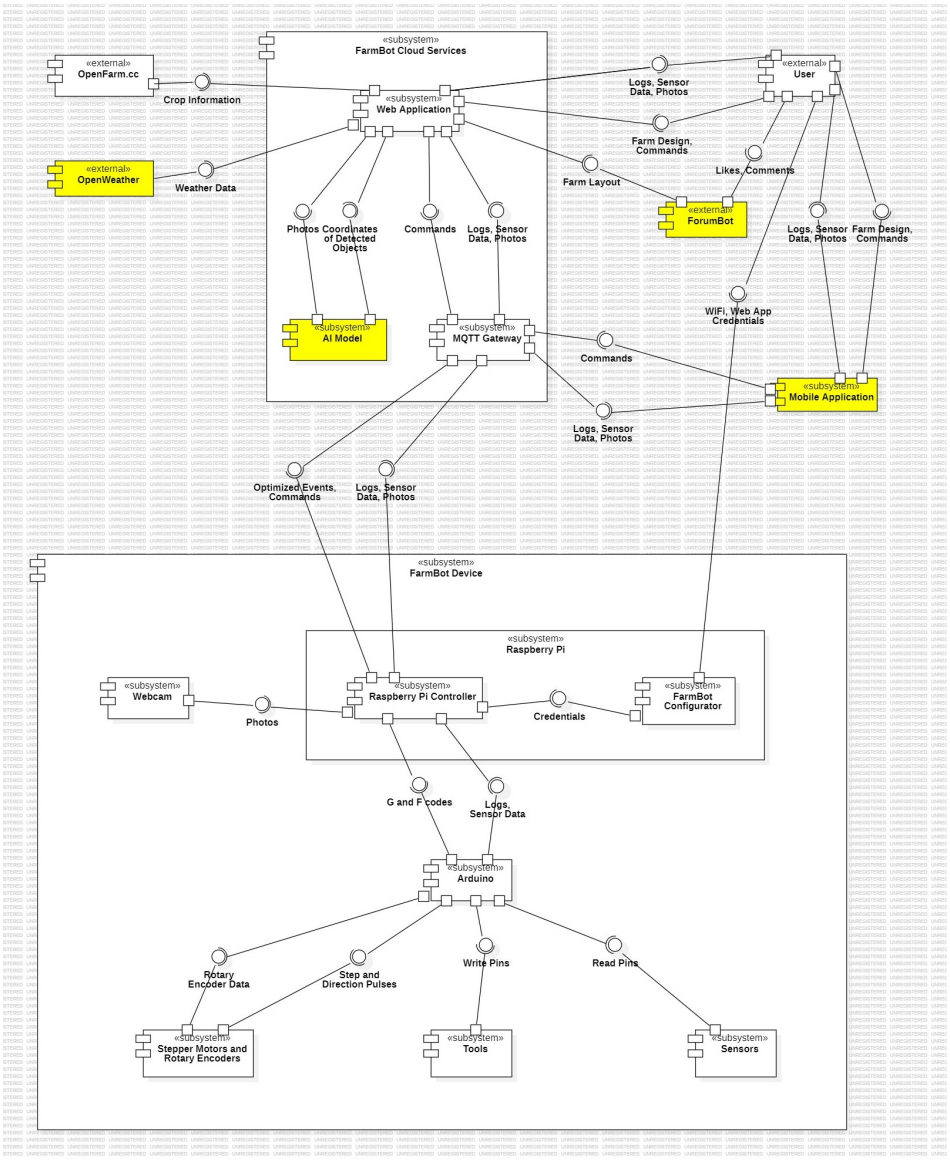

Figure 5.4: Component Diagram for the Improved System

The updated FarmBot system architecture integrates new subsystems and external interfaces, enhancing its capabilities and interaction with external data and user platforms:

- **OpenWeather API:** This integration provides real-time and forecasted weather data crucial for farming decisions. It feeds environmental data directly into the FarmBot Cloud Services, influencing the AI Model's predictive analyses for irrigation and crop management.
- **AI Model:** Positioned within the Cloud Services, the AI Model uses data from OpenWeather and onsite sensors to optimize farming operations. It processes this information to generate actionable insights and commands, which are relayed via the MQTT Gateway to the FarmBot device.
- **ForumBot:** As a new community platform, ForumBot facilitates user interaction by allowing the sharing of farming strategies and insights directly through the Web Application, enhancing community engagement and knowledge exchange.
- **Mobile Application:** This addition extends user interaction capabilities by enabling remote monitoring and management of the FarmBot system via smartphones. It syncs with the Web Application to provide real-time operational control and updates, ensuring a cohesive user experience.

These additions make FarmBot more adaptable, responsive, and user-friendly, enhancing its precision farming capabilities and ensuring it remains a leader in automated agricultural technology.

5.2.3 Internal Interfaces

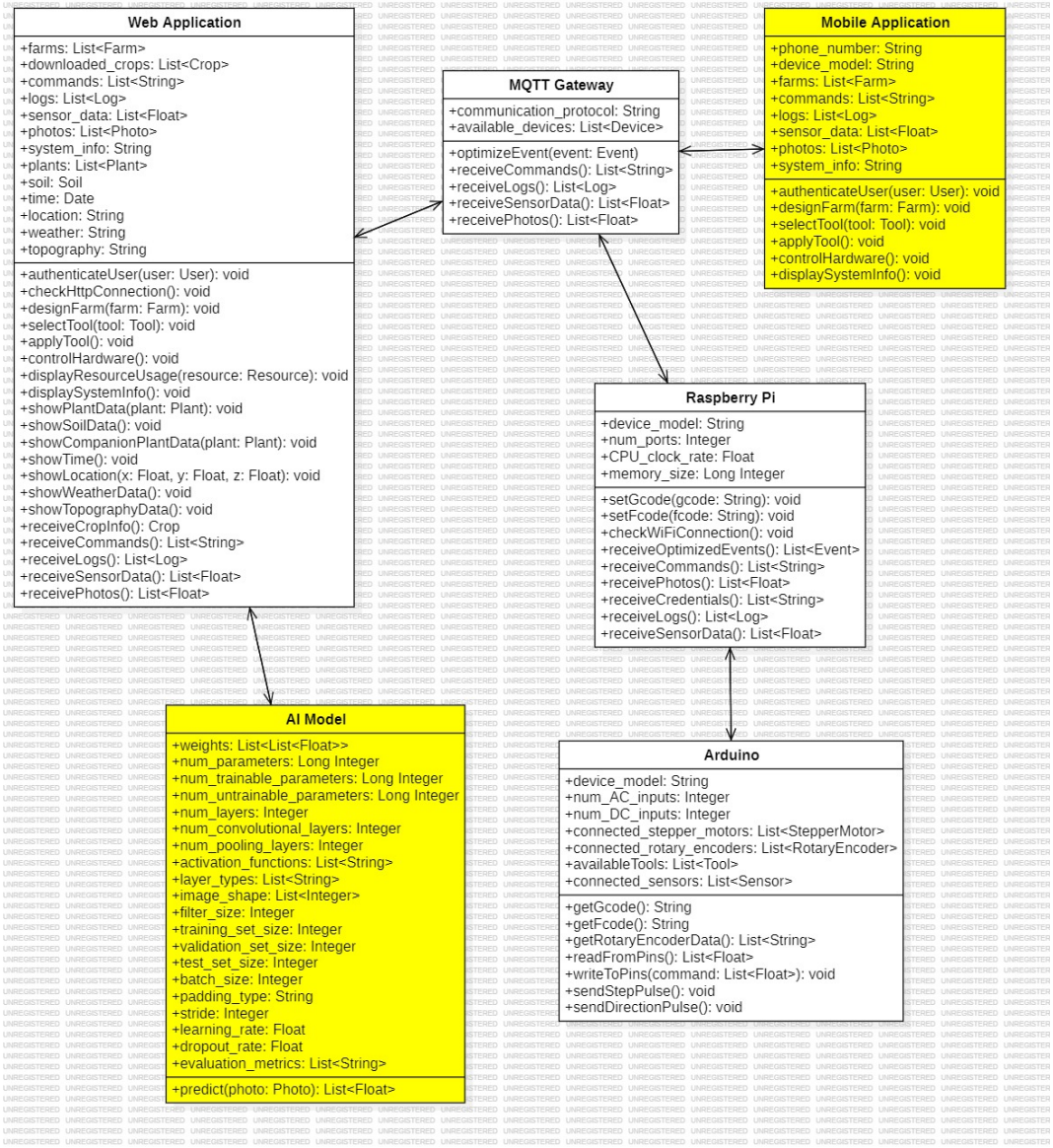

Figure 5.5: Internal Interfaces Class Diagram for the Improved System

The integration of additional internal interfaces significantly enhances the functionality and responsiveness of the FarmBot system. The updated interfaces include new components Mobile Application and the AI Model interfaces:

- **AI Model:**
  - **weights:** The parameters of the neural network that are adjusted during the training process to minimize error.
  - **num_parameters:** Total number of parameters in the AI model.
  - **num_trainable_parameters:** Parameters that can be adjusted during training to improve model accuracy.
  - **num_untrainable_parameters:** Fixed parameters that do not change during the model's training.
  - **num_layers:** The total number of layers in the neural network.
  - **num_convolutional_layers:** Number of layers that use convolutional neural network technology to process pixel data.
  - **num_pooling_layers:** Layers designed to reduce the spatial size of convoluted features to decrease computational load and overfitting.
  - **activation_functions:** Functions like ReLU or sigmoid that introduce nonlinear properties to the model.
  - **layer_types:** Types of layers used in the model, such as dense, dropout, or batch normalization.
  - **image_shape:** The dimensionality of input images that the model expects.
  - **filter_size:** The size of the filter used in convolutional layers.
  - **training_set_size:** The number of samples in the training dataset.
  - **validation_set_size:** The number of samples used to validate the model during training.
  - **test_set_size:** The number of samples in the dataset used to test the model after training.

- **batch_size:** Number of training examples utilized in one iteration.
- **padding_type:** The type of padding used during convolution (e.g., 'same', 'valid').
- **stride:** The stride of the sliding window during convolution operations.
- **learning_rate:** The step size at each iteration while moving toward a minimum of a loss function.
- **dropout_rate:** Probability at which neurons are turned off randomly to prevent overfitting.
- **evaluation_metric:** The metric used to measure the performance of the model during validation and testing.
- **predict**(): Function that applies the trained model to new data to generate predictions based on learned features.

- **Mobile Application:**
  - **phone_number:** The contact number linked to the user account for identity verification and notifications.
  - **device_model:** Specifies the model of the mobile device using the application, useful for tailoring the app's performance to device capabilities.
  - **farms:** A list of farms managed by the user, allowing easy switching and management of multiple locations.
  - **commands:** Stores the list of commands that can be sent from the mobile device to the FarmBot, such as watering or moving.
  - **logs:** Records of operations and system messages that help users track the activity and diagnose issues.
  - **sensor_data:** Real-time data collected from various sensors installed in FarmBot, providing insights into environmental conditions.
  - **photos:** Images captured by FarmBot that are accessible through the mobile application for monitoring crop growth and health.

- **system_info:** Information about the status and health of the FarmBot system, including software version and operational status.
- **authenticateUser**(): Function to verify the identity of the user, ensuring secure access to the app.
- **designFarm**(): Allows users to layout or modify the farm setup directly from their mobile device.
- **selectTool**(): Enables users to choose specific tools for tasks like planting or weeding via the app.
- **applyTool**(): Command to activate the selected tool on the FarmBot.
- **controlHardware**(): Direct control of the FarmBot hardware to execute tasks such as movement and adjustments.
- **displaySystemInfo**(): Displays detailed system status and diagnostics to the user on their mobile device.

5.2.4 Interaction Patterns

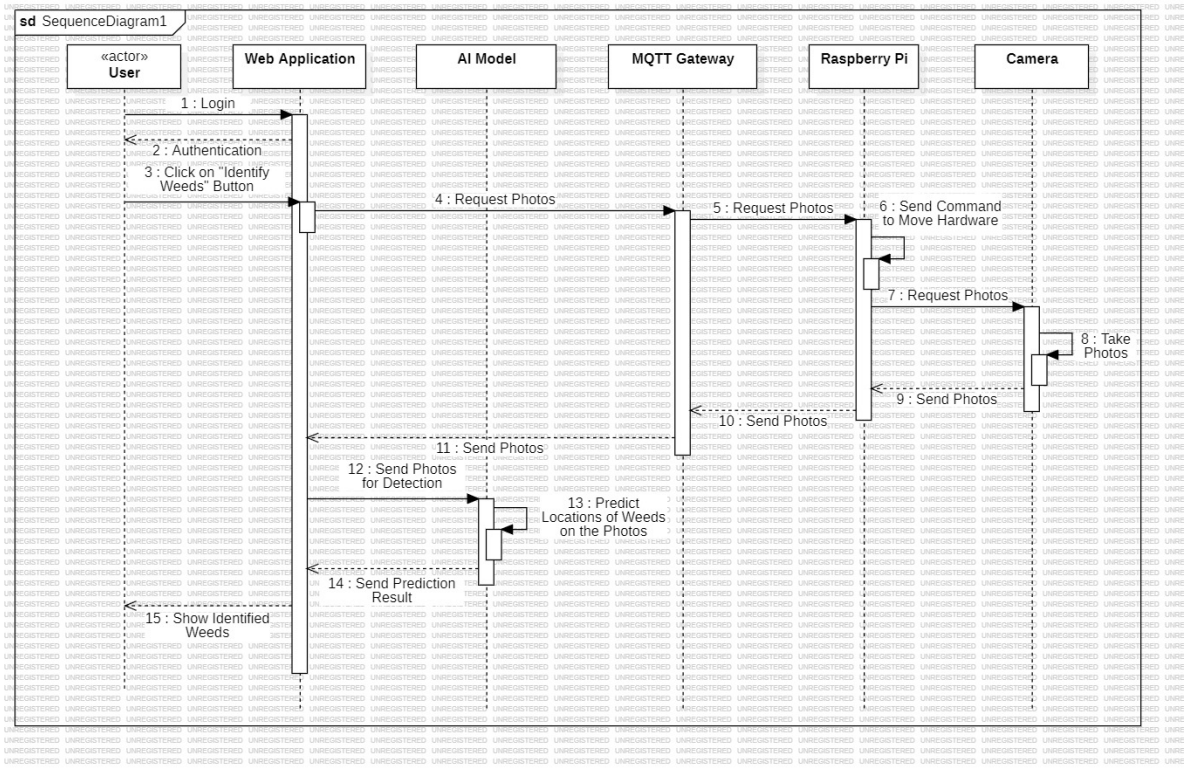

Figure 5.6: Sequence Diagram for "Identify Weeds" Interaction Pattern

5.3 Information View

5.3.1 Stakeholders' uses of this view

The Information View is essential for stakeholders to comprehend and manage how data flows within FarmBot, especially with the addition of new features like AI-Driven Weed Identification and Disease Detection, Mobile Application, ForumBot, and the OpenWeather API. Here's how these new features impact data management:

- **Hobbyist Gardeners and Professional Farmers:** These users look at the Information View to understand how data from the new OpenWeather API and AI-driven features are integrated and utilized. This view helps them see how weather forecasts affect FarmBot's operational decisions and how AI technologies improve weed and disease management, enhancing their farming practices with more precise data.

- **Educators:** They use the Information View to show students how data from external sources like weather APIs and community-generated content from ForumBot are managed and utilized by FarmBot. This helps illustrate the integration of real-world data into automated systems, providing a practical example of data-driven agriculture.

- **Researchers:** Interested in the data specifics for AI models and external data integration, researchers use this view to understand how FarmBot collects, stores, and analyzes data from the OpenWeather API and the AI systems for advanced agricultural studies.

- **Software Developers:** Developers rely on this view to grasp how new data streams from the mobile application and external interfaces like ForumBot and OpenWeather API are structured and stored within FarmBot's database. This understanding is crucial for developing robust features that seamlessly integrate with FarmBot's existing data architecture.

- **System Administrators:** System admins use the Information View to oversee the integration and management of new data sources and ensure the security and integrity of data flowing into FarmBot from the OpenWeather API and ForumBot. They monitor how data from these sources impacts system performance and troubleshoot any issues related to data handling.

- **Open-Source Contributors and Community Members:** These stakeholders examine the Information View to understand how new features like the mobile application and ForumBot alter data structures and flow within FarmBot. Insight into how these features collect and utilize data enables contributors to propose enhancements or develop additional functionalities that align with FarmBot's data models.

The Information View helps stakeholders ensure that the integration and management of data from new internal and external features align with FarmBot's objectives of enhancing efficiency, sustainability, and educational value in automated farming.

5.3.2 Database Class Diagram

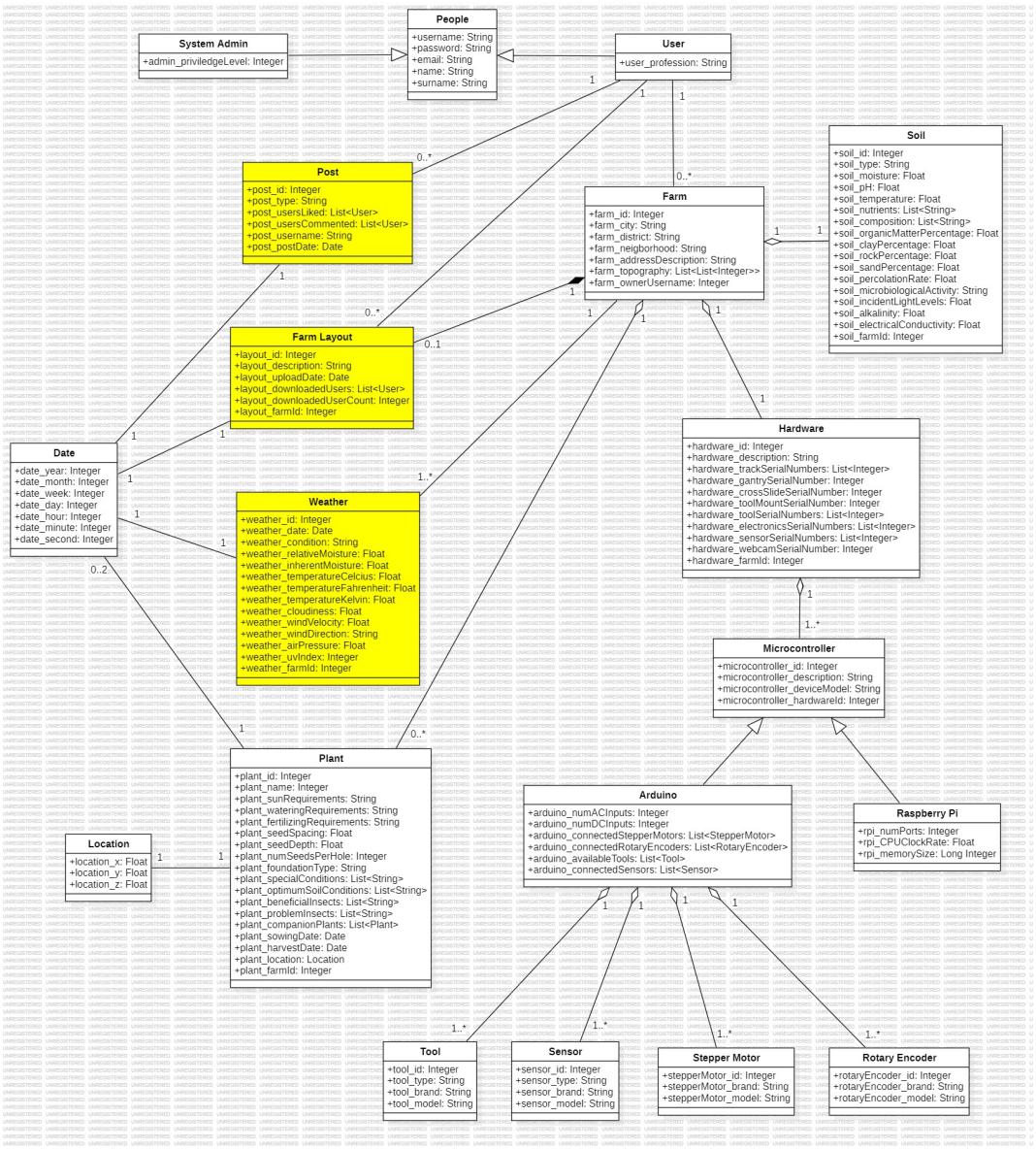

Figure 5.7: Database Class Diagram for the Improved System

5.3.3 Operations on Data

Table 5.1: CRUD Operations for the Improved System

Operation	CRUD Operations
CreatePost	Create: Add a new post entry with details such as type, user comments, username, and post date. Read: - Update: - Delete: -
ReadPost	Create: - Read: Retrieve details of a specific post by post ID. Update: - Delete: -
SeeUsersLiked	Create: - Read: Show all the users that have liked a specific post. Update: - Delete: -
SeeUsersCommented	Create: - Read: Show all the users that have commented on a specific post. Update: - Delete: -

Continued on next page

Table 5.1: CRUD Operations for the Improved System (Continued)

UpdatePost	Create: - Read: - Update: Update details of an existing post, such as type or comments. Delete: -
DeletePost	Create: - Read: - Update: - Delete: Remove a post entry from the database.
RegisterFarmLayout	Create: Register a new farm layout with description, upload date, and associated farm ID. Read: - Update: - Delete: -
RetrieveFarmLayout	Create: - Read: Retrieve details of a specific farm layout by layout ID. Update: - Delete: -
SeeUsersDownloaded	Create: - Read: Display usernames of users who have downloaded a specific farm layout. Update: - Delete: -

Continued on next page

Table 5.1: CRUD Operations for the Improved System (Continued)

UpdateFarmLayout	Create: - Read: - Update: Update description or download count of an existing farm layout. Delete: -
DeleteFarmLayout	Create: - Read: - Update: - Delete: Delete a specific farm layout from the database.
LogWeatherData	Create: Log new weather data entry including date, condition, moisture levels, temperature, wind details, and UV index. Read: - Update: - Delete: -
RetrieveWeatherData	Create: - Read: Retrieve weather data for a specific date or farm ID. Update: - Delete: -

Continued on next page

Table 5.1: CRUD Operations for the Improved System (Continued)

UpdateWeatherData	Create: - Read: - Update: Update existing weather data entries with new measurements or corrections. Delete: -
DeleteWeatherData	Create: - Read: - Update: - Delete: Remove weather data for a specific date and farm from the database.

5.4 Deployment View

5.4.1 Stakeholders' uses of this view

The Deployment View is essential for stakeholders to grasp how FarmBot's new and existing software components are distributed across its hardware and network infrastructure, particularly with the integration of AI-driven features, a mobile application, ForumBot, and the OpenWeather API:

- **Hobbyist Gardeners and Professional Farmers:** They benefit from understanding that features like the mobile application and OpenWeather API enhance FarmBot's reliability and real-time operational capabilities, regardless of local network conditions.
- **Educators:** Use this view to show students the practical application of distributed computing, explaining how features like AI-driven weed identification are implemented across cloud and local devices.
- **Researchers:** Examine how the deployment of AI and external data integrations like the OpenWeather API impacts data accuracy and system responsiveness.
- **Software Developers:** Focus on optimizing code for diverse environments, ensuring efficient operation of features across cloud platforms and mobile devices.
- **System Administrators:** Manage the integration and maintenance of new features, ensuring all components are properly configured for efficient operation.
- **Open-Source Contributors and Community Members:** Refer to this view to ensure their contributions integrate seamlessly with FarmBot's multi-layered architecture, enhancing functionality and compatibility.

This Deployment View helps stakeholders ensure that FarmBot's system is effectively scaled and robust, ready to meet operational demands and maintain high performance.

5.4.2 Deployment Diagram

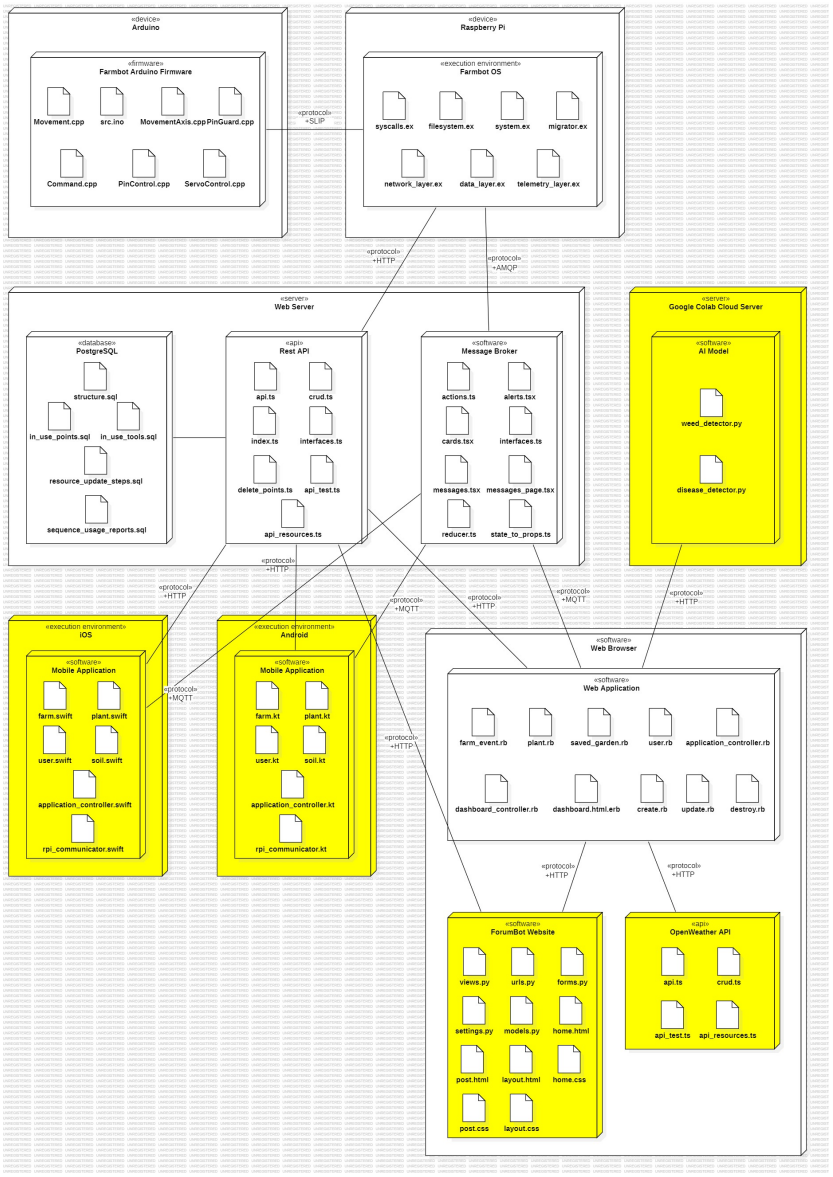

Figure 5.8: Deployment Diagram for the Improved System

The enhanced deployment diagram for FarmBot integrates several new components and services, reflecting the system's evolution and expansion to accommodate advanced functionalities and broader connectivity. These additions represent significant enhancements to the system's capabilities:

- **Google Colab Cloud Server:** This server hosts the AI Model, comprising scripts like weed_detector.py and disease_detector.py. These Python scripts leverage machine learning algorithms to identify weeds and detect plant diseases, enhancing FarmBot's autonomous capabilities. The deployment on Google's Colab platform allows for computational resources and scalability.

- **Mobile Applications for iOS and Android:** The Mobile Application on both iOS and Android platforms, illustrated separately for each operating system, features software components like farm.swift, plant.kt, and user.swift/user.kt. These components are responsible for providing mobile users with real-time access to FarmBot's functionalities, including farm management, plant monitoring, and user account management. The deployment on mobile platforms enhances user accessibility and interaction, utilizing MQTT for real-time communication.

- **ForumBot Website:** Deployed within the Web Browser, the ForumBot Website includes files such as views.py, urls.py, and forms.py which handle the dynamic content generation, URL routing, and form management for user interactions. This platform serves as a community hub where users can share insights, discuss various topics, and exchange farming strategies, fostering a collaborative environment.

- **OpenWeather API:** Also within the Web Browser, the OpenWeather API, with files like api.ts and crud.ts, allows FarmBot to retrieve real-time weather data crucial for farm management decisions. The integration of this API facilitates automated responses to weather changes, optimizing watering schedules and protecting plants from adverse conditions.

These new features are seamlessly integrated into FarmBot's existing deployment framework, ensuring robust performance and enhancing user experience. The use of HTTP/HTTPS for secure web interactions and MQTT for mobile and IoT communications reflects a well-structured deployment strategy that supports both scalability and reliability.

5.5 Design Rationale

Design Rationale for Context View

Integrating the OpenWeather API and introducing ForumBot into the Context View enhances FarmBot's environmental responsiveness and community interaction. The OpenWeather API allows real-time weather data to directly inform farming decisions, improving irrigation and pest management. ForumBot fosters community engagement, enabling users to share insights and strategies, supporting the open-source philosophy and broadening the collaborative knowledge base.

Design Rationale for Functional View

The addition of AI-Driven Weed Identification and Disease Detection and a Mobile Application in the Functional View enhances FarmBot's operational efficiency and user interaction. AI features automate plant health management, reducing the need for manual monitoring, while the mobile app increases accessibility, allowing users to manage and monitor FarmBot remotely.

Design Rationale for Information View

Updating the Information View to include data flows from the OpenWeather API and ForumBot ensures FarmBot can utilize real-time environmental data and user-generated content effectively. This approach enhances FarmBot's adaptability and decision-making processes, maintaining data integrity and scalability.

Design Rationale for Deployment View

The new deployment of cloud-based services for the OpenWeather API and ForumBot on scalable cloud infrastructure ensures high system responsiveness and adaptability. This deployment supports dynamic adjustments to environmental changes and user interactions, maintaining FarmBot's performance and reliability.