Department of Computer Engineering

CENG350 Software Engineering

Software Architecture Description (SAD) for FarmBot

Group 56

By

Hasan Küreli, 2580751

Yusuf Sami Lök, 2521748

Saturday 18th May, 2024

Contents

List of Figures

List of Tables

Revision History

Date	Description	Version
10.2011	Initialization	none
21.07.2016	FarmBot	Genesis (V1.0)

Table 1: Revision History of FarmBot

1. Introduction

This document serves as the Software Architecture Description (SAD) for the FarmBot V1.0 open-source project, which is a robot that uses farming in a designated region. The FarmBot firm developed and provided support for it.

1.1 Purpose and objectives of FarmBot

FarmBot is a flexible and effective instrument for precision farming that is intended to automate and optimize agricultural activities. FarmBot seeks to reduce labor and environmental impact while streamlining crop cultivation, optimizing resource consumption, and improving yields through the integration of cutting-edge technology with conventional agricultural methods. It makes it possible to sow, irrigate, weed, and harvest crops precisely, guaranteeing that each plant receives the right amount of space, time, and resources. Farmers can adopt flexible and adaptive farming practices by tailoring FarmBot's operations to their own crop varieties, planting times, and environmental conditions. FarmBot boosts farming efficiency and lowers the need for manual labor by automating repetitive chores and scheduling operations. This allows users to manage a greater area of land with less resources.

Reduced water consumption, less chemical inputs, and targeted treatments by FarmBot maximize plant health and promote resource stewardship and environmental conservation. Apart from its practical applications, FarmBot serves as a teaching platform covering robotics, technology, and agriculture. It encourages creativity by let-

ting users try out cutting-edge farming techniques, sensor technologies, and automation schemes.

In the end, FarmBot hopes to revolutionize agriculture by empowering individuals and groups to cultivate food in a productive, sustainable, and independent way. Food security, environmental sustainability, and technical innovation in farming practices would all benefit from this.

1.2 Scope

Software products work together to enable users to automate and manage various aspects of their farming operations using FarmBot hardware. Here are:

- **FarmBot OS**: With the help of this software, which acts as the FarmBot hardware's operating system, automated farming operations may be controlled and managed.
- **FarmBot Web Interface**: Through this web application, users can schedule tasks, schedule interactions with the FarmBot system, schedule tasks, check progress, and get notifications.
- **FarmBot Mobile Application**: A mobile application improves the online interface and gives users greater flexibility to operate and keep an eye on FarmBot from their smartphones or tablets.

Using an easy-to-use interface, you can manage FarmBot's motions and functions, such as planting, watering, weeding, and harvesting. Adjust planting dates, watering schedules, and upkeep assignments according to crop varieties, local conditions, and user preferences. Get up-to-date information, notifications, and alerts in real-time on tasks completed, system health, and possible problems that need to be fixed. To evaluate progress, assess trends, and make well-informed decisions for optimizing farming methods, access sensor readings, historical data, and farming analytics.

The application of the specified software aims to simplify agricultural procedures by boosting farming productivity, eliminating the need for manual labor, and automating repetitive operations. Encourage precision farming by increasing agricultural yields, reducing waste, and making the most use of available resources. Promote soil health through targeted treatments, minimize water use, and decrease chemical inputs to foster sustainability and environmental stewardship. Give people and communities the tools they need to grow food in a sustainable, self-sufficient, and efficient manner, thereby promoting environmental preservation and food security.

This scope aligns with higher-level requirements, such as system requirements, which outline the functionality, performance, and interoperability requirements for the FarmBot system. It also fits in with the larger goals and objectives of the FarmBot project, which prioritize innovation, education, and empowerment in the realms of agriculture and technology.

The business domain under consideration is agricultural technology, specifically focusing on precision farming solutions.

The range of business activities included in this business domain encompasses various divisions and functions related to modern agricultural practices:

- **Crop Cultivation**: Activities related to planting, watering, weeding, and harvesting crops.
- **Resource Management**: Management of resources such as water, soil nutrients, and pesticides to optimize crop growth and yield.
- **Equipment Management**: Maintenance and operation of farming equipment, including tractors, irrigation systems, and precision farming tools.
- **Data Analysis and Decision Making**: Analysis of sensor data, weather patterns, and crop performance to make informed decisions regarding planting, irrigation, and pest control.

- **Environmental Monitoring**: Monitoring of environmental factors such as soil moisture, temperature, and humidity to assess crop health and identify areas for improvement.

The FarmBot system, an automated precision farming solution, is the scope of the system being built or modified within this business domain. Crop cultivation, resource management, equipment management, data analysis, and decision-making are all supported by the system. It is presumed that the system will mainly concentrate on automating chores like planting, watering, and weeding in addition to offering real-time control and monitoring features to maximize farming operations. To improve functionality and aid in decision-making, the system may also link with outside organizations like weather services, soil testing facilities, and agricultural marketplaces.

1.3 Stakeholders and their concerns

FarmBot Users

The main concerns of FarmBot customers are the system's usability, efficacy, and dependability. They anticipate that the system will be easy to use, intuitive, and able to effectively handle their gardening responsibilities. Users also like how quickly and accurately the system can provide feedback on the health of the plants and their growth.

Agricultural Experts

Experts in agriculture are interested in FarmBot's scientific accuracy and effectiveness in supporting various farming and gardening strategies. They expect the system to incorporate best practices for irrigation, soil management, and plant maintenance. Agricultural specialists also search for cutting-edge features for data analysis and decision support in order to optimize crop health and yield.

System Admins

It is the duty of system Administrators to install, configure, and maintain the FarmBot system. Strong Administration tools are required to track system performance, manage user accounts, and resolve problems. Another tactic system glsadministrators use to

protect the system from internet threats and unauthorized access is to prioritize security measures.

Software Developers

The FarmBot software's scalability, extensibility, and maintainability are top priorities for engineers. To support future improvements and collaboration, they require standardized development procedures, modular architecture, and well-documented code. In order to improve FarmBot's capabilities, developers also look for support for integrating third-party products and services.

Regulatory Authorities

Regulatory authorities are concerned with ensuring adherence to relevant laws and policies concerning agricultural technology and data protection. In order to maintain regulatory compliance and safeguard client privacy, they must be transparent about the ways in which they collect, process, and retain data. Regulatory agencies may also offer guidelines for safety regulations and environmental impact evaluations.

Hardware Manufacturers

The adaptability and interoperability of FarmBot with hardware manufacturers' products piques their interest. To ensure a seamless integration with FarmBot systems, they expect specific norms and standards for hardware interfaces. Hardware manufacturers may also seek opportunities for collaboration in order to optimize hardware designs for FarmBot compatibility and performance.

[1]

2. References

[1] FarmBot Inc., "FarmBot - Open-Source CNC Farming," 2022. https://farm.bot/.

[2] IEEE, "ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering," *ISO/IEC/IEEE 29148:2018(E)*, Nov. 2018.

3. Glossary

Admin The administration of a business, organization, etc.

HTTP Hypertext Transfer Protocol, the protocol used for transmitting data on the World Wide Web..

MQTT Message Queuing Telemetry Transport..

OS Operating System, software that manages computer hardware and provides services for computer programs..

UI User Interface, the interface through which users interact with a software system..

4. Architectural Views

4.1 Context View

4.1.1 Stakeholders' uses of this view

FarmBot's comprehensive context view, which integrates software, hardware, and data analysis, offers specific benefits to each stakeholder group:

FarmBot Users: Benefit from an intuitive and reliable system that simplifies gardening tasks. The easy-to-use interface and accurate feedback on plant health and growth enhance user satisfaction and efficiency in managing their gardens.

Agricultural Experts: Gain access to a scientifically accurate system that supports advanced farming strategies. The inclusion of best practices for irrigation, soil management, and plant maintenance, along with robust data analysis tools, helps optimize crop health and yield.

System Admins: Utilize powerful administration tools that facilitate the installation, configuration, and maintenance of the FarmBot system. Enhanced performance tracking, user account management, and robust security measures ensure smooth and secure operation.

Software Developers: Work within a scalable and extensible framework with standardized development procedures and modular architecture. Well-documented code and support for third-party integrations foster innovation and continuous improvement of FarmBot's capabilities.

Regulatory Authorities: Benefit from FarmBot's adherence to agricultural technol-

ogy and data protection laws. The system's transparency in data collection, processing, and retention, combined with compliance with safety regulations and environmental impact assessments, ensures regulatory compliance and safeguards user privacy.

Hardware Manufacturers: Find value in FarmBot's adaptability and interoperability with their products. Standardized hardware interfaces and opportunities for collaboration enhance integration, enabling manufacturers to optimize their hardware designs for improved compatibility and performance with FarmBot systems.

By addressing the unique concerns of each stakeholder group, FarmBot's context view ensures a holistic approach to urban agriculture, fostering innovation and efficiency across the entire ecosystem

4.1.2 Context Diagram

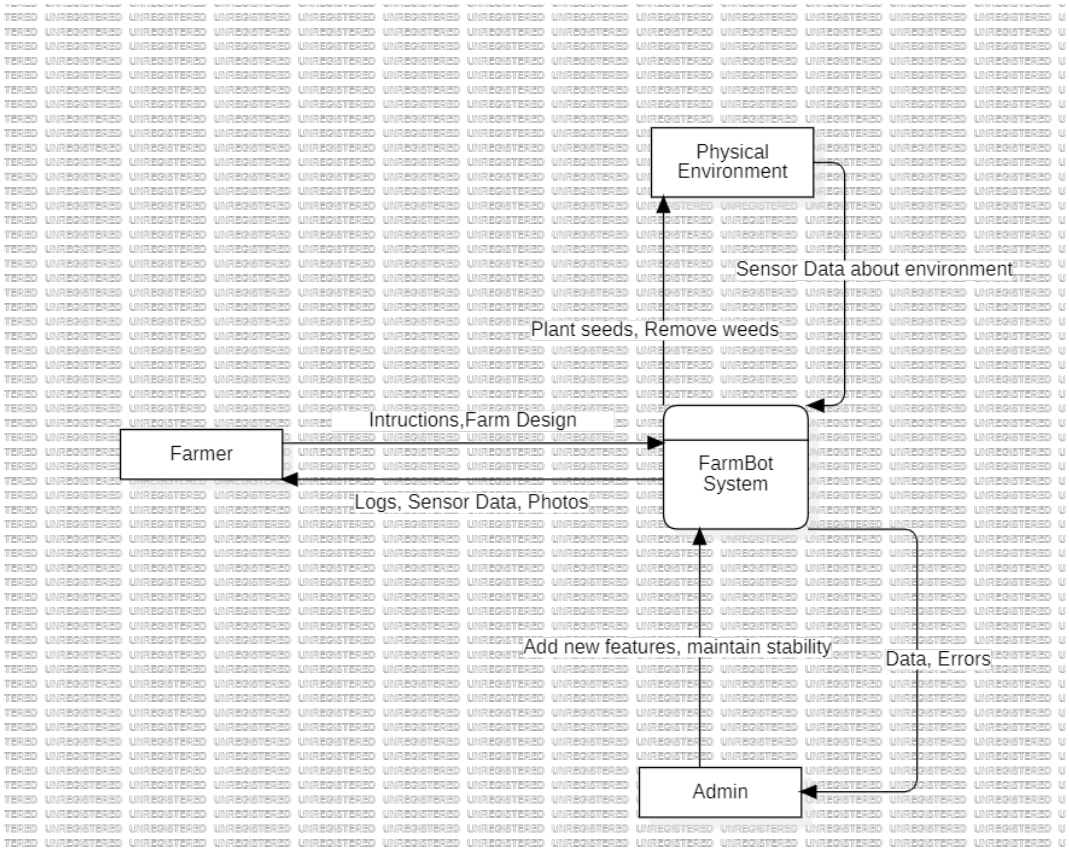

Figure 4.1: Context Diagram

This context diagram depicts the interactions between the Farmbot system and the external entities that influence its operation.

The Farmbot receives sensor data (temperature, humidity, etc.) from the physical environment to monitor growing conditions. In return, the Farmbot can plant seeds and remove weeds to manipulate the environment.

The Farmer provides instructions (planting schedules, watering needs) and the overall farm design layout to the Farmbot. In return, the Farmbot keeps the Farmer informed with logs, sensor data readings, and even captures photos for remote monitoring.

The Admin plays a maintenance role, adding new features and ensuring system stability. The Farmbot provides data and reports any errors encountered during operation to the

Admin for troubleshooting and improvement.

In short, the Farmbot acts as an automated farm assistant, receiving instructions and sensor data, and taking actions to manage the farm environment based on the Farmer's input and its own analysis. The Admin ensures the smooth operation of the system through updates and maintenance.

4.1.3 External Interfaces

Interfaces for cloud integration, control software communication, user engagement, and maintenance assistance are all included in the FarmBot system. Users can monitor status and control the system with the help of the User Interface (UI). The UI and control software can communicate more easily thanks to the Control Software Interface. For data interchange and updates, the system is integrated with cloud-based services via the Cloud Service Interface. Access to assistance and troubleshooting services is made possible through the Maintenance Support Interface. These interfaces together ensure the FarmBot system operates smoothly and is easily managed.

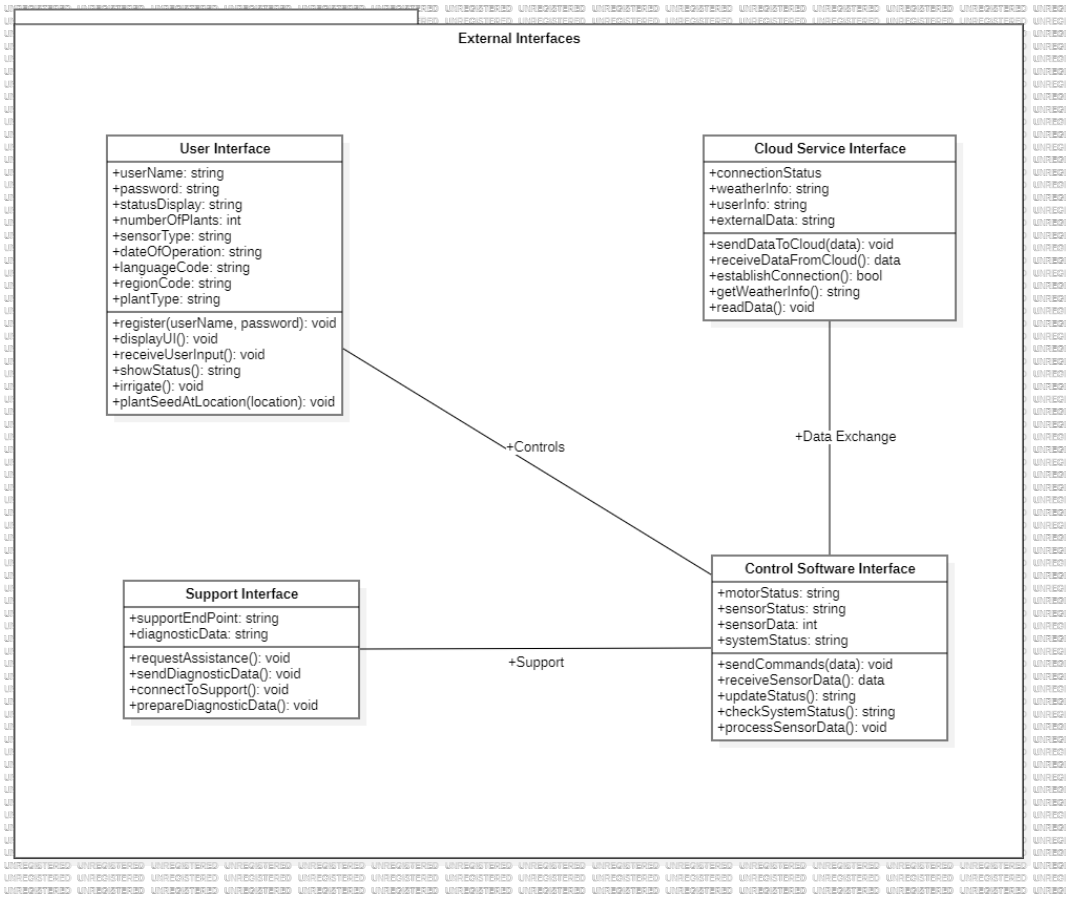

Figure 4.2: External Interface Diagram

User Interface

- **register(userName, password):** Registers a new user with a username and password.
- **displayUI():** Displays the user interface.
- **receiveUserInput():** Handles user input received through the UI.
- **showStatus():** Shows the current status of the system.
- **irrigate():** Initiates the irrigation process.
- **plantSeedAtLocation(location):** Plants a seed at a specified location.

Support Interface

- **requestAssistance():** Requests assistance from the support team.
- **sendDiagnosticData():** Sends diagnostic data to support.
- **connectToSupport():** Connects to the support system.
- **prepareDiagnosticData():** Prepares diagnostic data for transmission.

Cloud Service Interface

- **sendDataToCloud(data):** Sends data to the cloud.
- **receiveDataFromCloud():** Receives data from the cloud.
- **establishConnection():** Establishes a connection to the cloud.
- **getWeatherInfo():** Retrieves weather information.
- **readData():** Reads data from the cloud.

Control Software Interface

- **sendCommands(data):** Sends commands to the control system.
- **receiveSensorData():** Receives data from sensors.
- **updateStatus():** Updates the status of the system.
- **checkSystemStatus():** Checks the system's status.
- **processSensorData():** Processes data received from sensors.

4.1.4 Interaction scenarios

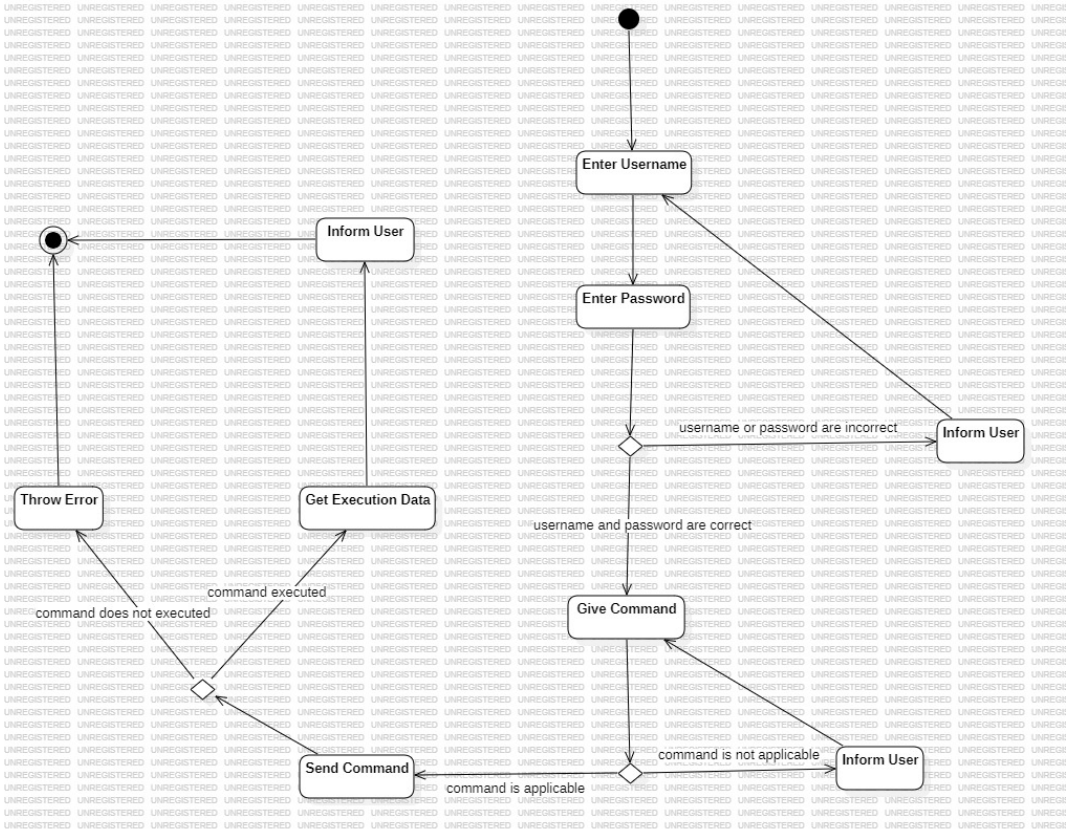

Figure 4.3: User Interface Activity Diagram

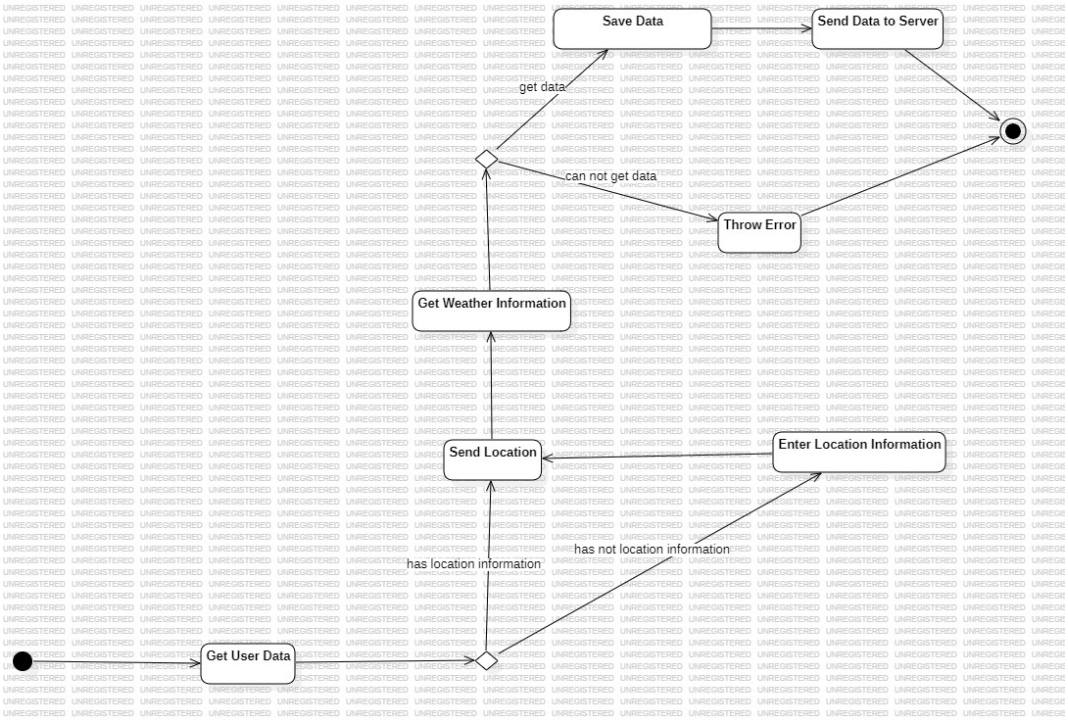

Figure 4.4: Data Collection via Cloud Service Interface Activity Diagram

4.2 Functional View

4.2.1 Stakeholders' uses of this view

The Functional View benefits various Farmbot stakeholders:

Users: Understand how the system translates their actions into commands for the hardware, fostering trust in its effectiveness.

Experts: Assess how Farmbot aligns with best practices and identify areas for potential development.

Admins: Gain insights for troubleshooting, maintaining performance, and managing user accounts.

Developers: Visualize the system's architecture to make improvements, ensure code maintainability, and evaluate third-party integrations.

Regulatory bodies: Understand data collection and processing to assess compliance

with data protection regulations.

Hardware Manufacturers: Ensure their products adhere to standards for seamless integration and optimal performance within the Farmbot system.

The Functional View provides a shared understanding of how Farmbot functions, empowering stakeholders to effectively interact with and contribute to the system's success.

4.2.2 Component Diagram

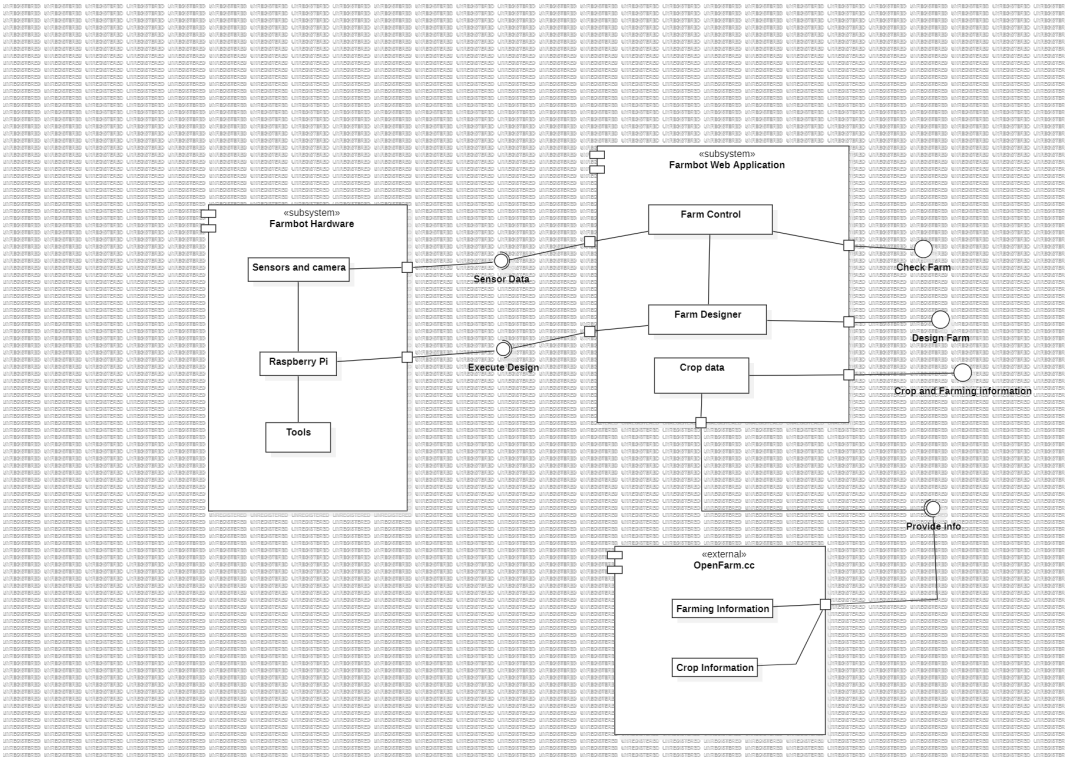

Figure 4.5: Component Diagram

The Farmbot consists of 2 subsystems the Hardware which is the express model for our case and the Farmbot web application. It also uses external component OpenFarm.cc for external data.

Farmbot Hardware consists of 3 basic parts. Sensors and camera, Raspberry pi and tools. These parts all interact with the Web application. The web application consists

of Farm Designer, Farm control and Crop data. The user can design a farm using the web app and then execute its design after connecting it to the farmbot device. The web app interacts with the raspberry pi for this which acts as the bridge between the two. The user can also check the state of the farm by interacting with the web app which then gets sensor data from the device.

Finally the user can get data in the web app about crops and farming guides. This is done by the connection of the web app to openfarm.cc. The user then uses this data to design its farm, select what to plant or learn how to plant it.

4.2.3 Internal Interfaces

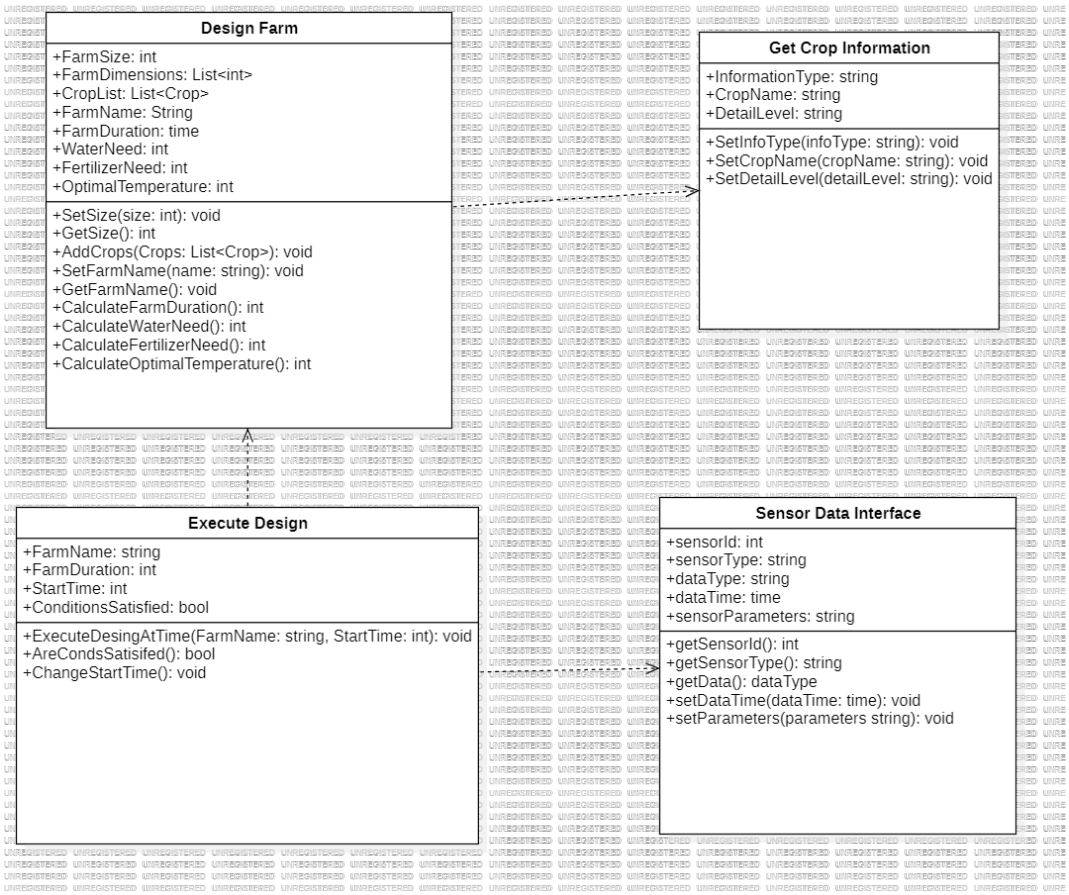

Figure 4.6: Internal Interfaces Class Diagram

- **Design Farm**
  - `+SetSize(size: int): void`

    Sets the size of the farm.
  - `+GetSize(): int`

    Retrieves the size of the farm.
  - `+AddCrops(Crops: List<Crop>): void`

    Adds a list of crops to the farm.
  - `+SetFarmName(name: string): void`

    Sets the name of the farm.
  - `+GetFarmName(): void`

    Retrieves the name of the farm.
  - `+CalculateFarmDuration(): int`

    Calculates the duration required for the farm operations.
  - `+CalculateWaterNeed(): int`

    Calculates the amount of water needed for the farm.
  - `+CalculateFertilizerNeed(): int`

    Calculates the amount of fertilizer needed for the farm.
  - `+CalculateOptimalTemperature(): int`

    Calculates the optimal temperature for the farm.
- **Get Crop Information**
  - `+SetInfoType(infoType: string): void`

    Sets the type of information to retrieve.
  - `+SetCropName(cropName: string): void`

    Sets the name of the crop to retrieve information about.
  - `+SetDetailLevel(detailLevel: string): void`

    Sets the detail level of the information to retrieve.

- **Execute Design**
  - `+ExecuteDesignAtTime(FarmName: string, StartTime: int): void`

    Executes the farm design at a specified start time.
  - `+AreCondsSatisfied(): bool`

    Checks if the conditions are satisfied for the farm design execution.
  - `+ChangeStartTime(): void`

    Changes the start time for the farm design execution.
- **Sensor Data Interface**
  - `+getSensorId(): int`

    Retrieves the sensor ID.
  - `+getSensorType(): string`

    Retrieves the sensor type.
  - `+getData(): dataType`

    Retrieves the data from the sensor.
  - `+setDateTime(dataTime: time): void`

    Sets the date and time for the sensor data.
  - `+setParameters(parameters: string): void`

    Sets the parameters for the sensor.

4.2.4 Interaction Patterns

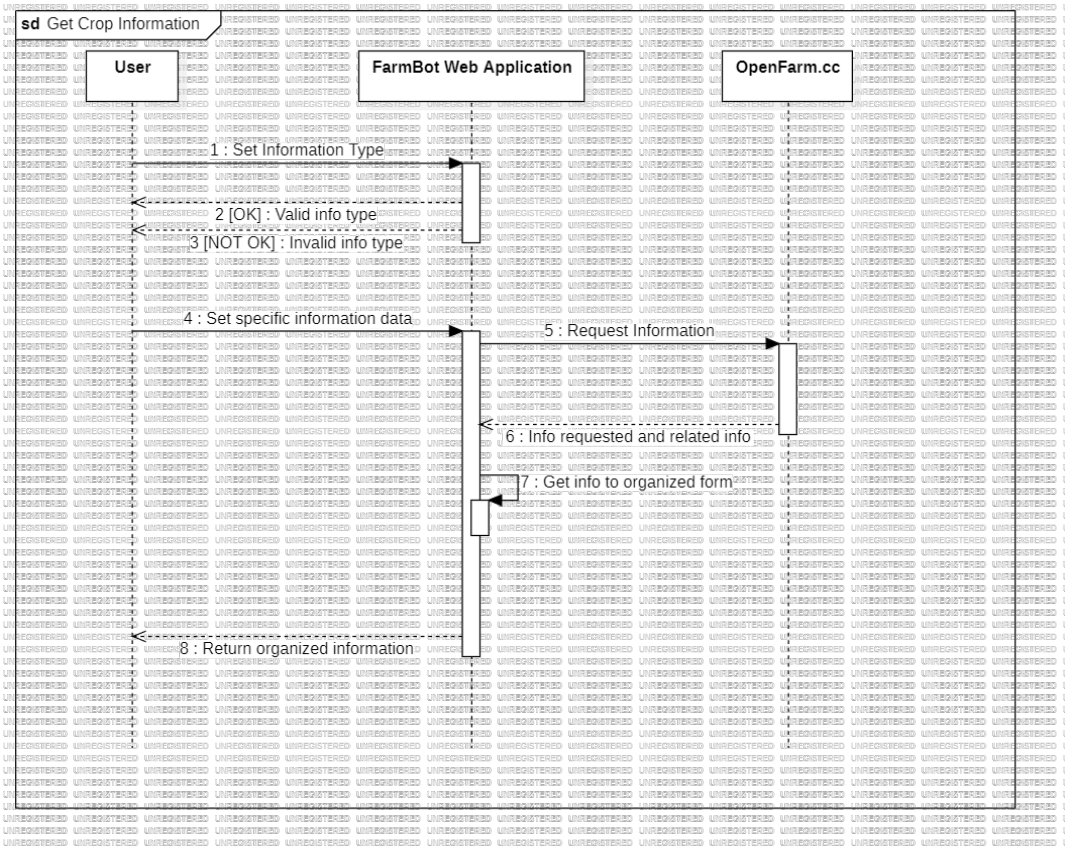

Figure 4.7: Sequence Diagram for Get Crop Information

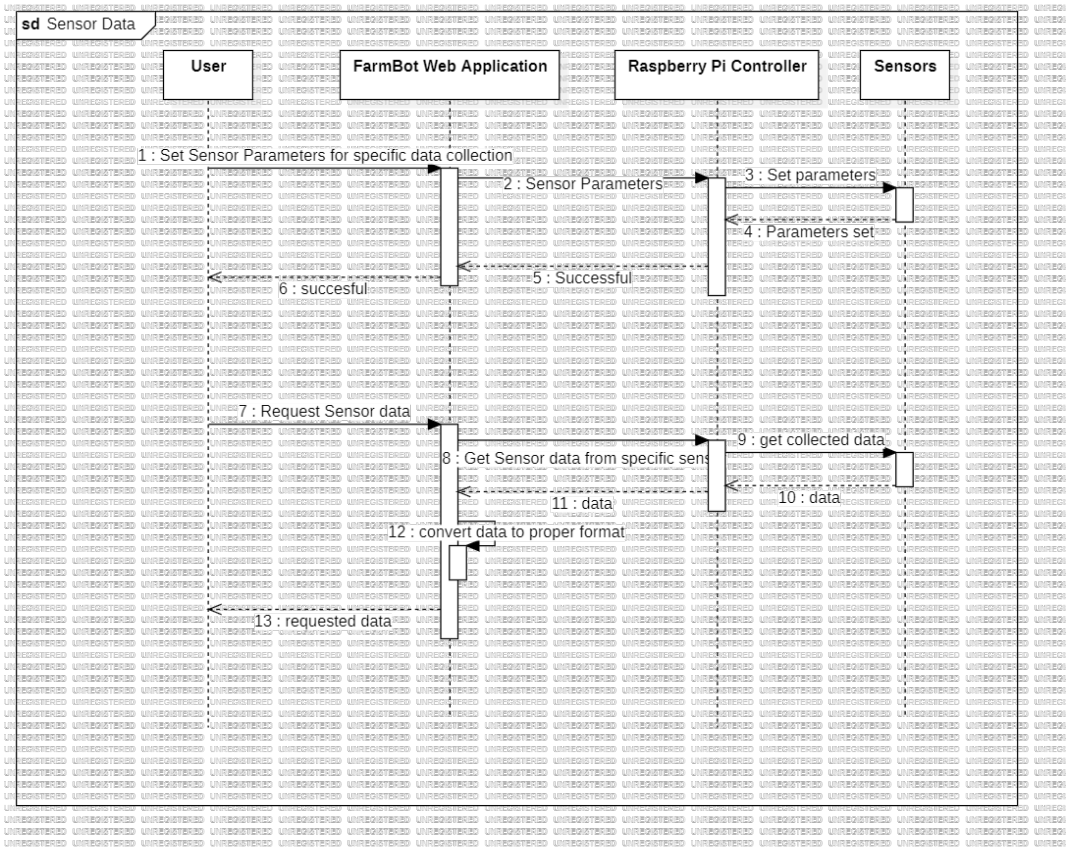

Figure 4.8: Sequence Diagram for Sensor Data

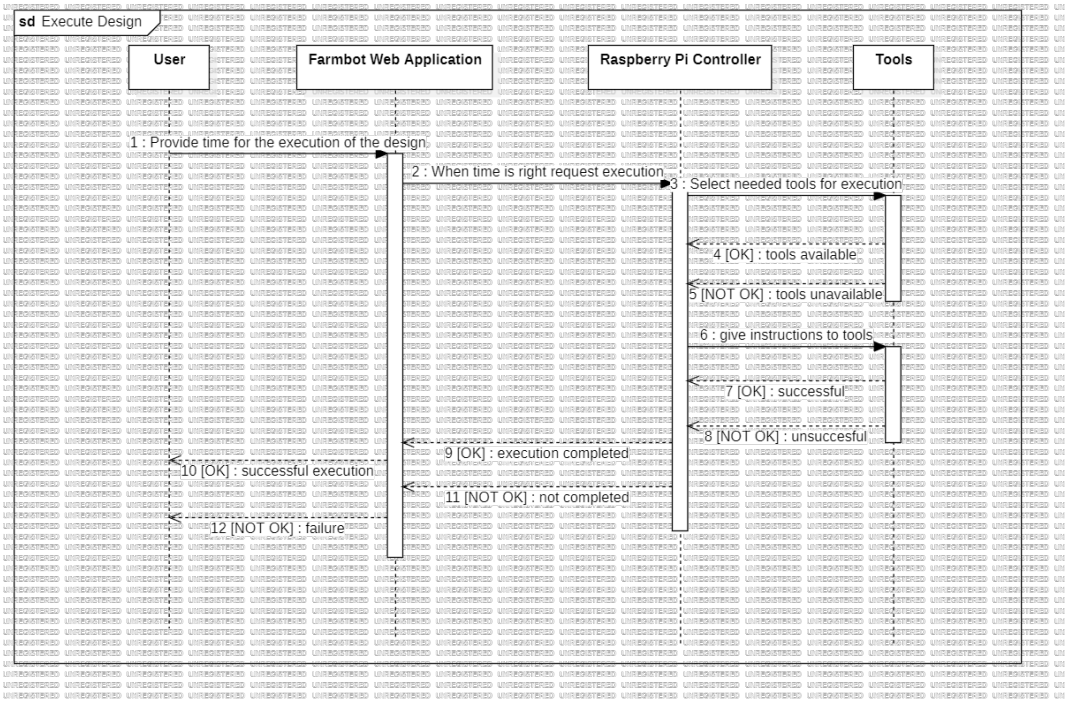

Figure 4.9: Sequence Diagram for Execute Design

4.3 Information View

4.3.1 Stakeholders' uses of this view

FarmBot Users:

Users utilize the Information View to access user-friendly interfaces and receive quick feedback on plant health and growth. They rely on intuitive dashboards and reports to manage gardening tasks effectively.

Agricultural Experts:

Agricultural experts leverage the Information View to access scientific data and analysis tools for optimizing farming strategies. They rely on comprehensive reports and data visualization features to implement best practices in irrigation, soil management, and plant maintenance.

System Admins:

System administrators depend on the Information View to monitor system performance and manage user accounts efficiently. They access system logs, error reports, and user management tools to ensure smooth operation and resolve issues promptly.

Software Developers:

Developers use the Information View to access documentation, code repositories, and development tools for enhancing FarmBot software. They rely on version control systems, API documentation, and collaboration platforms to maintain code quality and support future improvements.

Regulatory Authorities:

Regulatory authorities utilize the Information View to access data privacy policies, compliance reports, and audit trails for ensuring regulatory adherence. They rely on transparency reports, data processing logs, and compliance documentation to evaluate FarmBot's compliance with legal and regulatory requirements.

Hardware Manufacturers:

Hardware manufacturers access the Information View to understand hardware specifications, compatibility requirements, and integration guidelines for FarmBot systems. They rely on hardware documentation, interface specifications, and compatibility matrices to ensure seamless integration and optimize hardware designs for FarmBot compatibility and performance.

4.3.2 Database Class Diagram

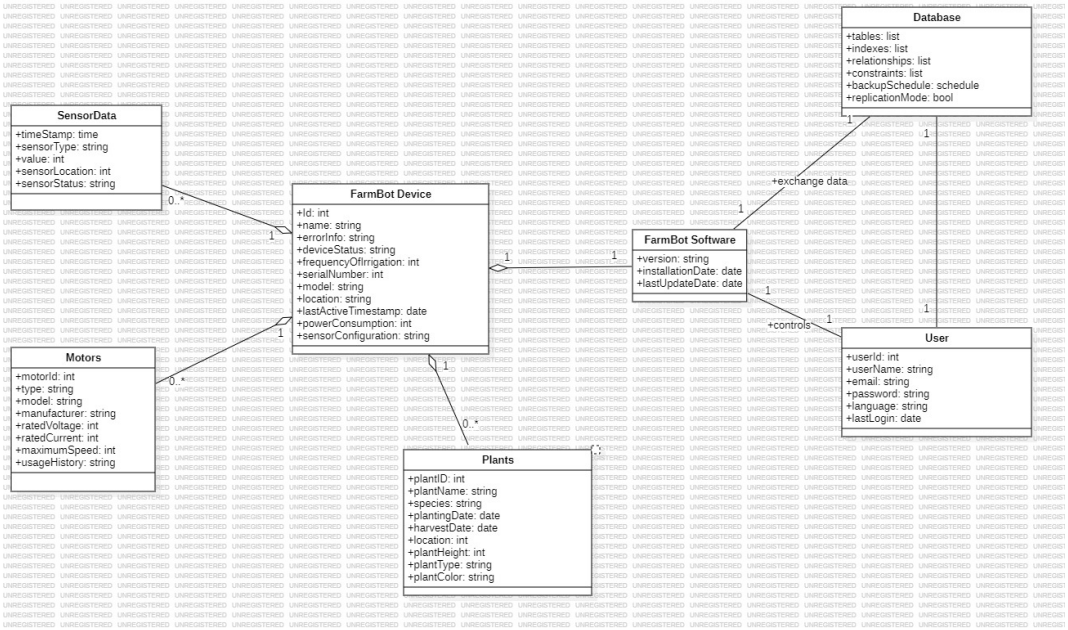

Figure 4.10: Database Diagram

FarmBot Device:

- **name**: Likely the name or identifier of the FarmBot device.
- **errorInfo**: Possibly stores information about any errors or issues encountered by the FarmBot device.
- **deviceStatus**: Indicates the current status or state of the FarmBot device.
- **frequencySwitchPosition**: May refer to the position or setting of a frequency switch, possibly related to communication or control.
- **serialNumber**: The unique serial number assigned to the FarmBot device.
- **model**: The specific model or variant of the FarmBot device.
- **lastDataTimestamp**: The timestamp of the last data received from or sent by the FarmBot device.

- **powerConsumption**: The amount of power consumed by the FarmBot device.
- **sensorConfiguration**: Likely contains configuration settings related to the sensors used by the FarmBot device.

FarmBot Software:

- **version**: The version of the FarmBot software running on the device.
- **installationDate**: The date when the FarmBot software was installed or deployed.
- **lastUpdateDate**: The date when the FarmBot software was last updated or patched.
- **controls**: Possibly refers to the control mechanisms or functions provided by the FarmBot software.

Database:

- **tables**: A list of tables present in the database.
- **indexes**: A list of indexes defined in the database.
- **relationships**: A list of relationships or associations between tables in the database.
- **constraints**: A list of constraints (e.g., primary keys, foreign keys) defined in the database.
- **backupSchedule**: The schedule or frequency for backing up the database.
- **replicationMode**: Indicates whether the database is configured for replication or not.

User:

- **userId**: A unique identifier for the user.
- **userName**: The username or login name of the user.

- **email**: The email address associated with the user.
- **password**: The password used for authentication by the user.
- **isDatabase**: It's unclear what this attribute represents, but it might be a flag indicating if the user has database privileges or not.
- **lastLogin**: The timestamp of the last login or access by the user.

4.3.3 Operations on Data

Operation	**Create**	**Read**	**Update**	**Delete**
createPlant	Add a new plant entry.	-	-	-
getPlants	-	Retrieve plant details.	-	-
updatePlant	-	-	Update plant information (e.g., growth stage, health status).	-
deletePlant	-	-	-	Remove plant entry.
createTaskSchedule	Create a new task schedule.	-	-	-
getTaskSchedules	-	View task schedules.	-	-
updateTaskSchedule	-	-	Modify existing task schedules.	-
deleteTaskSchedule	-	-	-	Delete a task schedule.

Operation	**Create**	**Read**	**Update**	**Delete**
createSensorData	Record new sensor data (e.g., soil moisture, temperature).	-	-	-
getSensorData	-	Retrieve sensor data.	-	-
updateSensorCalibration	-	-	Update sensor calibration settings.	-
deleteSensorData	-	-	-	Delete sensor data records.
createUserProfile	Register a new user.	-	-	-
getUserProfiles	-	Retrieve user profiles.	-	-
updateUserProfile	-	-	Update user details (e.g., email, preferences).	-
deleteUserProfile	-	-	-	Remove user profile.
createLogEntry	Create a new log entry.	-	-	-
getLogEntries	-	Access log entries.	-	-
updateLogEntry	-	-	Update log information.	-
deleteLogEntry	-	-	-	Delete log entries.

Operation	Create	Read	Update	Delete
createWateringSchedule	Create a new watering schedule.	-	-	-
getWateringSchedules	-	View watering schedules.	-	-
updateWateringSchedule	-	-	Update watering schedules.	-
deleteWateringSchedule	-	-	-	Delete watering schedules.
createSystemSetting	Add new system settings.	-	-	-
getSystemSettings	-	Retrieve system settings.	-	-
updateSystemSetting	-	-	Modify system settings (e.g., network configurations).	-
deleteSystemSetting	-	-	-	Delete system settings.

Table 4.1: CRUD Operations for FarmBot Database

4.4 Deployment View

4.4.1 Stakeholders' uses of this view

As it is defined in 1.3, the system has 6 different stakeholders which are FarmBot Users, Agricultural Experts, System Admins, Software Developers, Regulatory Au-

thorities,Hardware Manufacturers which can be described as follows:

FarmBot Users: Users rely on the Deployment View to understand the hardware and connectivity requirements for accessing FarmBot's functionalities. They seek clarity on device compatibility and network configurations to ensure seamless integration into their gardening routines.

Agricultural Experts: Experts utilize the Deployment View to assess FarmBot's deployment architecture and understand how it incorporates best practices in irrigation, soil management, and plant care. They seek insights into the system's deployment environment to optimize its performance in supporting various farming strategies.

System Admins: System administrators leverage the Deployment View to plan and manage the deployment of FarmBot systems. They analyze hardware configurations, estimate costs, and assess maintainability requirements to ensure efficient deployment and operation of the system.

Software Developers: Developers refer to the Deployment View to understand the system's architecture and deployment environment. They use this information to optimize code designs for hardware compatibility, scalability, and maintainability, ensuring smooth integration and future development of FarmBot software.

Regulatory Authorities: Regulatory authorities review the Deployment View to ensure FarmBot's compliance with agricultural and data protection regulations. They examine deployment details to assess data handling practices, security measures, and adherence to relevant laws and guidelines.

Hardware Manufacturers: Hardware manufacturers explore the Deployment View to understand FarmBot's compatibility with their products. They seek information on

hardware interfaces and deployment standards to ensure seamless integration and optimize hardware designs for FarmBot compatibility and performance.

4.4.2 Deployment Diagram

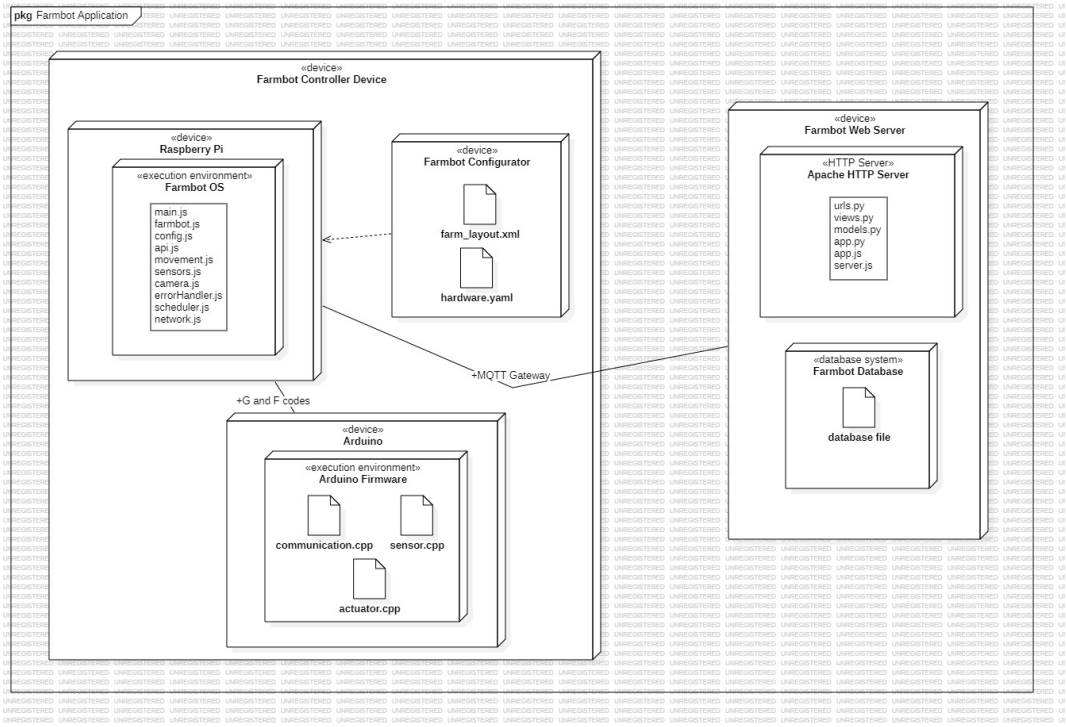

Figure 4.11: Deployment Diagram

The deployment architecture of the FarmBot application consists of three main components:

Raspberry Pi Device

- Runs the FarmBot OS as the execution environment.
- The FarmBot OS includes various JavaScript files (main.js, config.js, app.js, movement.js, sensors.js, camera.js, errorHandler.js, scheduler.js, network.js) that handle different functionalities of the FarmBot application.

FarmBot Controller Device

- Contains the FarmBot Configurator, which manages configuration files like farm_layout.xml and hardware.yaml.
- The FarmBot Configurator communicates with the MQTT Gateway to send configuration data to the Arduino device.

Arduino Device

- Runs the Arduino Firmware as the execution environment.
- The Arduino Firmware includes C++ files for communication (communication.cpp), sensors (sensor.cpp), and actuators (actuator.cpp).
- The Arduino device, along with the Raspberry Pi, handles the physical control and operation of the FarmBot hardware.

FarmBot Web Server

- Runs an Apache HTTP Server.
- Includes Python files for handling URLs (urls.py), views (views.py), models (models.py), APIs (app.py), and serving static files (server.js).
- Provides the web interface and API for interacting with the FarmBot application.

Database System

- Stores data related to the FarmBot application in a database file.

The FarmBot application follows a distributed architecture, where the Raspberry Pi and Arduino devices handle the low-level control and operations, while the Web Server provides the user interface and API for configuration and monitoring. The FarmBot Configurator acts as an intermediary component, translating the user configurations into instructions for the Arduino device via the MQTT Gateway.

4.5 Design Rationale

Context View Rationale

In order to show how the FarmBot system interacts with its external entities—such as users, administrators, and other entities—the Context View was created. Stakeholders can better grasp how FarmBot works with other systems and how users go about their daily lives using this view. We guarantee that all required interfaces and dependencies are taken into account by locating and mapping these external interactions, which promotes a more seamless integration and improved user experience.

Functional View Rationale

The FarmBot system's internal operations are the primary focus of the Functional View, which describes all of the parts and how they interact. This point of view is justified by the need to guarantee the modularity and scalability of the system design. Our ability to clearly define provides/requires links between components facilitates future upgrades and easy maintenance. This perspective aids stakeholders in comprehending how various system components work together to accomplish challenging tasks, guaranteeing reliable and effective operations.

Information View Rationale

The purpose of the Information View was to offer a general description of the database design and data structures that the FarmBot system used. For those involved in data integrity, efficient storage, and retrieval procedures, this perspective is essential. We make sure that the data administration of the system is efficient and well-organized, which supports the overall dependability and effectiveness of the system, by outlining the important database objects and their relationships.

Deployment View Rationale

The improved Deployment View includes the Smart Watch Notification Center and iOS platform, enhancing FarmBot's architecture. These additions provide real-time monitoring, notifications, and accessibility through Apple devices. The enhanced deployment ensures better performance, reliability, and security, meeting stakeholder

needs efficiently.

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

5.1 Context View

5.1.1 Stakeholders' uses of this view

FarmBot's comprehensive context view, which integrates software, hardware, and data analysis, offers specific benefits to each stakeholder group:

FarmBot Users: Benefit from an intuitive and reliable system that simplifies gardening tasks. The easy-to-use interface and accurate feedback on plant health and growth enhance user satisfaction and efficiency in managing their gardens.

Agricultural Experts: Gain access to a scientifically accurate system that supports advanced farming strategies. The inclusion of best practices for irrigation, soil management, and plant maintenance, along with robust data analysis tools, helps optimize crop health and yield.

System Admins: Utilize powerful administration tools that facilitate the installation, configuration, and maintenance of the FarmBot system. Enhanced performance tracking, user account management, and robust security measures ensure smooth and secure operation.

Software Developers: Work within a scalable and extensible framework with stan-

dardized development procedures and modular architecture. Well-documented code and support for third-party integrations foster innovation and continuous improvement of FarmBot's capabilities.

Regulatory Authorities: Benefit from FarmBot's adherence to agricultural technology and data protection laws. The system's transparency in data collection, processing, and retention, combined with compliance with safety regulations and environmental impact assessments, ensures regulatory compliance and safeguards user privacy.

Hardware Manufacturers: Find value in FarmBot's adaptability and interoperability with their products. Standardized hardware interfaces and opportunities for collaboration enhance integration, enabling manufacturers to optimize their hardware designs for improved compatibility and performance with FarmBot systems.

By addressing the unique concerns of each stakeholder group, FarmBot's context view ensures a holistic approach to urban agriculture, fostering innovation and efficiency across the entire ecosystem.

5.1.2 Context Diagram

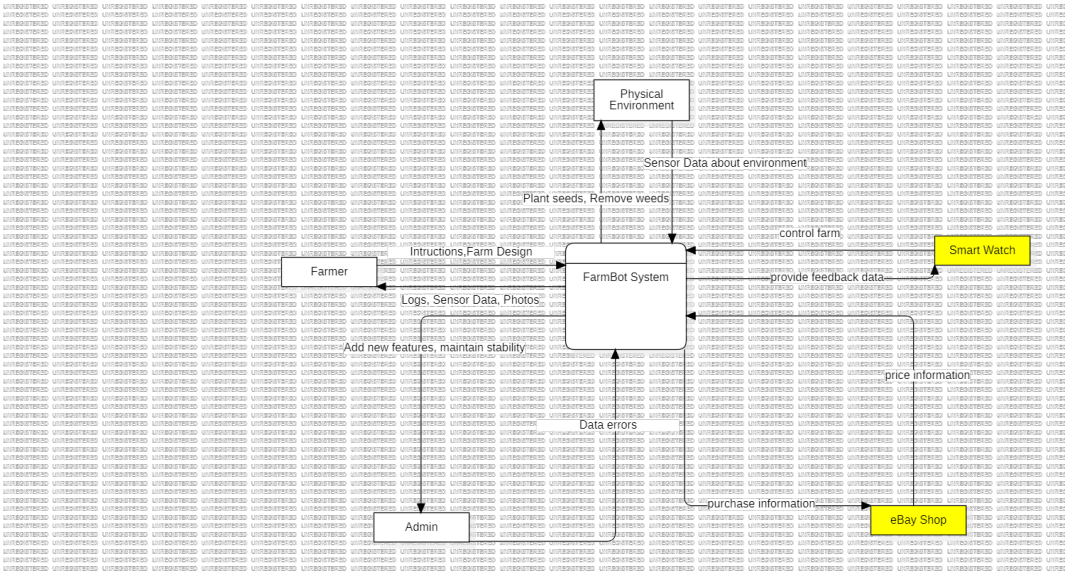

Figure 5.1: Sequence Diagram for Sensor Data

The enhanced FarmBot system expands its reach with optional smartwatch control and eBay price lookups for informed resource management. It seamlessly integrates with AgriCloud Services for weather forecasts and soil analysis, while the web dashboard empowers farm managers with advanced analytics for data-driven decision-making.

5.1.3 External Interfaces

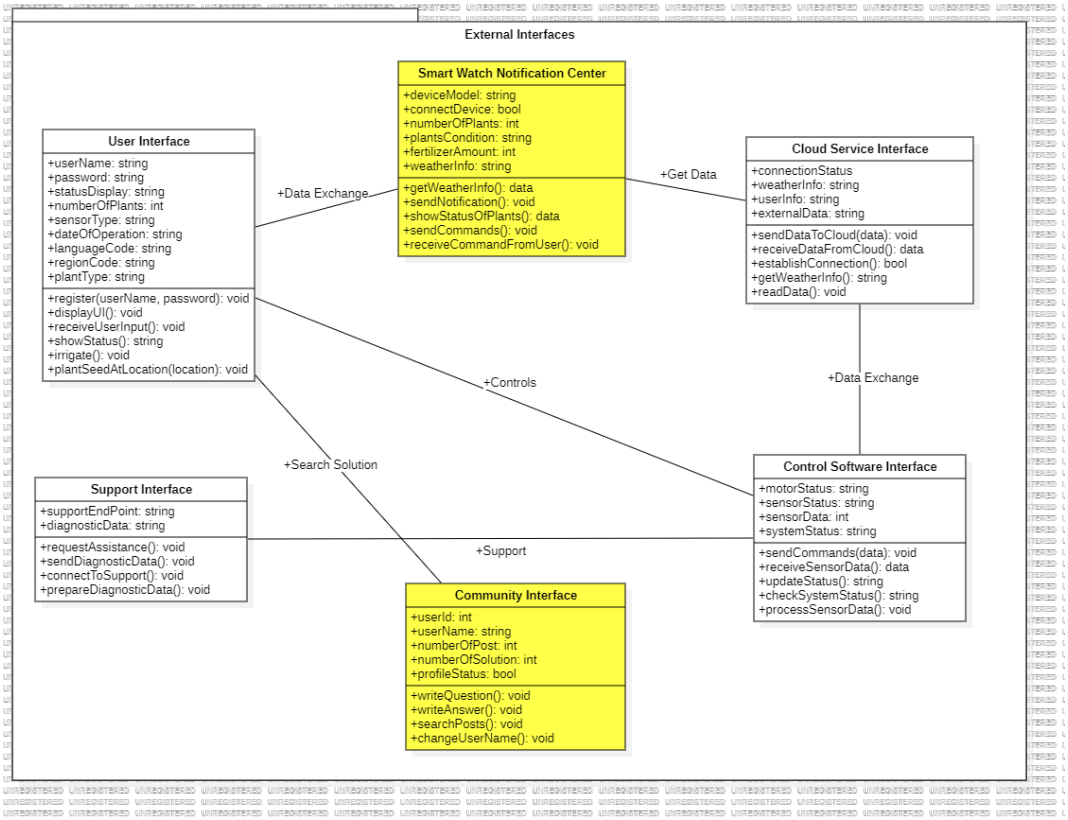

Figure 5.2: Improved External Interface Diagram

Smart Watch Notification Center

- **getWeatherInfo():** Retrieves weather information.
- **sendNotification():** Sends notifications to the user.
- **showStatusOfPlants():** Displays the status of the plants.

- **sendCommands():** Sends commands to the system.
- **receiveCommandFromUser():** Receives commands from the user.

Community Interface

- **writeQuestion():** Allows the user to write a question.
- **writeAnswer():** Allows the user to write an answer.
- **searchPosts():** Enables searching for posts.
- **changeUserName():** Allows the user to change their username.

5.1.4 Interaction scenarios

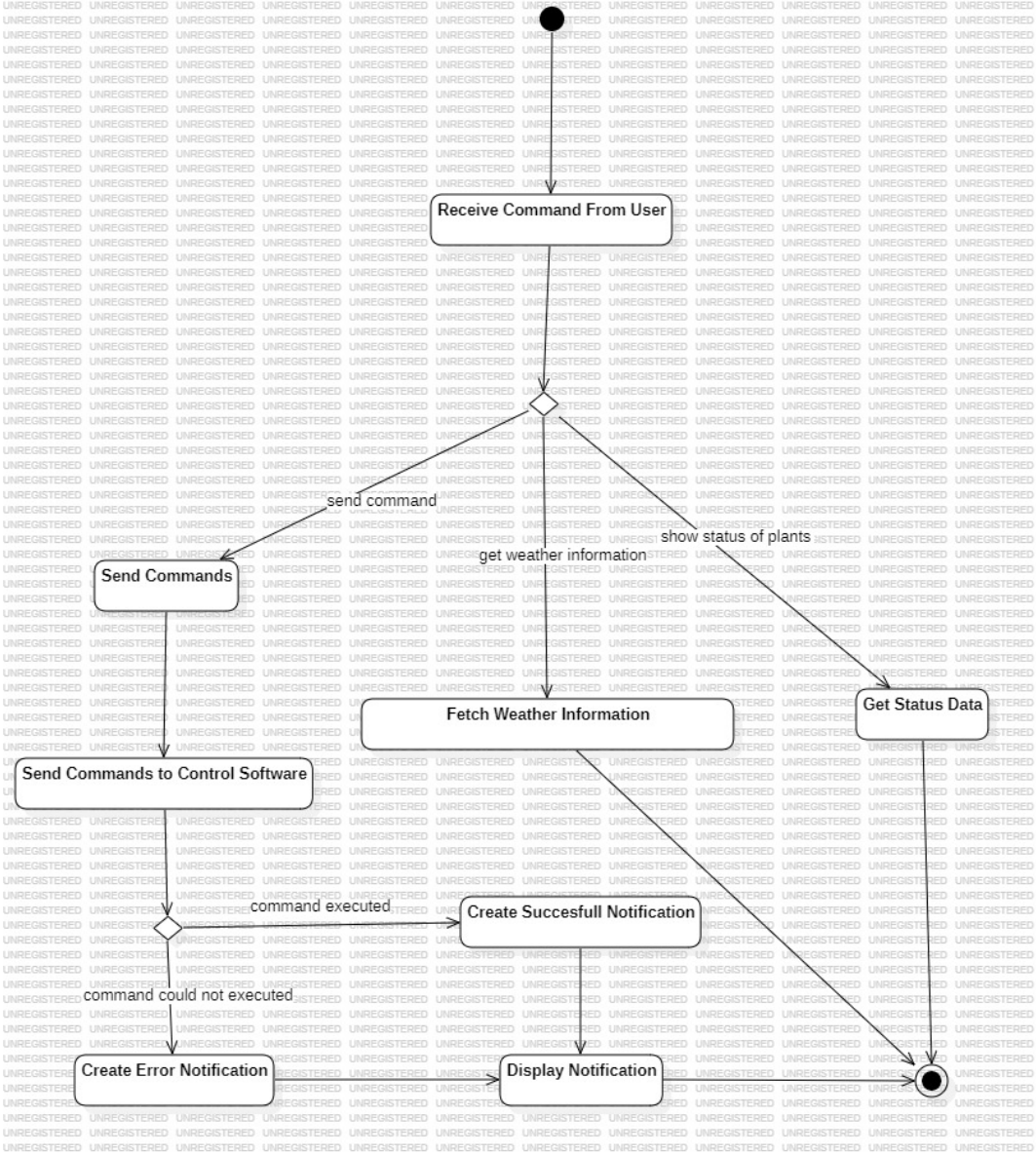

Figure 5.3: Smart Watch Notification Center Activity Diagram

5.2 Functional View

5.2.1 Stakeholders' uses of this view

The Functional View benefits various Farmbot stakeholders:

Users: Understand how the system translates their actions into commands for the hardware, fostering trust in its effectiveness.

Experts: Assess how Farmbot aligns with best practices and identify areas for potential development.

Admins: Gain insights for troubleshooting, maintaining performance, and managing user accounts.

Developers: Visualize the system's architecture to make improvements, ensure code maintainability, and evaluate third-party integrations.

Regulatory bodies: Understand data collection and processing to assess compliance with data protection regulations.

Hardware Manufacturers: Ensure their products adhere to standards for seamless integration and optimal performance within the Farmbot system.

The Functional View provides a shared understanding of how Farmbot functions, empowering stakeholders to effectively interact with and contribute to the system's success.

5.2.2 Component Diagram

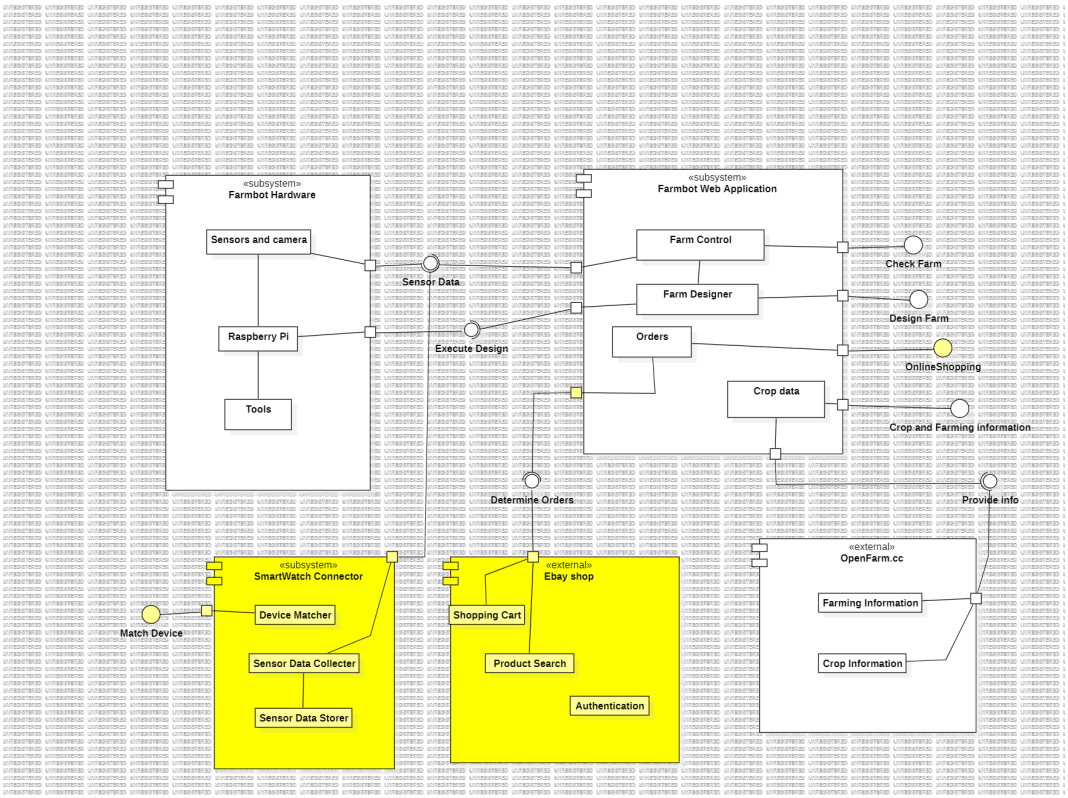

Figure 5.4: Improved Component Diagram

One of the key improvements is the introduction of the "SmartWatch Connector" component, which suggests the ability to connect and interface with smartwatch devices. This could potentially enable the system to gather data using wearable devices, expanding the access of the range of sensors and data sources available. Furthermore the user can access and store sensor data using its smartwatch thanks to the smartwatch connector component.

Furthermore, another external component Ebay shop component is introduced. Suggesting the integration of an e-commerce platform or marketplace functionality. This could enable users to purchase products or services related to farming or the system itself. The "Product Search" and "Authentication" parts are also included in the im-

proved diagram, indicating the addition of search capabilities and user authentication mechanisms for improved usability and security.

Overall, the improvements highlighted in yellow appear to focus on enhancing data collection, device integration, e-commerce capabilities, and user experience aspects of the system

5.2.3 Internal Interfaces

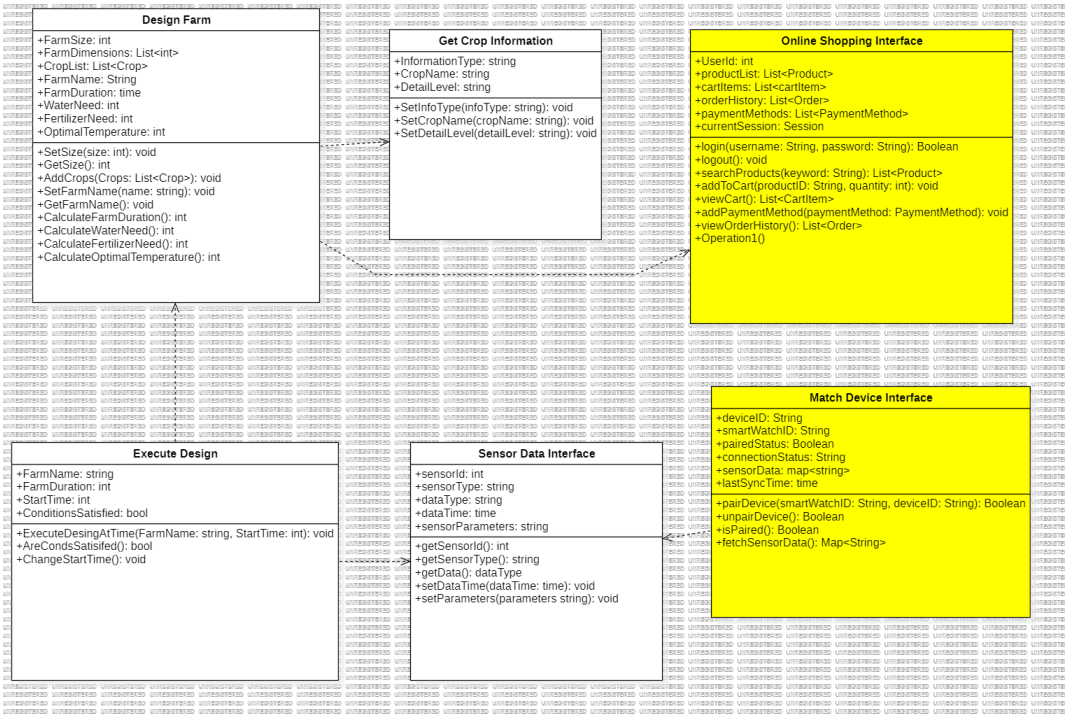

Figure 5.5: Improved Internal Interfaces Diagram

Here are the explanations for the additional operations added in the improved diagram.

Online Shopping Interface

login: *Authenticates a user with a username and password*

logout: *Logs out the current user session*

searchProducts: *Searches for products based on a keyword*

addToCart: *Adds a product to the shopping cart*

viewCart: *Displays the items in the shopping cart*

addPaymentMethod: *Adds a payment method for checkout*

viewOrderHistory: *Shows the user's previous orders*

checkout: *Completes the purchase and payment process*

Match Device Interface

pairDevice: *Pairs a smart watch with a device*

unpairDevice: *Unpairs a previously paired device*

isPaired: *Checks if a device is paired*

fetchSensorData: *Retrieves sensor data from a device*

5.2.4 Interaction Patterns

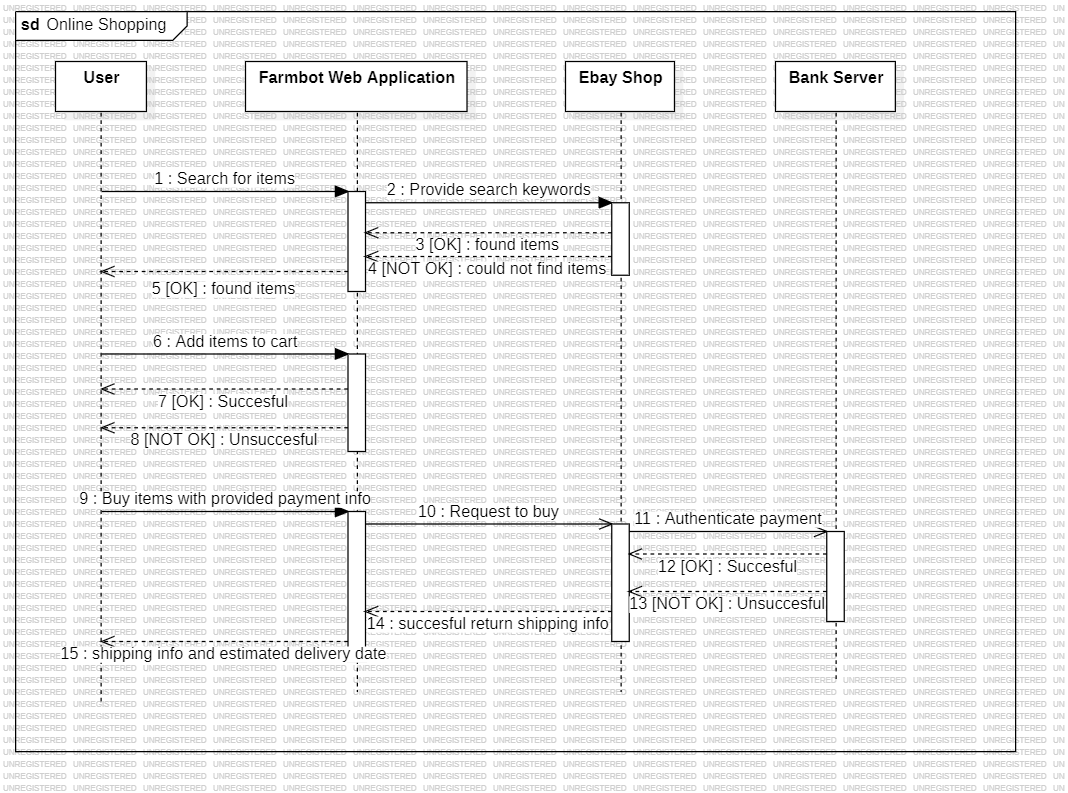

Figure 5.6: Online Shopping Interface Sequence Diagram

5.3 Information View

5.3.1 Stakeholders' uses of this view

FarmBot Users:

Users access user-friendly interfaces and quick feedback on plant health. The Smart Watch Center offers real-time monitoring and notifications, enhancing task management.

Agricultural Experts:

Experts use scientific data and analysis tools for farming strategies. They leverage

community insights for optimizing irrigation, soil management, and plant maintenance.

System Admins:

Admins monitor system performance, manage accounts, and ensure security compliance using logs, error reports, user management tools, and security constraints.

Software Developers:

Developers access documentation, repositories, and development tools, including security constraints. They maintain code quality and support improvements with community feedback.

Regulatory Authorities:

Authorities ensure compliance with data privacy policies, audit trails, and security constraints. They evaluate transparency reports, processing logs, and compliance documentation.

Hardware Manufacturers:

Manufacturers understand hardware specifications, compatibility, and integration guidelines. They utilize documentation and compatibility matrices to optimize designs, ensuring security compliance.

5.3.2 Database Class Diagram

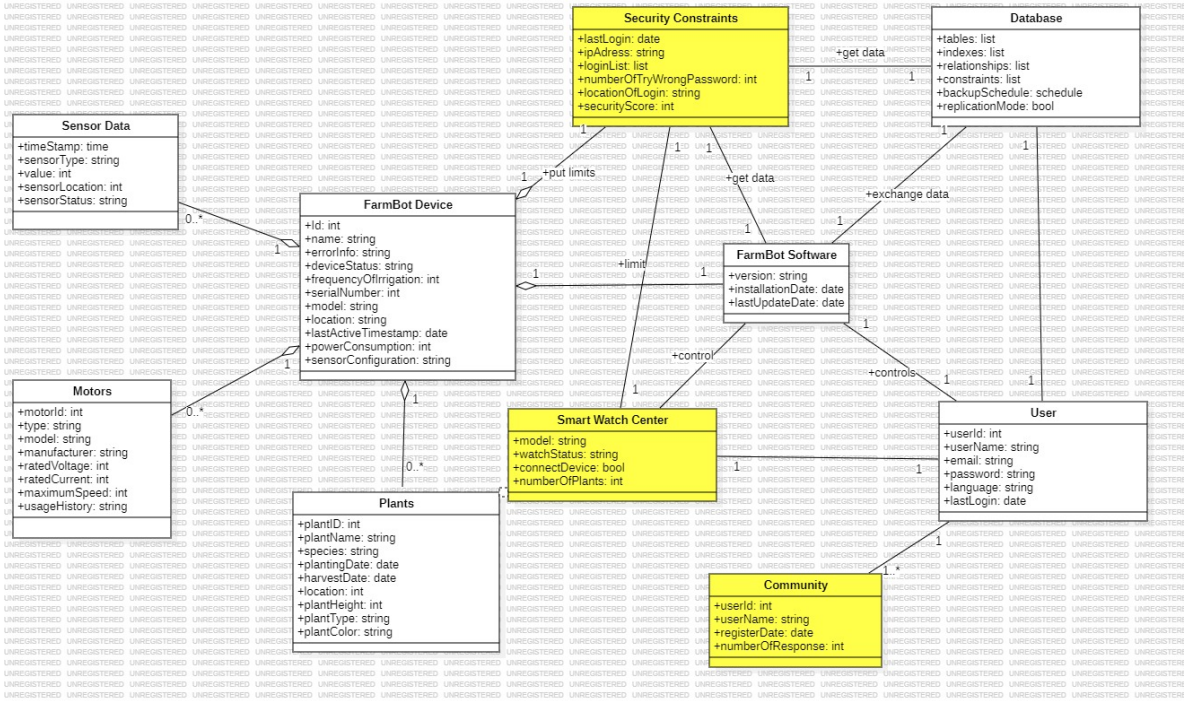

Figure 5.7: Improved Database Class Diagram

5.3.3 Operations on Data

Operation	Create	Read	Update	Delete
logCommand	Create a new log entry for a command.	-	-	-
watchNotification	Create a new notification.	-	-	-
sendNotification	Send a notification to a specific recipient.	-	-	-

Continued on next page

Table 5.1 – continued from previous page

Operation	Create	Read	Update	Delete
issueCommand	Issue a new command to the FarmBot system.	-	-	-
readCommandLog	-	Read and retrieve command log entries based on specific criteria (e.g., timestamp, command type).	-	-
readNotifications	-	Read and retrieve notifications based on specific criteria (e.g., timestamp, type, status).	-	-
getNotificationHistory	-	Retrieve the history of notifications based on specific criteria (e.g., recipient, date range).	-	-

Continued on next page

Table 5.1 – continued from previous page

Operation	Create	Read	Update	Delete
listenForCommands	-	Start listening for incoming commands.	-	-
updateCommandLog	-	-	Update an existing command log entry with new data.	-
updateNotificationStatus	-	-	Update the status of a notification (e.g., read, unread).	-
updateNotificationPreferences	-	-	Update the notification preferences for a recipient.	-
updateCommandParameters		-	Update the parameters of a pending or scheduled command.	-
deleteCommandLog	-	-	-	Delete a specific command log entry.
deleteNotification	-	-	-	Delete a specific notification.

Continued on next page

Table 5.1 – continued from previous page

Operation	Create	Read	Update	Delete
deleteNotificationHistory	-	-	-	Delete notification history based on specific criteria.
clearCommandQueue	-	-	-	Clear the command queue or buffer.

5.4 Deployment View

5.4.1 Stakeholders' uses of this view

As it is defined in 1.3, the system has 6 different stakeholders which are FarmBot Users, Agricultural Experts, System Admins, Software Developers, Regulatory Authorities, and Hardware Manufacturers which can be described as follows:

FarmBot Users: Users rely on the Deployment View to understand the hardware and connectivity requirements for accessing FarmBot's functionalities. They seek clarity on device compatibility, network configurations, and mobile platform integrations (iOS and smartWatchOS) to ensure seamless integration into their gardening routines and enable remote monitoring and control capabilities.

Agricultural Experts: Experts utilize the Deployment View to assess FarmBot's deployment architecture and understand how it incorporates best practices in irrigation, soil management, and plant care. They seek insights into the system's deployment environment, including the integration of sensor data receivers and notification modules on mobile platforms, to optimize its performance in supporting various farming

strategies.

System Admins: System administrators leverage the Deployment View to plan and manage the deployment of FarmBot systems. They analyze hardware configurations, estimate costs, assess maintainability requirements, and evaluate the integration of web server components (logCommand.js, watchNotification.js, Notifier.py, CommandListener.py) to ensure efficient deployment and operation of the system.

Software Developers: Developers refer to the Deployment View to understand the system's architecture and deployment environment. They use this information to optimize code designs for hardware compatibility, scalability, maintainability, and cross-platform support (web, iOS, and smartWatchOS). They also leverage the deployment details of components like Command.js, Notification.kt, App.kt, SmartWatchConnector.kt, and Command.kt to ensure smooth integration and future development of FarmBot software across various platforms.

Regulatory Authorities: Regulatory authorities review the Deployment View to ensure FarmBot's compliance with agricultural and data protection regulations. They examine deployment details, including the handling of sensor data, notifications, and command processing, to assess data handling practices, security measures, and adherence to relevant laws and guidelines.

Hardware Manufacturers: Hardware manufacturers explore the Deployment View to understand FarmBot's compatibility with their products. They seek information on hardware interfaces, deployment standards, and the integration of platform-specific components (iOS and smartWatchOS) to ensure seamless integration and optimize hardware designs for FarmBot compatibility and performance across multiple platforms.

5.4.2 Deployment Diagram

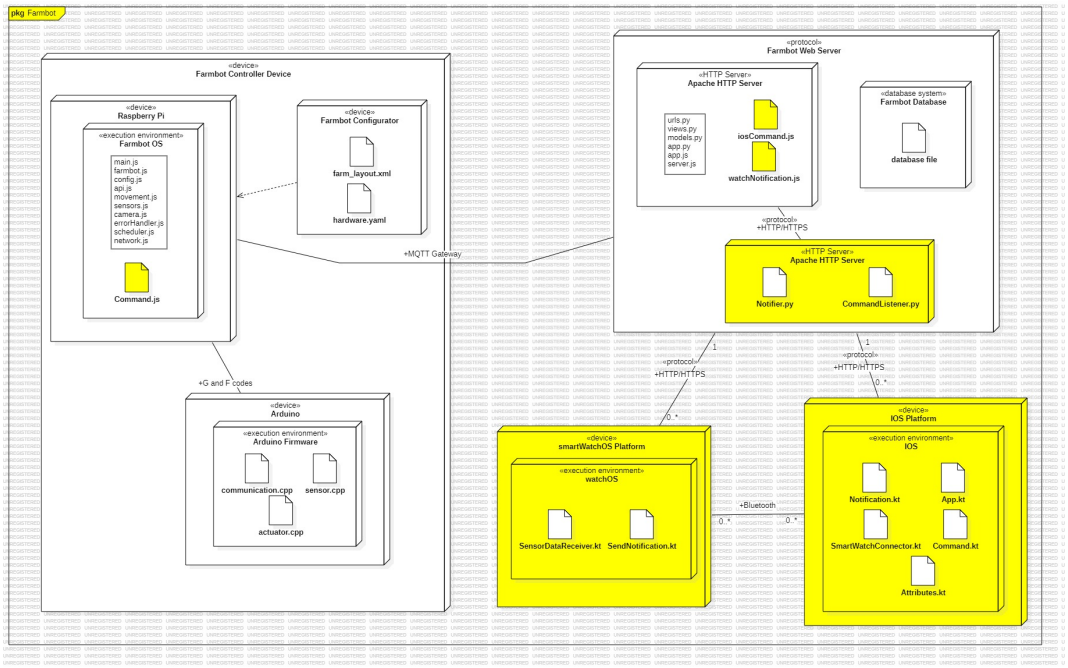

Figure 5.8: Improved Deployment Diagram

LogCommand.js and WatchNotification.js: These are likely JavaScript files or modules that handle logging commands and managing notifications within the FarmBot Web Server component.

HTTP Server - Apache HTTP Server: This is the web server component, which is an instance of the Apache HTTP Server. It serves as the main web server for the FarmBot application.

Notifier.py and CommandListener.py: These are Python scripts or modules that run within the Apache HTTP Server. Notifier.py is likely responsible for handling notifications or alerts, while CommandListener.py may be responsible for listening for and processing incoming commands from clients.

Command.js: This is a JavaScript file or module that resides within the FarmBot Controller Device component. It is likely responsible for handling and processing commands related to the FarmBot controller.

SmartWatchOS Platform: This represents the platform or operating system for smart watches. It includes the following components:

watchOS: The execution environment for the smart watch operating system.

SensorDataReceiver.kt and SendNotification.kt: These are likely Kotlin files or modules that handle receiving sensor data and sending notifications, respectively, on the smart watch platform.

iOS Platform: This represents the iOS platform or operating system for Apple devices. It includes the following components:

iOS: The execution environment for the iOS operating system.

Notification.kt and App.kt: These are likely Kotlin files or modules that handle notifications and the main application logic, respectively, on the iOS platform.

SmartWatchConnector.kt and Command.kt: These are likely Kotlin files or modules that facilitate communication and command handling between the iOS platform and smart watches.

Attributes.kt: This file or module may handle attribute management or configuration for the iOS platform.

The improved parts represent a combination of web server components, platform-specific components (smartWatchOS and iOS), and various modules or files written in different languages (JavaScript, Python, Kotlin) to handle different aspects of the FarmBot system, such as commands, notifications, sensor data handling, and platform-specific functionality.

5.5 Design Rationale

Context View Rationale

The enhanced Context View aims to depict FarmBot's interactions with external entities more comprehensively. We've added Smart Watch and eBay Shop interfaces to boost integration capabilities. The Smart Watch interface enables users to control FarmBot and receive sensor data on their wearables for convenience and real-time monitoring. Meanwhile, the eBay Shop interface facilitates seamless purchasing of FarmBot

components, ensuring easy access to necessary products and improving user experience.

Functional View Rationale

The upgraded Functional View integrates new functionalities to enhance FarmBot's capabilities. The Smart Watch interface allows users to pair their devices, fetch sensor data, and send control commands, supporting flexible and efficient farm management, especially for mobile-oriented users. Additionally, integration with eBay enables direct access to product information and purchases from the FarmBot web app, streamlining procurement and maintenance processes. These enhancements maintain system scalability and adaptability to evolving user needs.

Information View Rationale

The purpose of the Information View was to offer a general description of the database design and data structures that the FarmBot system used. For those involved in data integrity, efficient storage, and retrieval procedures, this perspective is essential. We make sure that the data administration of the system is efficient and well-organized, which supports the overall dependability and effectiveness of the system, by outlining the important database objects and their relationships.

Deployment View Rationale

The Deployment View describes how the FarmBot system is physically installed on different hardware and network nodes. For those working on the system's installation, configuration, and upkeep, this viewpoint is crucial. We make sure that the system is set up in a way that maximizes performance, reliability, and security while satisfying the operational requirements of the stakeholders by explicitly illustrating the deployment architecture of the system, including server locations, network configurations, and protocols for communication.

[2]