Department of Computer Engineering

CENG350 Software Engineering

Software Requirements Specification for FarmBot

Group 80

By

Batuhan Akçan / 2580181

Mert Sönmezer / 2516920

Monday 6th May, 2024

Contents

List of Figures

List of Tables

Revision History

Table 1: Revision History

Version	**Date**	**Description**
1.0.0	27/03/2024	SRS Part-1 Completion
1.1.0	05/04/2024	SRS Final Completion

1. Introduction

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

1.1 Purpose of the System

FarmBot is designed to integrate the advantages of both monocrop and polycrop agriculture systems, aiming to maximize the benefits while minimizing the drawbacks of each. Monocropping is highly efficient from a machinery perspective, which allows for large-scale operations with minimal human labor due to its simplicity and reliance on a single plant species. However, it lacks biological efficiency because it requires extensive inputs like fertilizers, pesticides, and water, and is prone to environmental issues such as topsoil depletion and groundwater pollution. Polycropping, on the other hand, excels in biological efficiency by enhancing ecosystem diversity, which reduces the need for external inputs and creates a more stable and sustainable farming system. Yet, it falls short in machine efficiency. It requires more human labor due to the absence of suitable equipment for managing diverse crops.

FarmBot addresses these challenges by being a scalable farming equipment that can manage a polycrop with similar machine efficiency and minimal labor requirements associated with the monocrop agriculture system. The main purpose of FarmBot is to establish farms that are abundant, resilient, sustainable, and efficient by leveraging the strengths of both monocrop and polycrop systems.

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 System Overview

1.3.1 System Perspective

FarmBot operates as an autonomous system that integrates with various services for comprehensive farm management. Users engage with the Web Application to design their farms and issue commands, which are sent to FarmBot for execution. The system captures a variety of farm-related data including logs, sensor readings, and visual documentation of the farm. For a broader scope of agricultural data, the Web Application interfaces with the OpenFarm.cc API, enriching the user's resources for crop management. Internally, FarmBot utilizes a Raspberry Pi Controller, communicating through MQTT protocol, to manage operations and collect data. This controller then issues precise operational commands to the Arduino Firmware, which engages with an array of sensors and tools to perform essential farming tasks such as seeding, watering, and monitoring environmental conditions. The system admin plays a vital role in maintaining the software's operational integrity, while the farm itself is the physical space where FarmBot's tools are deployed. The simplicity of the user experience is thanks to the sophisticated backend processes that enable seamless operation and real-time data exchange, ensuring that the user remains at the forefront of the agricultural management experience.

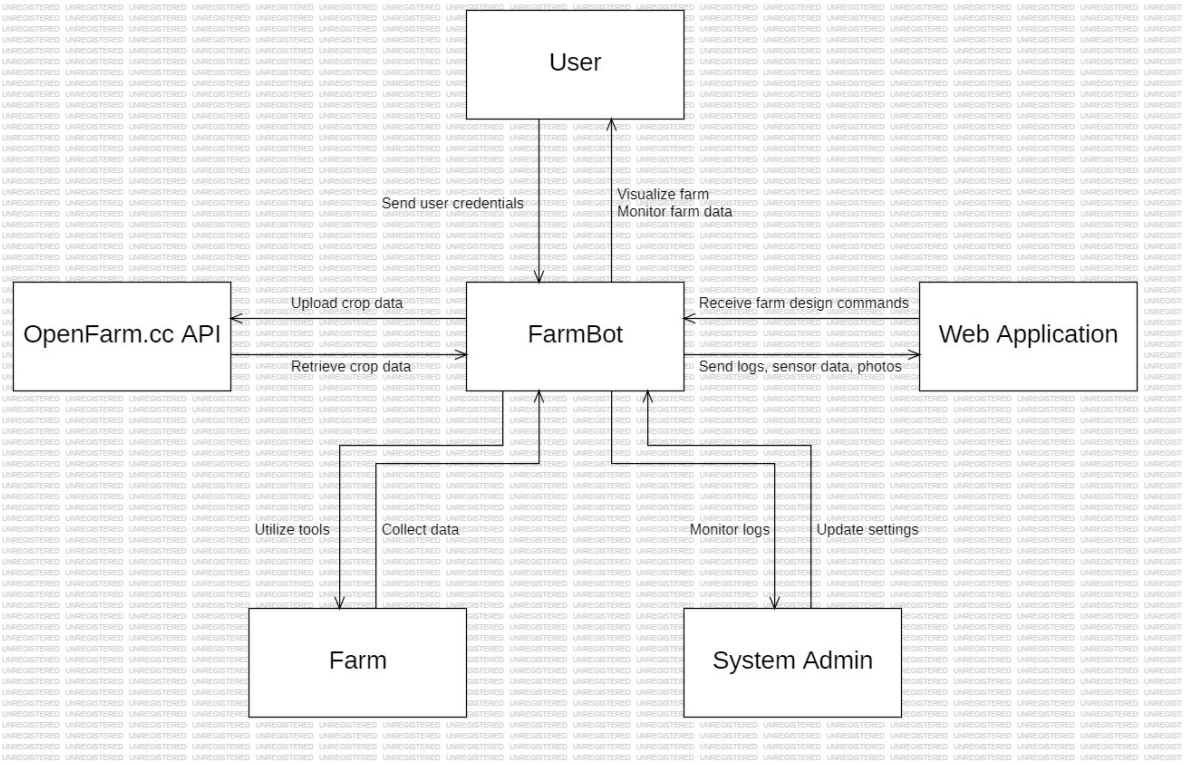

Figure 1.1: System Context Diagram

1.3.1.1 System Interfaces

- **Decision Support System:** The Decision Support System within FarmBot is a key algorithmic component designed to optimize farm management through data-driven decisions. It analyzes various data streams, such as weather patterns and soil properties, to determine the best settings for farming operations like seed spacing, watering, and planting schedules. A high-level overview in Figure 1.2 shows the Decision Support System combining data streams to make informed decisions for each FarmBot operation, significantly enhancing efficiency and productivity in farm management.

- **Data Management System:** Data Management System is a critical interface in FarmBot, focusing on the collection, storage, and utilization of diverse data types on an SQL database to optimize farming practices. Key components include:

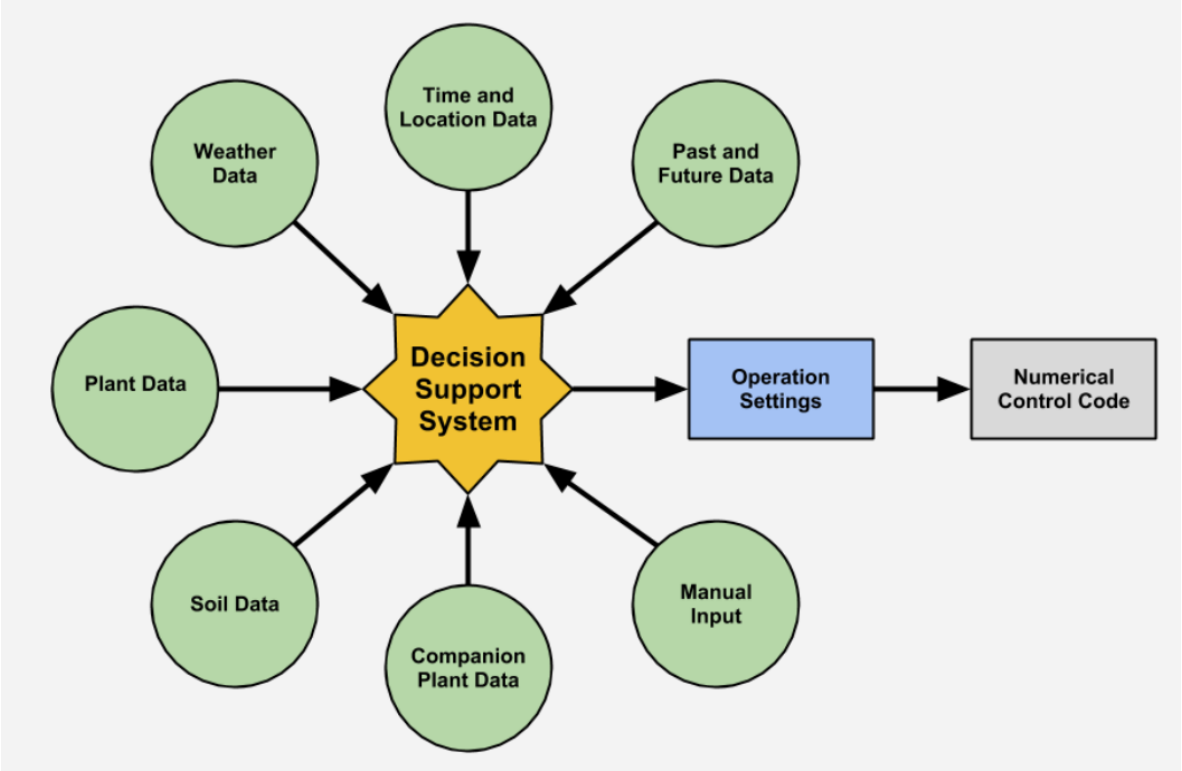

Figure 1.2: Decision Support System

  - **Plant Data:** Covers specifics like sun requirements, seed spacing, and watering needs for optimal growth conditions.
  - **Soil Data:** Involves systematic collection of soil characteristics (moisture, nutrients, composition) for precise soil management.
  - **Companion Plant Data:** Uses information on plant relationships to maximize polycropping benefits, such as pest control and nitrogen fixation.
  - **Time and Location Data:** Enables FarmBot to perform operations based on current time and geographical positioning, optimizing planting and watering schedules.
  - **Weather Data:** Integrates forecasts to make informed decisions on watering and protects crops from adverse weather conditions.
  - **Topography:** Assists in plant placement and water flow management by analyzing land contours and sunlight exposure.
  - **Past and Future Data:** Utilizes historical farm data and future plans for crop rotation and soil improvement strategies.
  - **Manual Input Data:** Allows for the customization of farming operations based on farmer knowledge and unique environmental factors.

- **RESTful JSON API:** The FarmBot WebApp uses RESTful JSON API, crucial for enabling efficient communication across the software's various components. This API plays a pivotal role in managing data flow, including user account details, farm designs, sequences, authorization tokens, and other essential resources. By leveraging JSON for data formatting, the API ensures streamlined and lightweight data exchange, enhancing web application performance. This RESTful approach facilitates a dynamic and interactive user experience, allowing for seamless operation and customization of the FarmBot system. The API's design, rooted in open-source and cloud-based technologies, supports the flexible and accessible management of automated farming operations, making it a fundamental element of the FarmBot WebApp's infrastructure.

- **Webcam:** The webcam interface in FarmBot captures images of the farm, allowing for visual monitoring and progress tracking. These photographs are sent to the Raspberry Pi microcontroller, providing users with up-to-date visual data accessible via the FarmBot WebApp, aiding in both immediate and long-term farm management and planning.

1.3.1.2 User Interfaces

The FarmBot WebApp's user interface is designed to be engaging, intuitive, and efficient. It allows users straightforward access to all software functionalities. The interface employs a tabbed layout, with each tab featuring multiple panes for specific user actions and needs.

- **Register & Login Tab:** The Register & Login Tab serves as the entry point to the FarmBot WebApp, requiring users to authenticate via email and password, as shown on the Figure 1.3.

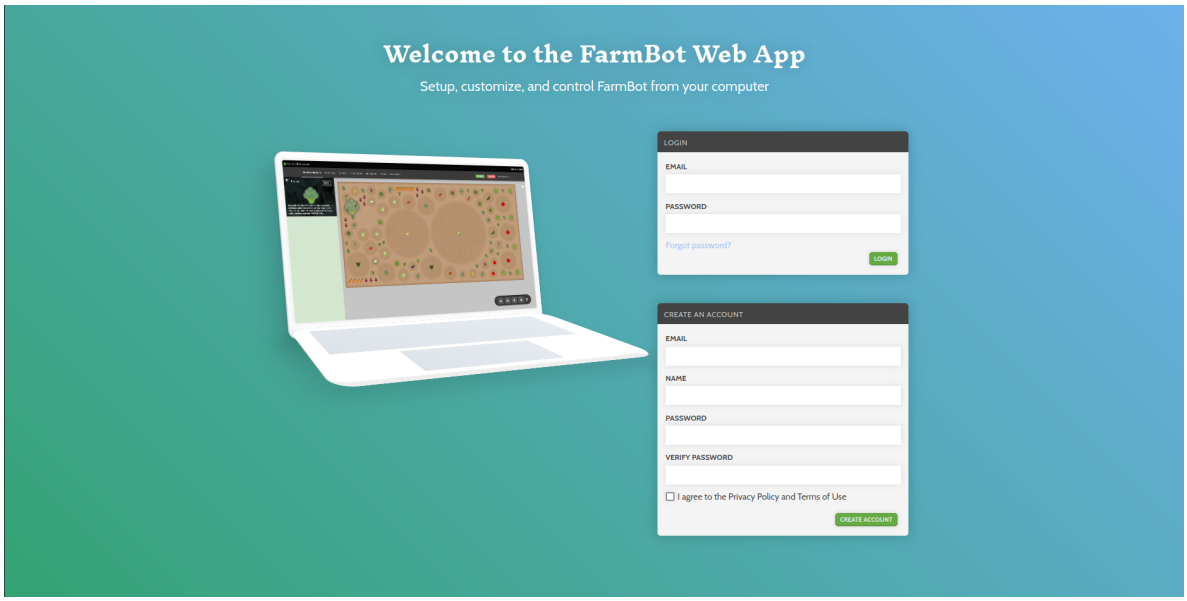

Figure 1.3: Register & Login Tab

- **Dashboard Tab:** Overview of this tab can be seen in Figure 1.4. This tab pro-

vides a comprehensive overview of the FarmBot system's operations and statistics, divided into various panes for detailed insights:

- **Hardware Information Pane:** Users can select and manage hardware components through dropdown menus, enabling easy configuration of the FarmBot system's physical setup.
- **Resource Usage Pane:** An interactive graph displays the consumption of resources (electricity, water, seeds, etc.) over time, with breakdowns and cost analysis based on user input.
- **System Information and Control Pane:** Displays general system information and manual control options for immediate adjustments.
- **Financial Analysis Pane:** Presents an overview of the operation's financial aspects, including cost and revenue projections.

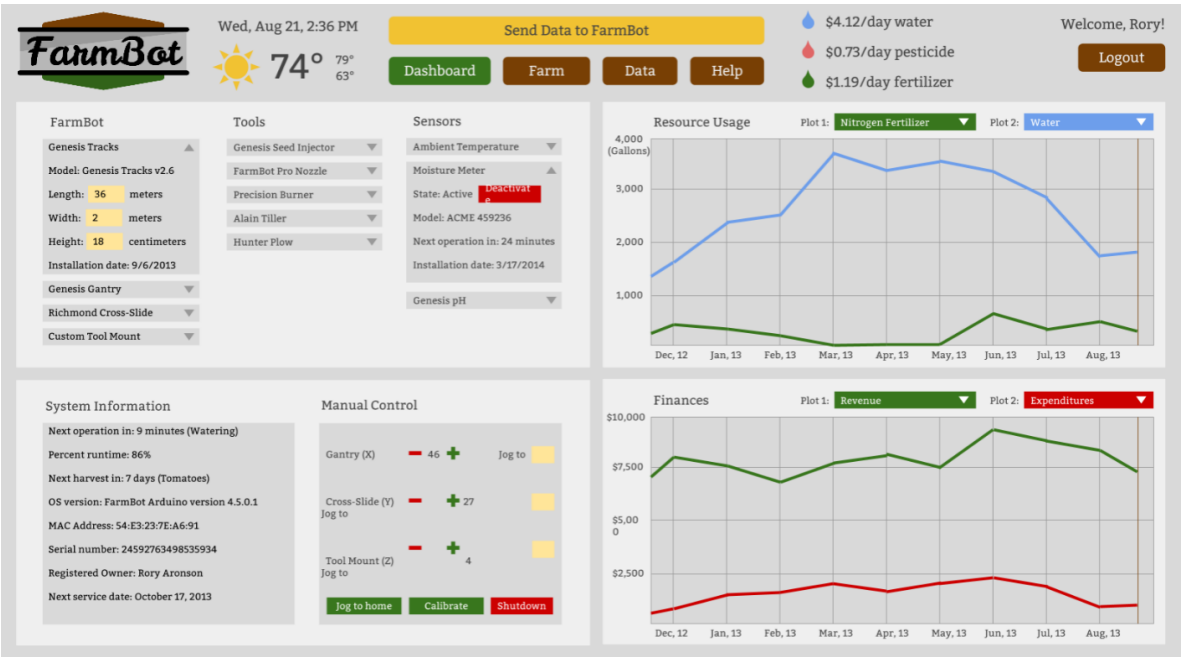

Figure 1.4: Dashboard Tab

- **Farm Tab:** The Farm Tab, which can be seen in Figure 1.5, is central to planning and executing farming operations, consisting of three main sections:

- **Plants and Operations Toolbox:** Enables users to select and customize plantings and operations.
- **Farm Map:** An interactive, zoomable map of the farm layout where users can arrange and modify plantings and operations.
- **Operations Agenda:** Lists scheduled operations with a calendar view, allowing users to plan and adjust farming activities.

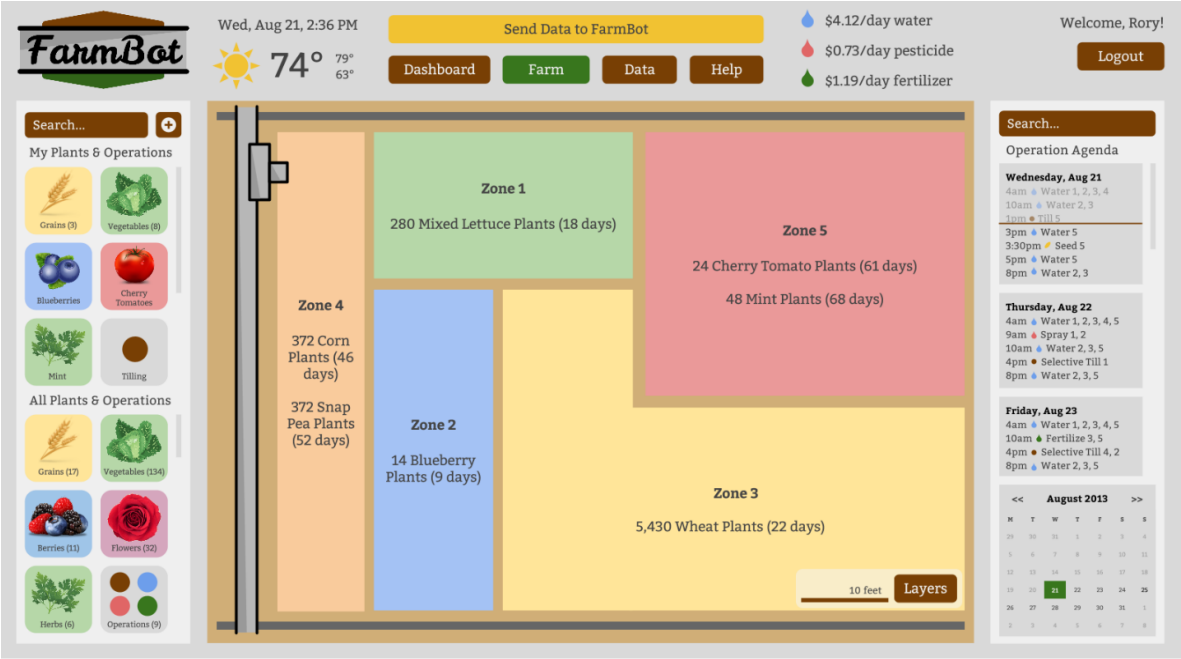

Figure 1.5: Farm Tab

- **Data Tab:** Utilizes FarmBot hardware for detailed data collection. Enables the creation of data maps that inform decision-making processes. These maps highlight areas needing adjustments (e.g., watering, soil amendments) and provide insights into soil moisture, temperature, and other critical factors.

1.3.1.3 Hardware Interfaces

- **Microcontrollers**
  - **Raspberry Pi:** The Raspberry Pi acts as the brain of the FarmBot system, handling lightweight processes generated by the Decision Support System.

It communicates with the Arduino to send G-Codes and F-Codes for the execution of various operations and receives feedback on the operations performed.

  - **Arduino:** The Arduino microcontroller is tasked with receiving G-Codes and F-Codes from the Raspberry Pi, executing these commands, and reporting the outcomes back to the Raspberry Pi. It directly controls the mechanical components of the FarmBot for precise operation execution.

- **Other Hardware Interfaces**

  FarmBot's hardware closely resembles that of 3D printers and CNC milling machines, comprising components like tracks, gantry, cross-slide, and tool mounts that facilitate movement and operation execution in three dimensions (X, Y, and Z). Hardware overview can be seen in Figure 1.6. The system's design allows for scalability in all dimensions, ensuring FarmBot's adaptability to various farm sizes and operational needs. Key hardware components include:

  - **Tracks:** Fixed rails that provide precision and support for the gantry, allowing for highly accurate positioning.
  - **Gantry:** Bridges the tracks and moves along the X-direction, serving as a base for the Y-direction movement of the cross-slide and Z-direction movement of the tool mounts.
  - **Cross-Slide:** Moves across the gantry in the Y-direction, enabling the FarmBot to perform operations anywhere within the XY plane.
  - **Tool Mounts:** Attached to the cross-slide, facilitates Z-direction movement and serves as the point of attachment for various tools required for farming operations.
  - **Tools:** Customized agricultural tools adapted for use with FarmBot, including seed injectors, watering nozzles, sensors, and plows, among others.
  - **Electronics:** Comprise motors, servos, solenoids, valves, and sensors controlled by the microcontrollers, essential for the operation of FarmBot.

– **Sensors:** Utilized for data collection to inform decisions about farm setup and operations, enhancing the "Smart Farming" capabilities of FarmBot.

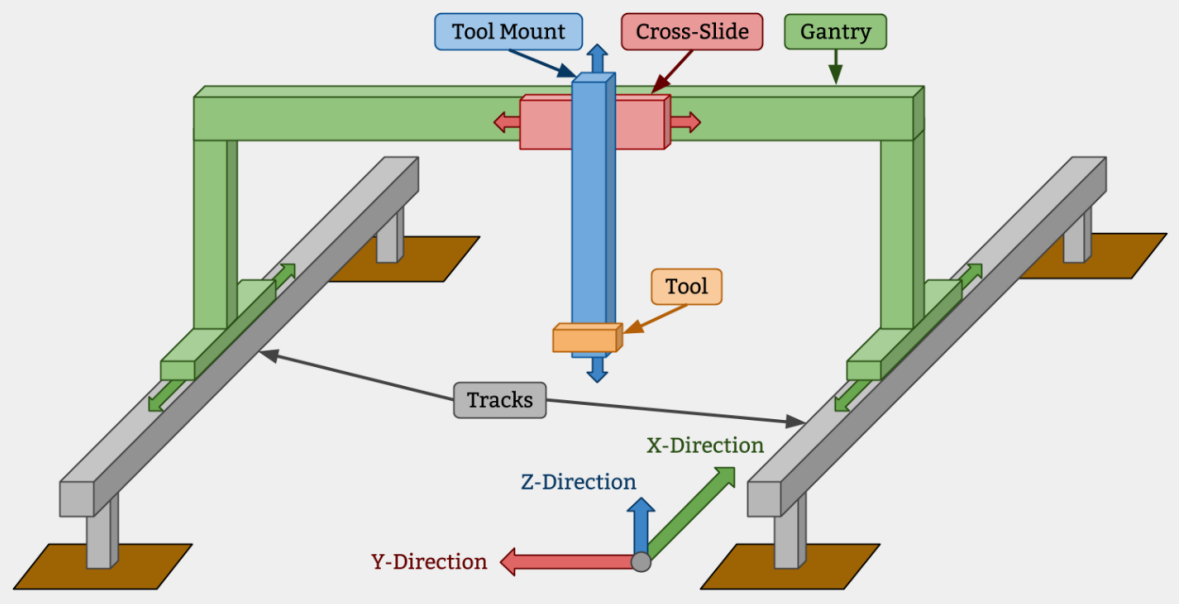

Figure 1.6: Hardware Overview

1.3.1.4 Software Interfaces

- **FarmBot WebApp:** Provides a central interface for users to manage and program their FarmBot. It includes a RESTful JSON API for data management and a Dockerized MQTT server for real-time device communication. The web app is designed with a tabbed interface for easy navigation, allowing users to oversee FarmBot operations, design farm layouts, and analyze data through interactive maps and panels.

- **FarmBot OS:** Custom-designed for the Raspberry Pi, facilitates communication and operation capabilities. It connects with the web application via hosted MQTT gateway for synchronization tasks, including downloading sequences and farm designs, and uploading logs and sensor data. Additionally, it handles real-time commands, and sends them to the Arduino for command execution and

data collection, and manages images with compatibility for USB or Raspberry Pi cameras.

- **Microcontroller Software:** Embedded within FarmBot's hardware, the microcontroller software interprets numerical codes from the backend, manages hardware operations like sensor readings and motor movements, and communicates real-time data back to the cloud. Optimized for efficiency, it conducts complex computational tasks remotely, relying on cloud-based decision support systems to process data and send simplified operational instructions to FarmBot, similar to the G-code used in CNC machining and 3D printing. This setup emphasizes a division of labor, with heavy processing offloaded to the cloud and the microcontroller focusing on direct hardware control.

1.3.1.5 Communication Interfaces

The communication interfaces of FarmBot facilitate secure and efficient interaction between the user, the FarmBot WebApp, and the FarmBot hardware, utilizing:

- **RPC:** This mechanism is primarily employed for user authentication, enabling secure connections to the FarmBot WebApp. It allows the system to perform actions on behalf of the user after verifying their identity, ensuring that only authorized users can access and control the FarmBot operations.

- **HTTP/HTTPS:** The backbone of communication within the FarmBot WebApp, HTTP/HTTPS is utilized for all web-based interactions. HTTPS, the secure version of HTTP, ensures that data transferred between the user's browser and the FarmBot WebApp is encrypted, providing a protected channel for the transmission of sensitive information such as user account details, farm designs, and operational data.

- **MQTT:** MQTT in FarmBot enables real-time messaging between the web application and Raspberry Pi microcontroller, crucial for immediate command execution and data exchange. WebApp uses MQTT, a lightweight protocol ideal for

IoT communications, managed through a broker for efficient message distribution.

- **WiFi:** A link for configuring the FarmBot system, WiFi enables the transmission of user credentials to the FarmBot Configurator. This interface is essential for initializing communication between the user and the Raspberry Pi, setting up the device with the necessary access permissions for operation.
- **USB:** The USB interface in FarmBot enables data transfer between the Arduino and Raspberry Pi, essential for synchronized operations and task execution. This direct connection via USB ports ensures continuous communication, critical for the system's precision and functionality.
- **Ethernet:** The Ethernet port on the Raspberry Pi provides an optional wired network connection for FarmBot, offering an alternative to WiFi. This feature allows users to prefer a more stable and secure connection, accommodating environments where WiFi reliability may be a concern, thus enhancing the system's connectivity flexibility.

1.3.1.6 Memory Constraints

Since the FarmBot application is cloud-based, it does not face any significant memory constraints. The extensive farm data is managed and stored on the cloud, ensuring that the system's performance remains unaffected by the volume of data. This setup allows for scalability and efficient data handling without burdening local resources.

1.3.1.7 Operations

FarmBot operations can be divided into:

- **User Operations:**
  - Design farm layouts
  - Schedule tasks
  - Send manual control commands

  - Observe logs, sensor data and photos
  - Monitor performance

- **System Operations:**
  - Automatically execute scheduled tasks
  - Regularly backup data such as user accounts, farm designs, and operational histories
  - Recover data in the case of system failures

1.3.2 System Functions

Table 1.1: System Functions

Function	**Summary**
Design Farm Layouts	Users can create and modify the spatial arrangement of crops and infrastructure on their digital farm using the WebApp.
Schedule Tasks	Users have the capability to plan and set specific times for FarmBot to perform agricultural activities such as planting, watering, and harvesting.
Send Manual Control Commands	This operation allows users to directly command FarmBot to perform immediate actions, bypassing automated schedules for tasks like troubleshooting or specific, non-routine interventions.

Continued on next page

Table 1.1: System Functions (Continued)

Observe Logs, Sensor Data, and Photos	Users can access detailed records of FarmBot's operations, environmental sensor readings, and visual documentation captured by the device, enabling monitoring and decision-making based on real-time and historical data.
Monitor Performance	This involves users tracking the effectiveness and efficiency of FarmBot's operations over time, including its adherence to schedules and the success rate of completed tasks.
Automatically Execute Scheduled Tasks	FarmBot autonomously carries out agricultural operations based on the schedules set by the user, operating independently from the user during these periods.
Regularly Backup Data	The system periodically saves copies of crucial data such as user accounts, farm designs, and logs of operational history to cloud storage, safeguarding against data loss.
Recover Data in Case of System Failures	In the event of hardware or software malfunctions, the system can restore previously backed-up data to return to normal operations with minimal disruption.

1.3.3 Stakeholder Characteristics

The FarmBot project serves diverse user categories, including hobbyist gardeners, professional farmers, educators, and researchers.

- **Hobbyist gardeners and professional farmers:** Primarily use FarmBot for its automated farming capabilities, requiring minimal technical skills thanks to its user-friendly interface.
- **Educators:** Employ FarmBot as a practical tool to demonstrate the integration of technology in agriculture, necessitating a good understanding of the system for effective teaching.
- **Researchers:** Focus on using FarmBot for studies in sustainable farming and crop optimization. They need to have a strong background in agricultural science and research methodologies.

1.3.4 Limitations

- **Regulatory Policies:** FarmBot is an open source hardware and software project. Therefore the project files and codes are accessible for everyone.
- **Hardware Limitations:** The precision and efficiency of FarmBot are dependent on the performance of its hardware components, such as motors, sensors, and microcontrollers. These components must function without delay to maintain operational accuracy and reliability.
- **Interfaces to Other Applications:** FarmBot's functionality is enhanced through compatibility with various software applications, including cloud-based services and data analytics tools.
- **Parallel Operation:** FarmBot is designed to perform multiple operations simultaneously, such as watering and collecting the sensor data. This requires robust system design to manage concurrent tasks effectively.

- **Audit Functions:** While FarmBot collects extensive operational data, its audit capabilities are designed to focus on system performance and user actions, potentially limiting comprehensive auditing of all system interactions.
- **Control Functions:** FarmBot relies on both automated sequences and user-initiated commands for operation.
- **Higher-order Language Requirements:** The system's software is developed using high-level programming languages, such as TypeScript, Ruby, Elixir, Python, and C++. This ensures ease of maintenance, updates, and scalability.
- **Signal Handshake Protocols:** Communication between FarmBot's components, including the Raspberry Pi and Arduino microcontrollers, or web app and hardware utilizes protocols to ensure data integrity and synchronization.
- **Quality Requirements:** The reliability and durability of FarmBot are critical. The system meets agricultural productivity and efficiency standards.
- **Criticality of the Application:** The FarmBot is not a critical system. It will not cause any huge impact if it fails.
- **Safety and Security Considerations:** Given its internet connectivity, FarmBot contains security measures to protect user data and system integrity, although being open-source may expose it to potential vulnerabilities that require ongoing attention.
- **Physical/Mental Considerations:** FarmBot is designed to be accessible to users with varying levels of physical ability and technical expertise.
- **Limitations from Other Systems:** FarmBot's performance and capabilities may be influenced by the limitations of integrated systems and platforms, including real-time data processing and cloud storage services.

1.4 Definitions

Table 1.2: Definitions

SRS	A software requirements specification (SRS) is a description of a software system to be developed.
SaaS	Software as a service (SaaS) is a software licensing and delivery model in which software is licensed on a subscription basis and is centrally hosted.
MQTT	Message Queue Telemetry Transport (MQTT) is a lightweight, publish-subscribe, machine to machine network protocol for message queue/message queuing service.
WiFi	Wireless Fidelity (WiFi) is a family of wireless network protocols based on the IEEE 802.11 family of standards, which are commonly used for local area networking of devices and Internet access, allowing nearby digital devices to exchange data by radio waves.
G-Code	Geometry code (G-code) is the most widely used computer numerical control (CNC) and 3D printing programming language.

Continued on next page

Table 1.2: Definitions (Continued)

WebApp	A web application (web app) is application software that is accessed using a web browser. Web applications are delivered on the World Wide Web to users with an active network connection.
DSS	A decision support system (DSS) is an information system that supports business or organizational decision-making activities.
SQL	Structured Query Language (SQL) is a domain-specific language used to manage data, especially in a relational database management system (RDBMS).
REST	Representational State Transfer (REST) is a software architectural style that was created to guide the design and development of the architecture for the World Wide Web.
JSON	JavaScript Object Notation (JSON) is an open standard file format and data interchange format that uses human-readable text to store and transmit data objects consisting of attribute–value pairs and arrays (or other serializable values).

Continued on next page

Table 1.2: Definitions (Continued)

API	An application programming interface (API) is a way for two or more computer programs or components to communicate with each other.
CNC	Computer Numerical Control (CNC) is the automated control of tools by means of a computer. It is used to operate tools such as drills, lathes, mills, grinders, routers and 3D printers.
USB	Universal Serial Bus (USB) is an industry standard that allows data exchange and delivery of power between many various types of electronics.
RPC	In distributed computing, a remote procedure call (RPC) is when a computer program causes a procedure (subroutine) to execute in a different address space (commonly on another computer on a shared network), which is written as if it were a normal (local) procedure call, without the programmer explicitly writing the details for the remote interaction.

Continued on next page

Table 1.2: Definitions (Continued)

HTTP	The Hypertext Transfer Protocol (HTTP) is an application layer protocol in the Internet protocol suite model for distributed, collaborative, hypermedia information systems.
HTTPS	Hypertext Transfer Protocol Secure (HTTPS) is an extension of the Hypertext Transfer Protocol (HTTP), and it uses encryption for secure communication over a computer network, and is widely used on the Internet.
IoT	The Internet of things (IoT) describes devices with sensors, processing ability, software and other technologies that connect and exchange data with other devices and systems over the Internet or other communications networks.

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. Specific Requirements

3.1 External Interfaces

- **Web Application Interface:** Handles user interactions with the FarmBot via a web platform. Functions include user authentication, data retrieval (such as sensor and hardware information), displaying resource usage, system information, and scheduling tasks on the FarmBot.
- **User Interface:** Direct interface for the user to interact with FarmBot's functions. It includes personal identification details and commands for various farming operations like watering, planting, fertilizing, and harvesting.
- **System Admin Interface:** Tailored for administrators to maintain the FarmBot system, providing access to system configurations, user management, and oversight of operational logs and updates.
- **OpenFarm.cc API:** Connects with the OpenFarm database, offering users extensive crop data and farming knowledge exchange.

Each interface serves a distinct role, ensuring the FarmBot ecosystem is accessible and manageable by different stakeholders, from end-users to developers and community contributors.

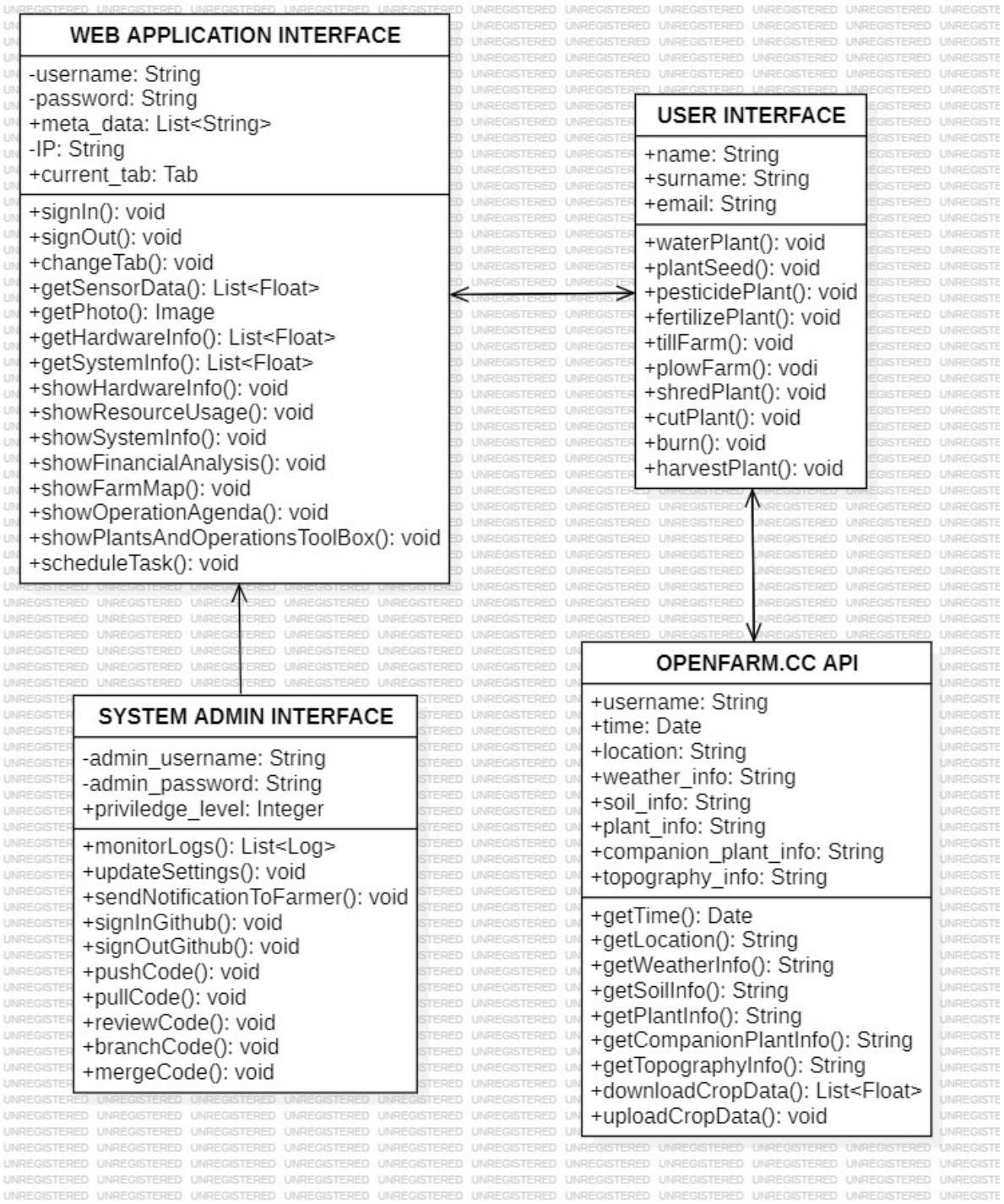

Figure 3.1: External Interfaces

3.2 Functions

The use case diagram showcases ten distinct scenarios, including four actors. On the diagram's left-hand side, the primary actors, namely the user and system admin, are positioned, while on the right-hand side, the secondary actors, which include the Web Application and OpenFarm.cc API, are positioned. The use case diagrams are supported with description tables. Additionally, the use cases titled "Control Hardware," "Monitor Farm Data," and "Store Farm Data" are further explained using an activity diagram, a sequence diagram, and a state diagram, respectively.

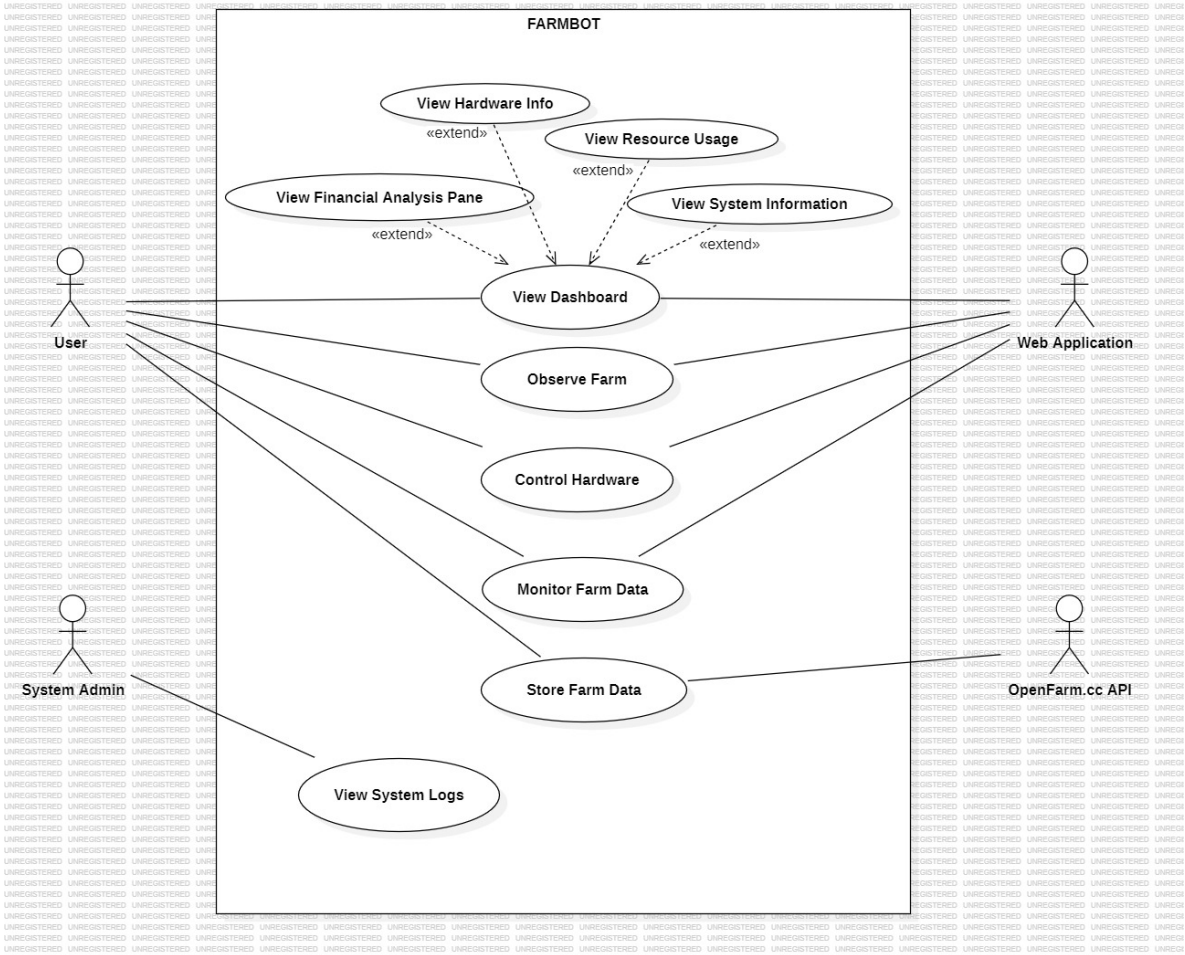

Figure 3.2: Use Case Diagram

Table 3.1: Tabular Description of the **View Dashboard** Use Case

Use Case Name	View Dashboard
Actors	User, Web Application
Description	The user views the dashboard containing various information from his/her farm.
Preconditions	The user must be logged into the FarmBot Web Application.
Data	System, Resource, Hardware, and Financial information and data obtained from the sensors of FarmBot
Response	A GUI facilitating control and analysis of the Farm.
Stimulus	The user selects the "Dashboard" option within the Web Application.

Continued on next page

Table 3.1: Tabular Description of the **View Dashboard** Use Case (Continued)

Normal Flow	1. The user is directed to the dashboard pane on button click. 2. If s/he is already logged into the site, the system automatically sends a signal to the FarmBot device to obtain the necessary information. 3. FarmBot sends the data obtained from the various sensors to the Web Application in order to be displayed on the dashboard tab. 4. While the user is dragging the graph about resources, the new data corresponding to the point to where the user dragged graph is obtained by the sensors. 5. If the user uses the manual control tab, the web application sends a signal to the hardware to move.
Alternative Flow	-
Exception Flow	If the user is not logged into the site, or his/her session has expired, s/he is automatically directed to the login page.
Post Conditions	The dashboard tab must be displayed on the screen.

Continued on next page

Table 3.1: Tabular Description of the **View Dashboard** Use Case (Continued)

Comments	Sending localized data offers a more smooth experience to users since it reduces the loading time of the graphs. Moreover, calculating the session time and checking for the user authentication improves the security of the application.

Table 3.2: Tabular Description of the **Observe Farm** Use Case

Use Case Name	Observe Farm
Actors	User, Web Application
Description	The user observes his/her farm thanks to the camera integrated to the FarmBot.
Preconditions	The user must be logged into the FarmBot Web Application.
Data	Images and visual data from the farm captured by the camera.
Response	The Web Application displays the latest farm images and data, allowing for visual analysis.
Stimulus	The user selects the "Farm" option within the Web Application.

Continued on next page

Table 3.2: Tabular Description of the **Observe Farm** Use Case (Continued)

Normal Flow	1. User selects "Observe Farm" from the Web Application. 2. The system verifies that the user is logged in. 3. The Web Application sends a request to the FarmBot to capture current farm images. 4. The camera captures images and sends them to the Raspberry Pi. 5. The Raspberry Pi processes and uploads the images to the Web Application. 6. The Web Application displays the images to the user for observation.
Alternative Flow	-
Exception Flow	If the user is not logged in, or the session has expired, they are redirected to the login page.
Post Conditions	The user is able to visually inspect the farm through the latest images displayed on the Web Application.

Continued on next page

Table 3.2: Tabular Description of the **Observe Farm** Use Case (Continued)

Comments	Providing real-time or regularly updated images allows users to closely monitor their farm's condition without needing to be physically present. This feature enhances the overall user experience by facilitating remote farm management and decision-making.

Table 3.3: Tabular Description of the **Control Hardware** Use Case

Use Case Name	Control Hardware
Actors	User, Web Application
Description	Users send commands to control FarmBot's hardware via the Web Application.
Preconditions	The user must be logged into the FarmBot Web Application and the FarmBot hardware must be operational and connected to the internet.
Data	Command inputs for hardware operation (e.g., watering, moving, planting).
Response	The hardware executes the specified operations and provides feedback on the action taken.
Stimulus	The user inputs hardware control commands through the dashboard and farm pane.

Continued on next page

Table 3.3: Tabular Description of the **Control Hardware** Use Case (Continued)

Normal Flow	1. User navigates to the dashboard pane. 2. User selects the desired tools and hardware to control (e.g., move the gantry or cross slide, water a specific area). 3. User gives commands. 4. The Web Application sends the command to Raspberry Pi. 5. Raspberry Pi executes the command and sends a completion signal back to the Web Application. 6. The Web Application displays a confirmation or result of the executed command to the user.

Continued on next page

Table 3.3: Tabular Description of the **Control Hardware** Use Case (Continued)

Alternative Flow	1. User sets a new schedule and chooses the hardware operations to automate (e.g., watering every morning at 7 AM, seeding specific areas on specific dates). 2. User sets the specific timing and frequency for each operation, along with any necessary parameters (e.g., water volume, seed type). 3. The Web Application saves the schedule and automatically sends commands to the FarmBot hardware based on the defined schedule. 4. At the scheduled times, FarmBot executes the pre-defined operations without the need for real-time user input. 5. After each scheduled operation, FarmBot sends a completion signal back to the Web Application.
Exception Flow	If the hardware is unreachable or an error occurs during operation, an error message is displayed.
Post Conditions	The specified hardware operation is completed, and the user is informed of the outcome.

Continued on next page

Table 3.3: Tabular Description of the **Control Hardware** Use Case (Continued)

Comments	This use case is crucial for remote management of the FarmBot, allowing users to interact with their farm in real-time. Error handling and feedback mechanisms are essential for a smooth user experience and for troubleshooting purposes.

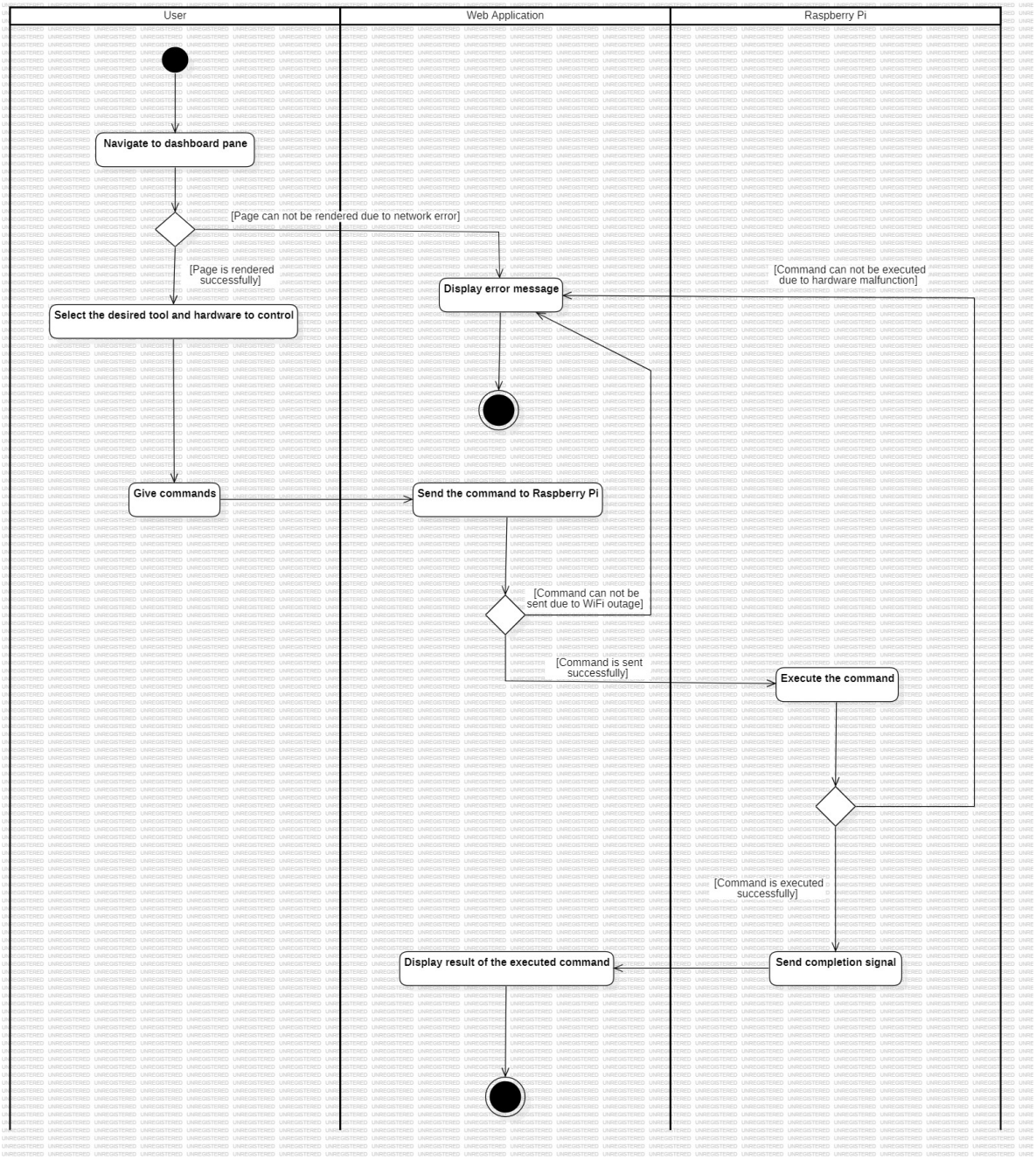

Figure 3.3: Activity Diagram for **Control Hardware** Use Case

Table 3.4: Tabular Description of the **Monitor Farm Data** Use Case

Use Case Name	Monitor Farm Data
Actors	User, Web Application
Description	Users access and review real-time and historical data collected by FarmBot's sensors and obtained from the Internet to monitor farm conditions.
Preconditions	The user must be logged into the FarmBot Web Application. FarmBot's sensors must be operational and collecting data.
Data	Sensor data including soil moisture, temperature, pH levels, and plant growth.
Response	The hardware executes the specified operations and provides feedback on the action taken.
Stimulus	The user inputs hardware control commands through the dashboard and farm pane.

Continued on next page

Table 3.4: Tabular Description of the **Monitor Farm Data** Use Case (Continued)

Normal Flow	1. User logs into the Web Application and navigates to the data section. 2. The Web Application queries the latest sensor data from Raspberry Pi. 3. FarmBot sensors transmit the requested data to the Raspberry Pi. 4. The Raspberry Pi processes and sends data to the Web Application. 5. The Web Application presents data to the user, including historical trends and real-time information.
Alternative Flow	-
Exception Flow	If sensor data is not available due to hardware issues, an error message is displayed.
Post Conditions	The user has accessed and reviewed the farm's sensor and other users' data, enabling monitoring and decision-making for farm management.
Comments	Monitoring farm data is essential for effective farm management, allowing users to adjust practices based on environmental conditions and plant needs. This use case facilitates proactive and responsive farm care, enhancing productivity and sustainability.

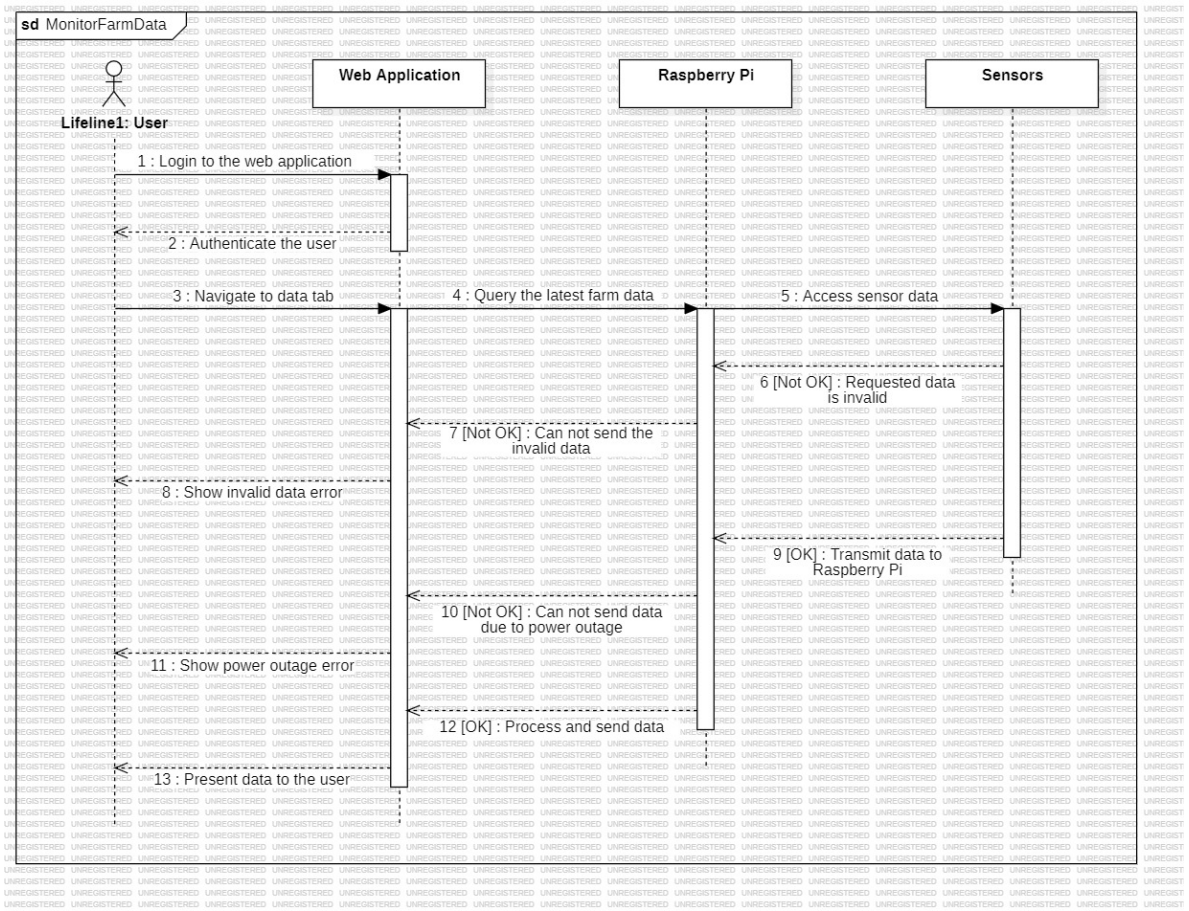

Figure 3.4: Sequence Diagram for **Monitor Farm Data** Use Case

Table 3.5: Tabular Description of the **View System Logs** Use Case

Use Case Name	View System Logs
Actors	System Admin
Description	System Admins access and review the system log to monitor FarmBot's software activities, operational history, and system alerts.
Preconditions	The System Admin must have administrative credentials. Log files must be stored.
Data	Log entries containing information about system operations, errors, user activities, and hardware statuses.
Response	Display of system logs.
Stimulus	One of the system admins attempts to view system logs.
Normal Flow	1. System Admin logs into the FarmBot Web Server with administrative credentials. 2. System admin access log files on the server.
Alternative Flow	-
Exception Flow	-
Post Conditions	System log records in the log files must be displayed to the admin.

Continued on next page

Table 3.5: Tabular Description of the **View System Logs** Use Case (Continued)

Comments	This use case is essential for system maintenance and troubleshooting, allowing the System Admin to identify and address issues proactively.

Table 3.6: Tabular Description of the **View Financial Analysis Pane** Use Case

Use Case Name	View Financial Analysis Pane
Actors	User, Web Application
Description	Users access the Financial Analysis Pane within the FarmBot Web Application to review financial data related to their farming operations.
Preconditions	The user must be logged into the FarmBot Web Application.
Data	Financial data including costs of resources (water, seeds, fertilizers, etc.), operational expenses, and projected revenues from crop yields.
Response	Display of financial analysis and projections in a graphical or tabulated format.
Stimulus	The user selects the Financial Analysis Pane from the Web Application's dashboard.

Continued on next page

Table 3.6: Tabular Description of the **View Financial Analysis Pane** Use Case (Continued)

Normal Flow	1. User navigates to and selects the Financial Analysis Pane option. 2. System Admin makes desired changes to the settings and submits the modifications. 3. The Web Application retrieves and displays financial data related to the user's farm. 4. User reviews the information for planning and decision-making purposes.
Alternative Flow	-
Exception Flow	If financial data cannot be retrieved, an error message is displayed.
Post Conditions	The user has accessed and reviewed the financial analysis of their farming operations.
Comments	This use case is vital for users to understand the financial performance of their farm, enabling better resource management, operational planning, and profitability analysis.

Table 3.7: Tabular Description of the **View Hardware Information** Use Case

Use Case Name	View Hardware Information

Continued on next page

Table 3.7: Tabular Description of the **View Hardware Information** Use Case (Continued)

Actors	User, Web Application
Description	Users access a dedicated pane within the FarmBot Web Application to view and manage information about their FarmBot's hardware and tooling
Preconditions	The user must be logged into the FarmBot Web Application.
Data	Details on installed hardware components (tracks, gantry, tool mounts, sensors), their status, and configuration settings including dimensions and operational status (active/inactive).
Response	The Web Application displays the hardware information pane, allowing users to view, modify, and update their hardware configurations.
Stimulus	The user selects the hardware information option in the Web Application.

Continued on next page

Table 3.7: Tabular Description of the **View Hardware Information** Use Case (Continued)

Normal Flow	1. User navigates to the hardware information pane using the dropdown menus. 2. The pane displays current hardware configurations, with options to add new components or modify existing ones. 3. Users can activate or deactivate specific hardware components, adjust dimensions, or update the system with new hardware additions.
Alternative Flow	-
Exception Flow	If there's an issue accessing or modifying hardware data, an error message is displayed.
Post Conditions	The user has successfully accessed and possibly modified the hardware information for their FarmBot.
Comments	This use case facilitates effective management and customization of FarmBot's hardware setup, enabling users to optimize their system according to specific farming needs and space requirements.

Table 3.8: Tabular Description of the **View Resource Usage** Use Case

Use Case Name	View Resource Usage

Continued on next page

Table 3.8: Tabular Description of the **View Resource Usage** Use Case (Continued)

Actors	User, Web Application
Description	Users access the Resource Usage Pane within the FarmBot Web Application to examine the consumption and costs of resources such as electricity, water, seeds, and fertilizers.
Preconditions	The user must be logged into the FarmBot Web Application.
Data	Historical, current, and projected resource usage data, including volume, timing, and monetary costs.
Response	Display of an interactive graph showing resource usage with zoom and filter capabilities for detailed analysis.
Stimulus	The user selects the Resource Usage Pane from within the Web Application.
Normal Flow	1. User navigates to and selects the Resource Usage Pane. 2. The Web Application presents an interactive graph detailing resource usage. 3. User interacts with the graph to zoom in on details or filter by specific resources like water usage on tomatoes.
Alternative Flow	-

Continued on next page

Table 3.8: Tabular Description of the **View Resource Usage** Use Case (Continued)

Exception Flow	If data cannot be retrieved, an error message is displayed.
Post Conditions	The user has accessed detailed resource usage information, aiding in efficient management and cost-saving strategies.
Comments	This feature enables users to closely monitor and optimize resource consumption, contributing to sustainable and cost-effective farm management.

Table 3.9: Tabular Description of the **View System Information** Use Case

Use Case Name	View System Information
Actors	User, Web Application
Description	Users access the System Information Pane within the FarmBot Web Application to review detailed information about their FarmBot system's status.
Preconditions	The user must be logged into the FarmBot Web Application.
Data	System status data, manual control options, financial expenditures on inputs, expected revenues from crops, and other operational factors such as maintenance and logistics costs.

Continued on next page

Table 3.9: Tabular Description of the **View System Information** Use Case (Continued)

Response	The Web Application displays the system information with the capability for users to apply advanced filtering to the data presented.
Stimulus	The user selects the System Information Pane from the Web Application's dashboard.
Normal Flow	1. User navigates to and selects the System Information Pane option. 2. The Web Application retrieves and displays the current system information. 3. User interacts with the pane to view specific system details, utilizing filtering options as needed for detailed analysis.
Alternative Flow	-
Exception Flow	If data cannot be retrieved, an error message is displayed.
Post Conditions	The user has accessed and reviewed the system status and financial analysis of their farming operations.

Continued on next page

Table 3.9: Tabular Description of the **View System Information** Use Case (Continued)

Comments	This use case provides users with crucial information for understanding their FarmBot's operational status and the financial health of their farm, facilitating informed decision-making and system management.

Table 3.10: Tabular Description of the **Store Farm Data** Use Case

Use Case Name	Store Farm Data
Actors	User, OpenFarm.cc API
Description	Users upload and store their farm data on OpenFarm.cc, enabling them to share their experiences and leverage data shared by others for optimizing their FarmBot operations.
Preconditions	The user must be registered with OpenFarm.cc and have an active internet connection.
Data	Farm layouts, plant growth data, operational sequences, and other relevant farming data.
Response	The data is successfully uploaded and stored on OpenFarm.cc, making it accessible to other users.

Continued on next page

Table 3.10: Tabular Description of the **Store Farm Data** Use Case (Continued)

Stimulus	The user selects the option to upload data to OpenFarm.cc through the FarmBot Web Application.
Normal Flow	1. User is prompted to log in or verify their OpenFarm.cc account if not already authenticated. 2. User selects the option to store farm data on OpenFarm.cc. 3. User selects or inputs the farm data they wish to upload. 4. The Web Application sends the data to OpenFarm.cc using its API. 5. OpenFarm.cc confirms successful data storage. 6. The user receives confirmation of the successful upload within the FarmBot Web Application.
Alternative Flow	-
Exception Flow	If there's an issue with internet connectivity or OpenFarm.cc API is unavailable, the user is informed of the failure to store data. If authentication fails, the user is prompted to re-authenticate.
Post Conditions	The user's farm data is stored on OpenFarm.cc, available for their personal use and sharing with the community.

Continued on next page

Table 3.10: Tabular Description of the **Store Farm Data** Use Case (Continued)

Comments	This use case expands the utility of FarmBot by integrating community-driven data sharing, offering users a platform to contribute to and benefit from collective farming knowledge and data.

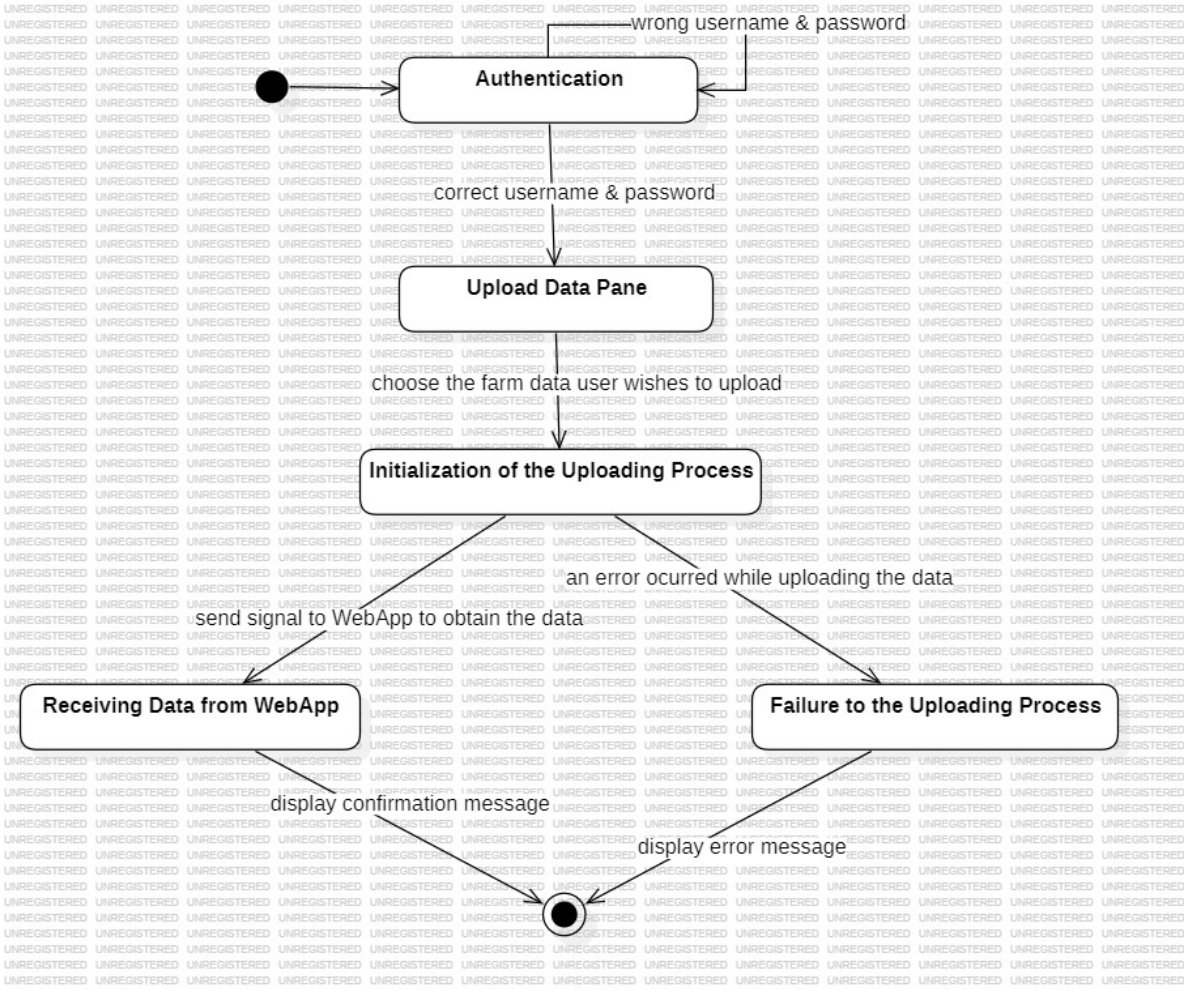

Figure 3.5: State Diagram for **Store Farm Data** Use Case

3.3 Logical Database Requirements

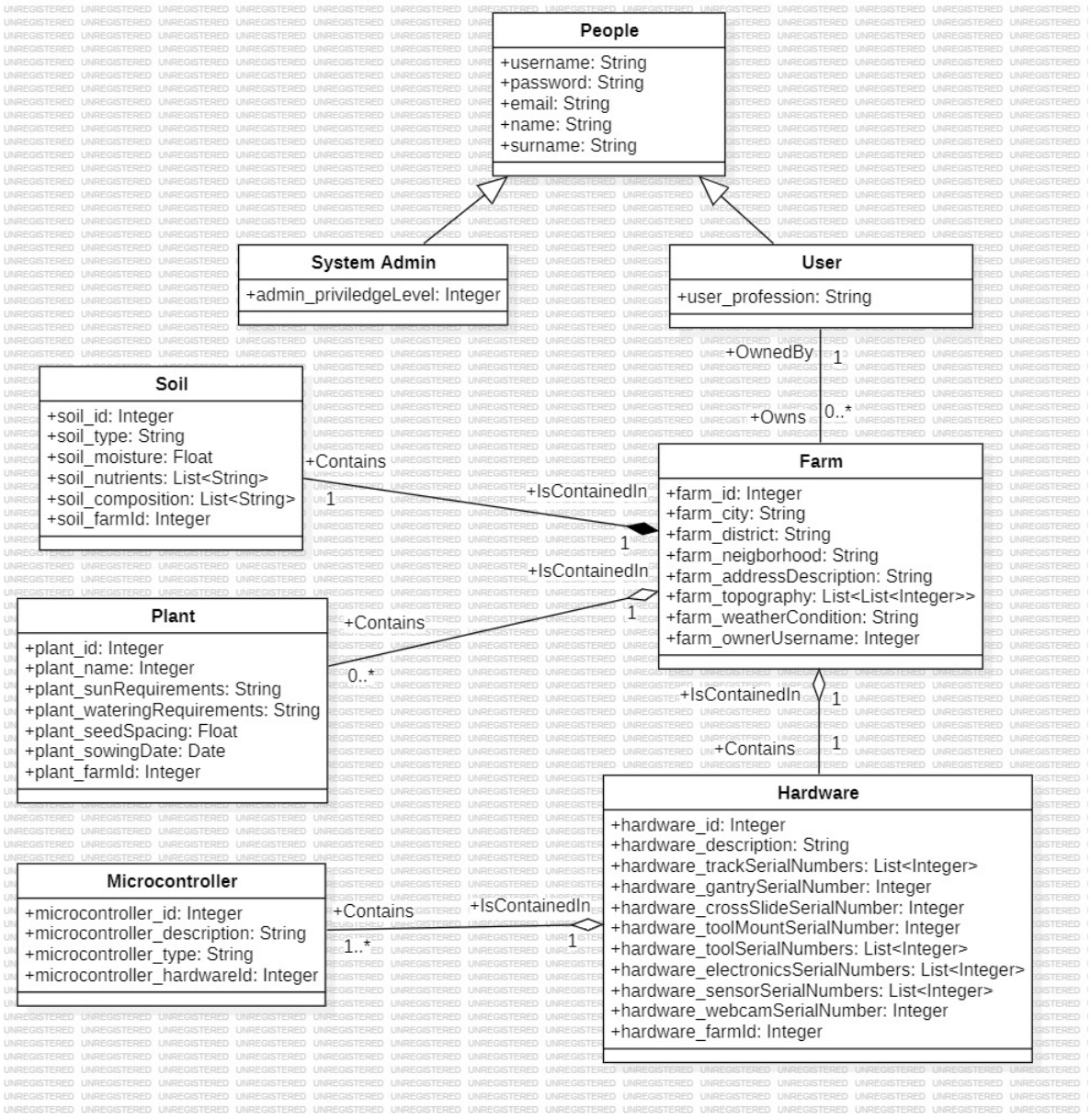

Figure 3.6: Logical Database Diagram

3.4 Design Constraints

- FarmBot must adhere to international data privacy laws like GDPR, ensuring user data is handled according to legal standards.
- Its development must align with the open-source licenses like MIT of its hardware and software components, maintaining public accessibility and respecting intellectual property rights.
- Compliance with electrical and mechanical safety standards is crucial to prevent user injury or crop damage.
- FarmBot should integrate into various farming operations without harming soil quality or the ecosystem, adaptable to different farming scales.
- Requires industry-standard data exchange and system integration protocols to ensure compatibility with existing agricultural technology and cloud services.

These constraints guide FarmBot's development, focusing on safety and effectiveness while meeting legal and environmental standards.

3.5 System Quality Attributes

- **Reliability:** FarmBot requires a highly reliable software system to ensure accurate and timely execution of farming tasks. This includes:
  - Regular backup of farm data, including designs, user settings, and operational logs, to prevent data loss.
  - Zero tolerance for operational errors that could impact plant health or resource use efficiency.
- **Availability:** The FarmBot system is designed for high availability to support continuous agricultural activities. The cloud-based services of FarmBot should

ensure data synchronization and system updates without interrupting core functionalities.

- **Security:** Protecting user data and ensuring the integrity of FarmBot operations is essential:
  - Maintain access logs for monitoring and auditing purposes.
  - Implement role-based access controls to segregate user functionalities and protect sensitive system settings.
  - Enforce data integrity checks for critical variables to prevent unauthorized modifications.
  - Ensure privacy of user data in compliance with data protection regulations.
- **Maintainability:** The software architecture of FarmBot emphasizes maintainability to facilitate updates and enhancements:
  - Adhere to modularity in design to simplify updates and module replacements.
  - Adopt continuous integration and deployment practices to streamline software updates and patch implementations.
- **Portability:** FarmBot software aims to be easily portable to support a wide range of hardware setups and operating environments. It has software designed to be compatible with major operating systems and platforms, including Linux-based systems for Raspberry Pi and web browsers for the WebApp.

3.6 Supporting Information

The FarmBot project is an open-source initiative focused on advancing agricultural practices through automation and technology. The entire source code and design files for both the hardware and software components are accessible on GitHub, allowing

anyone interested to view, analyze, and contribute to the project's development. Numerous open issues on the GitHub page are awaiting solutions, offering opportunities for contributors to engage in problem-solving and feature enhancement.

Beyond software development, there are additional ways to support FarmBot. Enthusiasts and professionals can contribute by sharing insights on optimal farming practices, suggesting improvements for hardware design, or even participating in educational outreach to spread knowledge about this technology. By joining the FarmBot community, you contribute to a global movement aimed at making sustainable farming accessible to everyone, leveraging the collective knowledge and skills of a diverse group of contributors.

4. Suggestions to Improve The Existing System

4.1 System Perspective

Improved FarmBot includes a Mobile Application and ForumBot. The Mobile Application brings farm management to mobile devices, while ForumBot allows users to exchange farming tips and layouts. These new features integrate seamlessly with the existing system, enhancing the user experience with additional flexibility and community support.

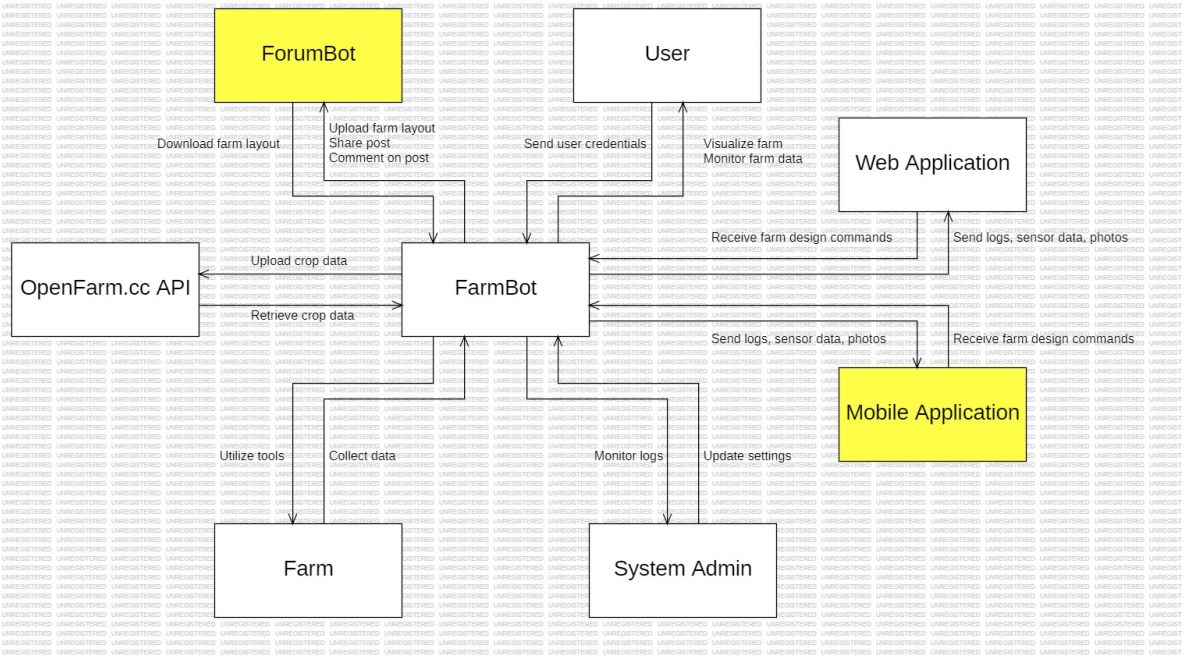

Figure 4.1: Context Diagram of the Improved System

4.2 External Interfaces

In addition to the external interfaces represented in 3.1, Mobile Application and ForumBot are introduced. Each of them has its own external interface:

- **Mobile Application Interface:** This interface manages the interactions between the user's mobile device and FarmBot. It supports user authentication, task scheduling, and retrieval of data such as sensor readings and hardware information. It's tailored for mobile usage, taking into account the device's name and the user's phone number for personalized interactions.

- **ForumBot Interface:** This interface is designed for community engagement within the FarmBot ecosystem. It allows users to create a username and password, post discussions, share insights, comment on existing posts, and manage their profile and activity logs. Users can also download and upload farm layouts, making it a platform for sharing knowledge and farming strategies.

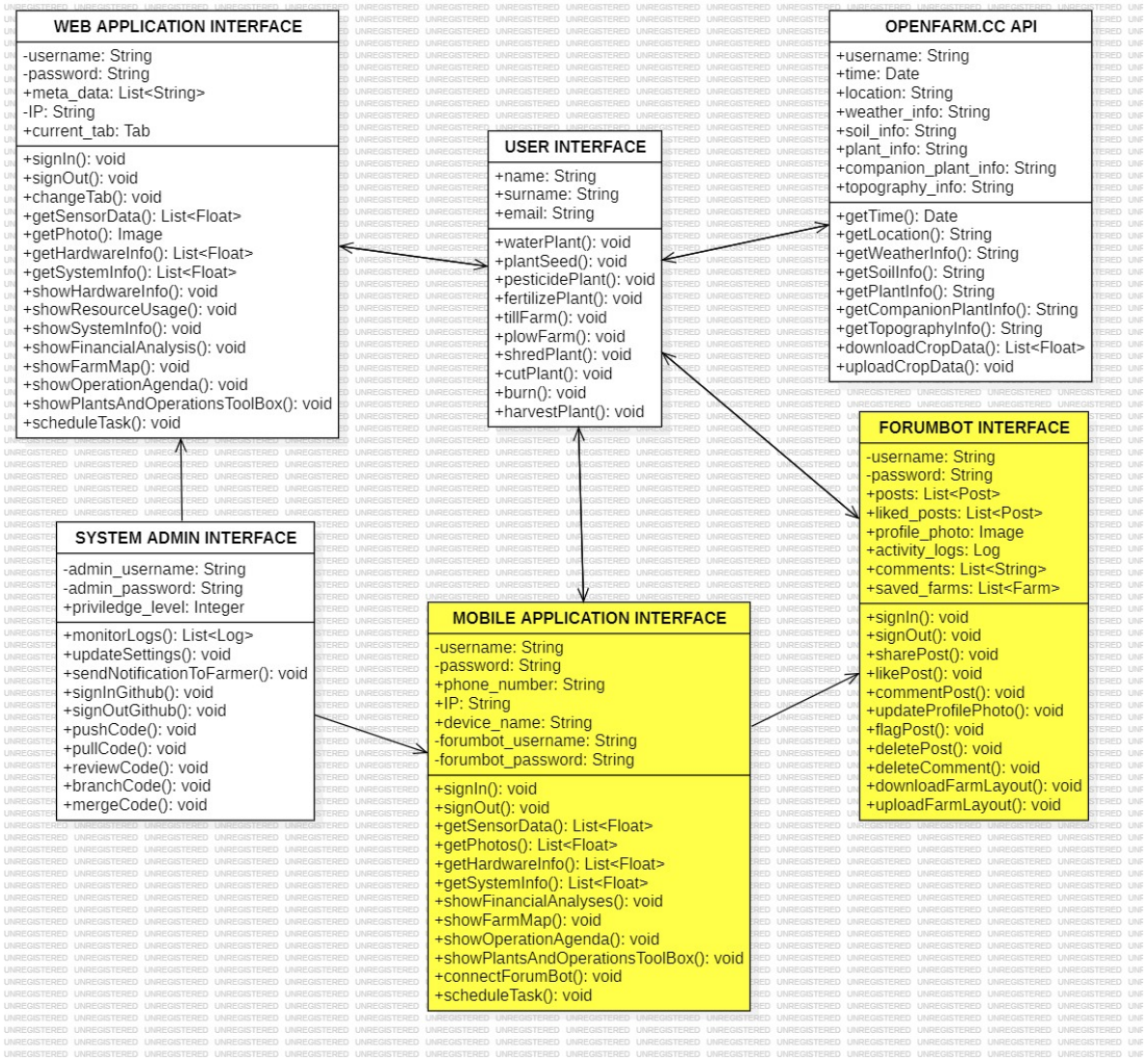

Figure 4.2: External Interfaces of the Improved System

4.3 Functions

The improved use case diagram integrates new external entities, which are Mobile Application and ForumBot. Now, users can engage with the FarmBot community by sharing and commenting on posts and downloading farm layouts from ForumBot. The Mobile Application enables users to schedule watering tasks directly from their mobile devices, enhancing FarmBot's accessibility and ease of use. These new use cases are pivotal for a more interactive and mobile-friendly FarmBot experience.

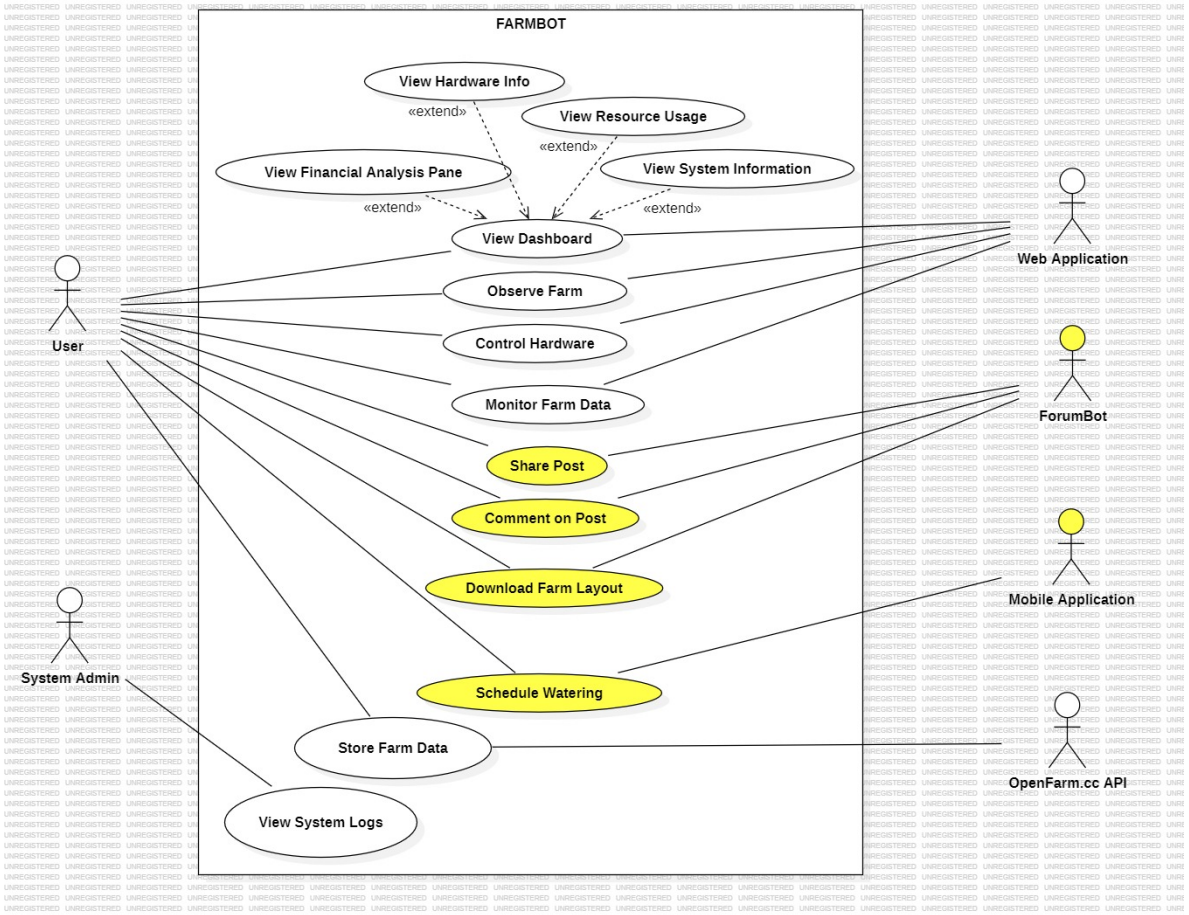

Figure 4.3: Use Case Diagram of the Improved System

Table 4.1: Tabular Description of the **Download Farm Layout** Use Case of the Improved System

Use Case Name	Download Farm Layout
Actors	User, ForumBot
Description	The user downloads a farm layout shared by another user from ForumBot, a platform for sharing and downloading FarmBot farm layouts.
Preconditions	The user must be registered and logged into ForumBot.
Data	Farm layout files including plant placement, watering schedules, and other operational sequences.
Response	The user successfully downloads the selected farm layout to their local device.
Stimulus	The user selects a farm layout and initiates the download process on ForumBot.

Continued on next page

Table 4.1: Tabular Description of the **Download Farm Layout** Use Case of the Improved System (Continued)

Normal Flow	1. User logs into ForumBot website. 2. User browses farm layouts available on ForumBot. 3. User selects a desired farm layout. 4. User clicks the download button for the selected layout. 5. ForumBot processes the request and initiates the file download. 6. The farm layout file is downloaded to the user's local device.
Alternative Flow	If the desired farm layout is not available or the download link is broken, the user can request the layout directly from the layout's author via a messaging feature on ForumBot.
Exception Flow	If the user is not logged in or the session has expired, they are redirected to the login page.
Post Conditions	The selected farm layout is stored on the user's local device, ready for upload to their FarmBot Web Application.

Continued on next page

Table 4.1: Tabular Description of the **Download Farm Layout** Use Case of the Improved System (Continued)

Comments	This use case enables users to leverage community-shared farm layouts, fostering a collaborative environment where users can learn from and improve upon each other's designs. It emphasizes the importance of user interaction and content sharing within the FarmBot community.

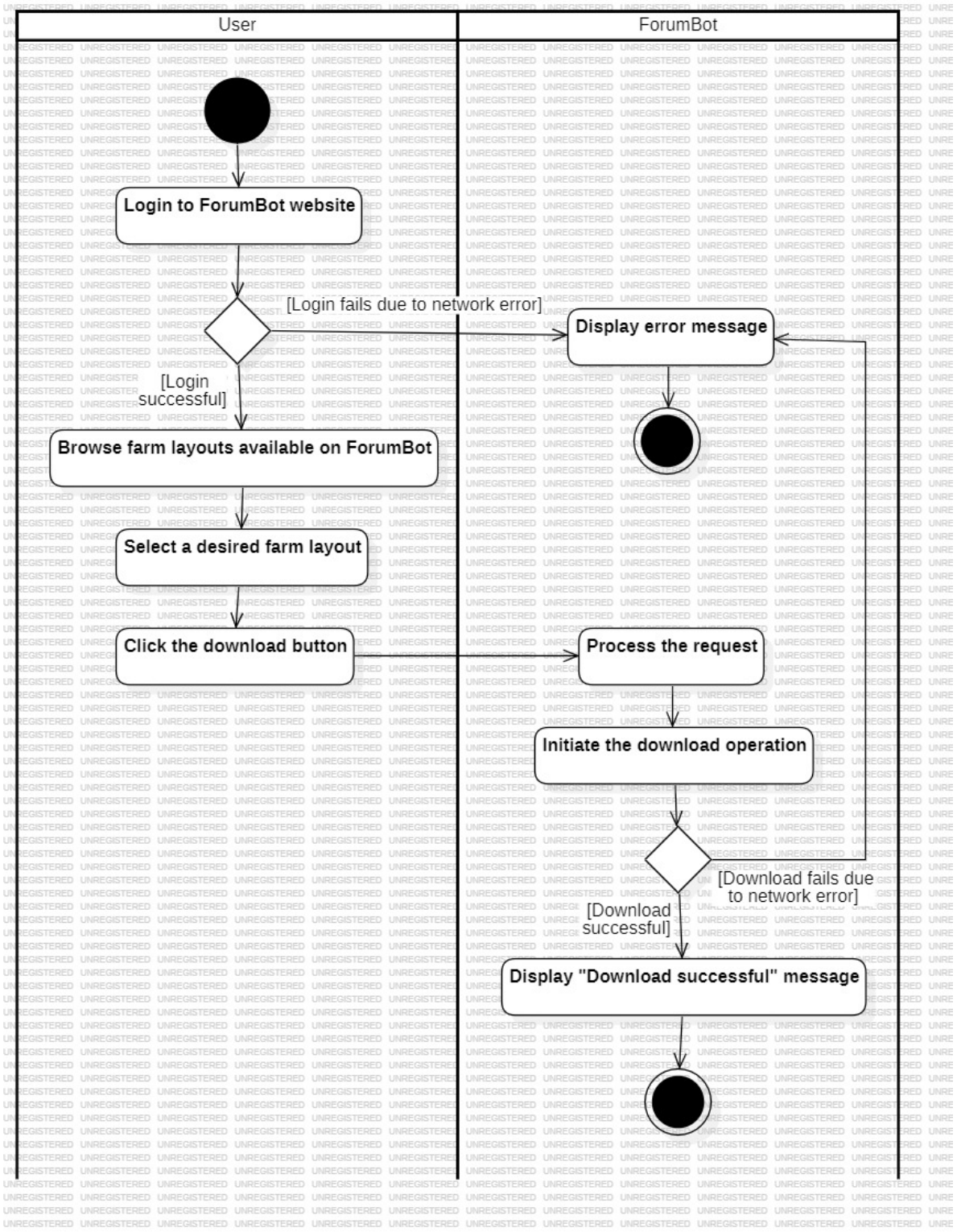

Figure 4.4: Activity Diagram for **Download Farm Layout** Use Case of the Improved System

Table 4.2: Tabular Description of the **Share Post** Use Case of the Improved System

Use Case Name	Share Post
Actors	User, ForumBot
Description	The User shares a post on ForumBot, a community platform where FarmBot users can share insights, ask questions, and discuss various topics related to FarmBot.
Preconditions	The user must be registered and logged into ForumBot.
Data	Text content, images, or links included in the post.
Response	The post is successfully published on ForumBot, visible to other users.
Stimulus	The user composes a post and submits it for publication on ForumBot.
Normal Flow	1. User logs into ForumBot and clicks to the "Create Post" button. 2. User composes the post and attaches any relevant images or links. 3. User submits the post. 4. ForumBot processes the submission and publishes the post. 5. The post becomes visible to other ForumBot users in the relevant discussion category.

Continued on next page

Table 4.2: Tabular Description of the **Share Post** Use Case of the Improved System (Continued)

Alternative Flow	-
Exception Flow	If the user is not logged in or the session has expired, they are redirected to the login page.
Post Conditions	The user's post is publicly accessible on ForumBot, facilitating community engagement and information exchange.
Comments	This use case fosters an active community dialogue, allowing users to contribute knowledge, share experiences, and support one another in the effective use of FarmBot.

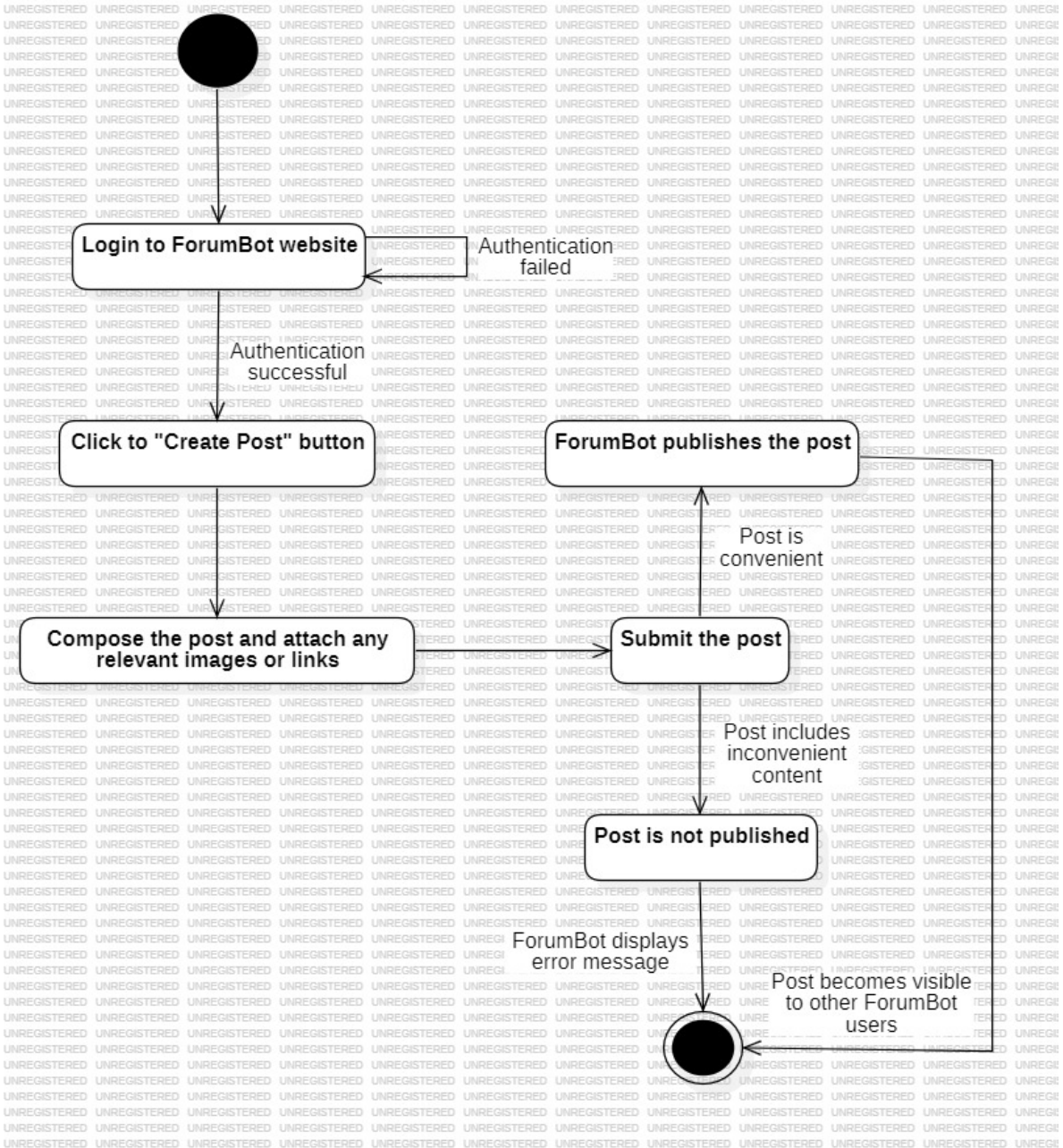

Figure 4.5: State Diagram for **Share Post** Use Case of the Improved System

Table 4.3: Tabular Description of the **Comment on Post** Use Case of the Improved System

Use Case Name	Comment on Post
Actors	User, ForumBot
Description	The User submits a comment on an existing post within the ForumBot platform.
Preconditions	The user must be registered and logged into ForumBot.
Data	Text content, images, or links included in the comment.
Response	The user's comment is successfully added below the original post, visible to other users.
Stimulus	The user decides to contribute to a discussion by commenting on a post on ForumBot.

Continued on next page

Table 4.3: Tabular Description of the **Comment on Post** Use Case of the Improved System (Continued)

Normal Flow	1. User logs into ForumBot and navigates to a post of interest. 2. User enters their comment in the designated comment section below the post. 3. ForumBot processes the submission and displays the comment under the post. 4. ForumBot processes the submission and publishes the post. 5. The comment becomes visible to other ForumBot users, facilitating further discussion.
Alternative Flow	-
Exception Flow	If the user is not logged in or the session has expired, they are redirected to the login page. If the user attempts to submit an empty comment or one that violates community guidelines, ForumBot displays an error message.
Post Conditions	The user's comment is publicly accessible on ForumBot, contributing to the ongoing dialogue and community engagement around the post.

Continued on next page

Table 4.3: Tabular Description of the **Comment on Post** Use Case of the Improved System (Continued)

Comments	This use case encourages active participation and exchange of ideas within the FarmBot community, enabling users to ask questions, provide answers, and share insights, thereby enriching the collective knowledge base.

Table 4.4: Tabular Description of the **Schedule Watering** Use Case of the Improved System

Use Case Name	Schedule Watering
Actors	User, Mobile Application
Description	The user schedules a watering operation for their FarmBot via the Mobile Application, setting up the timing and duration for watering specific plant zones.
Preconditions	The user must be registered with and logged into the Mobile Application linked to their FarmBot.
Data	Specific zones to be watered, start time, duration of watering.
Response	The Mobile Application schedules the watering operation on the FarmBot, and the user receives confirmation of the scheduled task.

Continued on next page

Table 4.4: Tabular Description of the **Schedule Watering** Use Case of the Improved System (Continued)

Stimulus	The user decides to schedule a watering operation through the Mobile Application.
Normal Flow	1. User logs into the Mobile Application. 2. User navigates to the "Schedule Task" and then the "Watering Schedule" section. 3. User selects the plant zones to water, sets the start time, and specifies the duration. 4. User confirms and submits the watering schedule. 5. Mobile Application sends the request to Raspberry Pi. 6. Raspberry Pi executes the scheduled watering operation in the specified time. 7. Raspberry Pi sends signals to Mobile Application every time the watering operation happens. 8. Mobile Application sends notifications to the user every time the watering operation happens.
Alternative Flow	-

Continued on next page

Table 4.4: Tabular Description of the **Schedule Watering** Use Case of the Improved System (Continued)

Exception Flow	If the user is not logged in or the session has expired, they are redirected to the login page. If FarmBot is offline or unable to execute the scheduled task, the user is notified of the failure to schedule or execute the watering operation.
Post Conditions	The watering operation is successfully scheduled and confirmed, ready to be executed by FarmBot at the specified time.
Comments	This use case enables precise and efficient water management, allowing users to optimize watering schedules based on the specific needs of their plants, thereby conserving water and ensuring plant health.

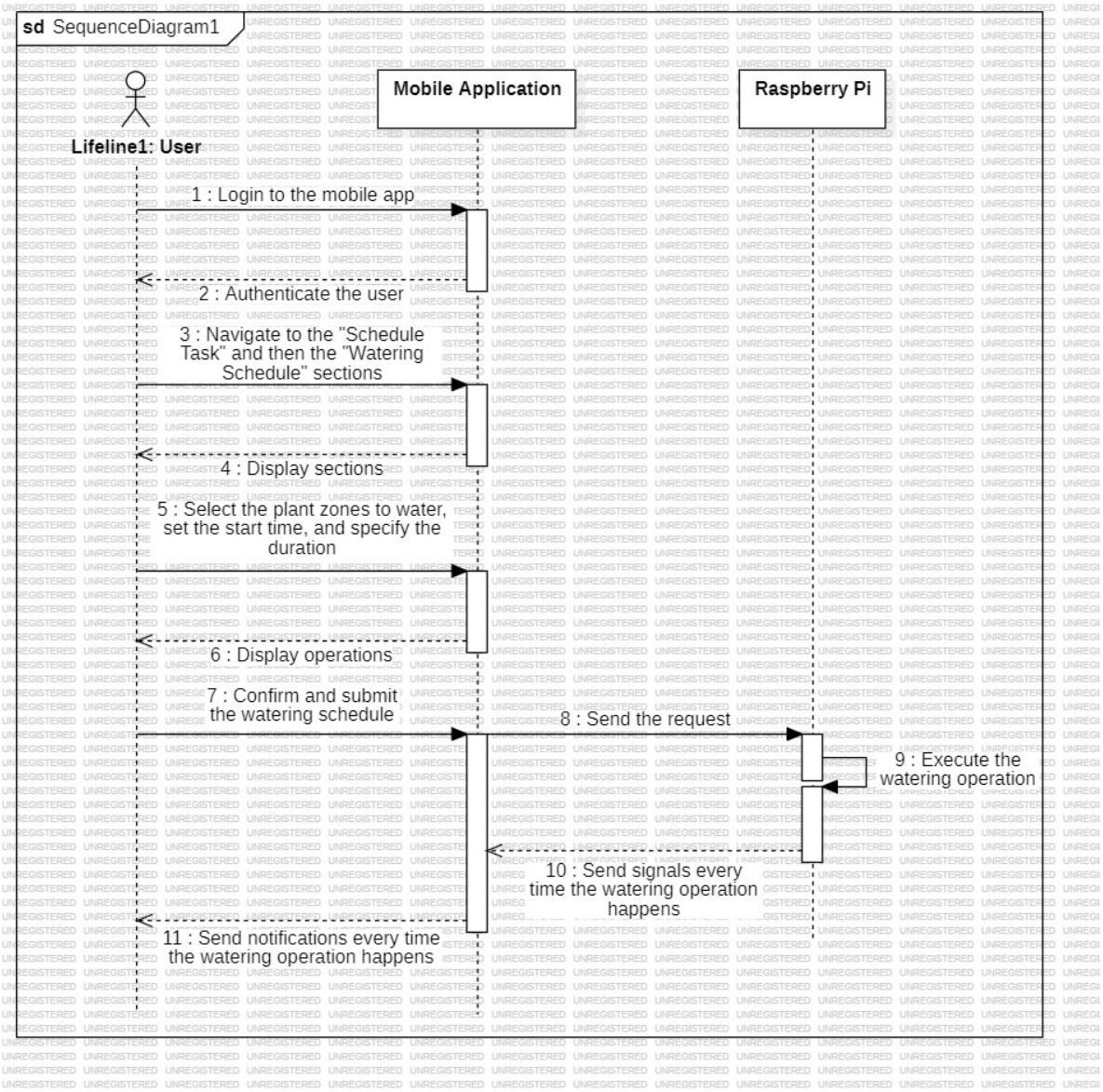

Figure 4.6: Sequence Diagram for **Schedule Watering** Use Case of the Improved System

4.4 Logical Database Requirements

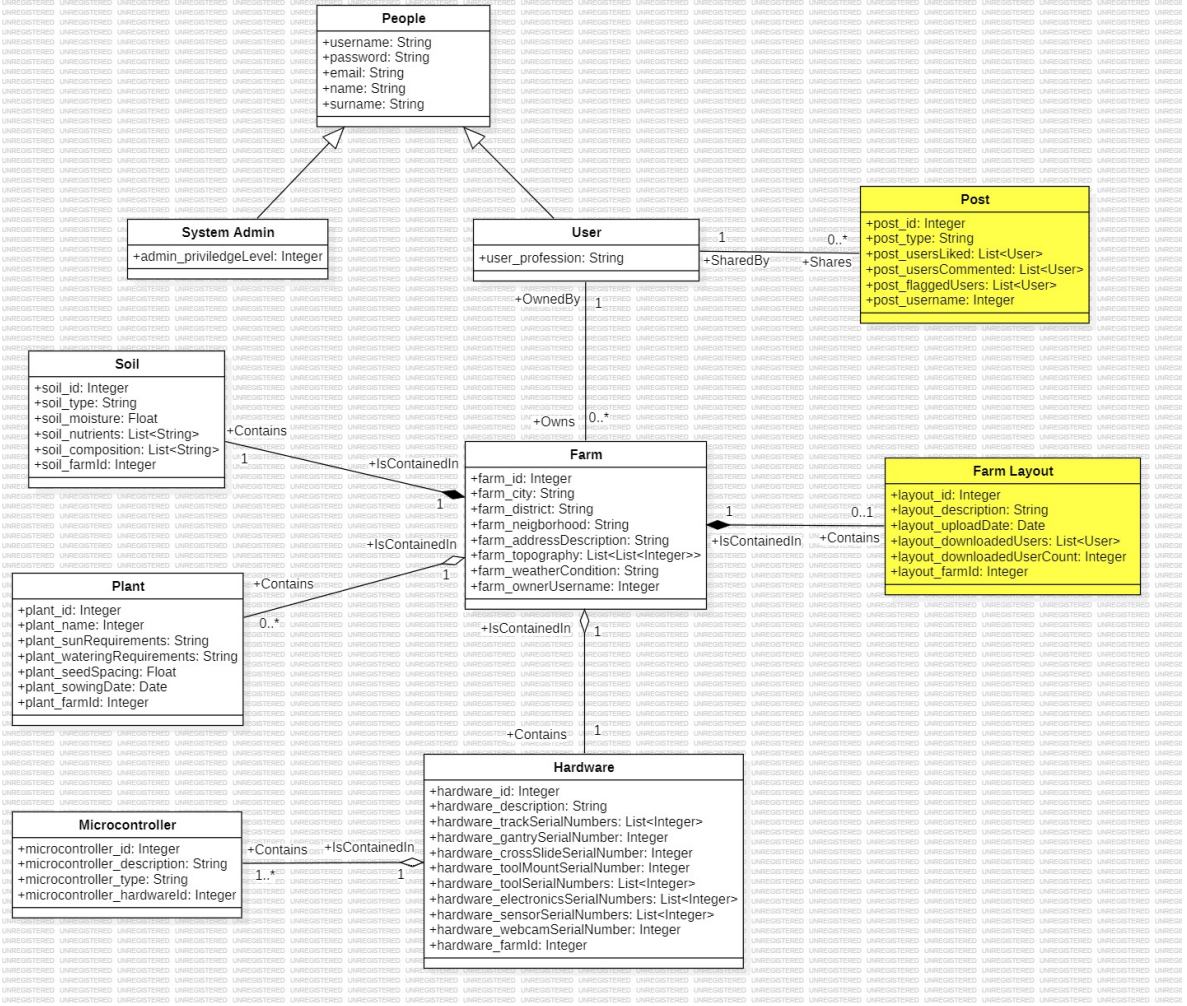

Figure 4.7: Logical Database Diagram of the Improved System

4.5 Design Constraints

Introducing the Mobile Application and ForumBot to the FarmBot project brings new design constraints:

- **Mobile App Security & Compatibility:** The app must be secure and work smoothly on different devices and operating systems by protecting users' information.
- **ForumBot Safety & Privacy:** ForumBot needs features for safe community interaction, including privacy protection and tools to handle inappropriate content.
- **Content Moderation:** Tools to monitor and manage content are essential to keep ForumBot positive and respectful.
- **Data Sharing:** Sharing and using farm layouts between FarmBot and ForumBot should be straightforward, with common data formats for compatibility.

These new constraints aim to make sure FarmBot's expanded features are user-friendly, secure, and accessible to all.

4.6 System Quality Attributes

To enhance the system quality attributes of FarmBot, considering the new additions of the Mobile Application and ForumBot, the following points are integrated:

- **Reliability (ForumBot & Mobile App):**
  - Ensure consistent performance of ForumBot to support community engagement without disruptions.
  - The Mobile Application should have mechanisms to handle offline scenarios or weak network conditions, ensuring task scheduling and monitoring are always accessible.

- **Availability (ForumBot & Mobile App):**
  - ForumBot should implement failover strategies to maintain its availability for user interactions even during high traffic or server maintenance.
  - The Mobile Application requires effective synchronization with FarmBot's cloud services to keep user data up-to-date across all platforms.
- **Security (ForumBot & Mobile App):**
  - Secure authentication and encryption for ForumBot and the Mobile Application to protect user interactions and personal information.
  - Implement additional security measures for ForumBot to prevent spam, abuse, and data breaches within the community platform.
- **Maintainability (ForumBot):**
  - Design ForumBot with a scalable architecture to accommodate growing user base and content without compromising performance.
  - Regular updates and community feedback loops in ForumBot to enhance features and address user needs effectively.
- **Portability (Mobile App):**
  - The Mobile Application should be developed with cross-platform frameworks to ensure seamless functionality across iOS, Android, and other mobile operating systems.

These additions aim to enhance the quality of FarmBot's ecosystem, ensuring that the new features provide a secure, reliable, and user-friendly experience while accommodating the expanding scope of the project.

4.7 Supporting Information

The FarmBot project has expanded with the Mobile Application and ForumBot. This broadens community engagement and interaction possibilities.

- **Mobile Application:** The FarmBot Mobile Application brings farm management to users' smartphones, allowing them to control their FarmBot from anywhere. Developers can help to improve the app's functionality and design to meet the community's needs.
- **ForumBot:** This platform enables the FarmBot community to exchange knowledge, experiences, and layouts. Users can contribute by posting content, answering questions, and sharing feedback, encouraging a collaborative space for discussion and innovation.
- **Community Contributions:** The introduction of Mobile Application and ForumBot opens new avenues for contributions beyond traditional software development, including community support and educational outreach, enriching the FarmBot ecosystem for users worldwide.

Integration of these new features strengthens the FarmBot community's collaborative efforts in advancing sustainable, technology-driven agriculture.