Department of Computer Engineering

CENG350 Software Engineering

Software Architecture Description (SAD) for FarmBot

Group 6

By

Berk Ulutaş 2522084

Koray Özgür 2580843

Saturday 18th May, 2024

Contents

List of Figures

List of Tables

Revision History

Revision	**Date**	**Authors**	**Description**
0.1	27.04.2024	Berk Ulutaş, Koray Özgür	Initalization
1.0	29.04.2024	Berk Ulutaş, Koray Özgür	Part-1
2.0	18.05.2024	Berk Ulutaş, Koray Özgür	Final

Table 1: Revision History

1. Introduction

1.1 Purpose and objectives of FarmBot

FarmBot is an open-source, fully customizable robotic farming system. With its remote controllability and ability to automate food-growing tasks, FarmBot revolutionizes agriculture by streamlining processes and making farming more accessible and efficient. By enabling users to automate repetitive tasks, FarmBot empowers individuals and communities to cultivate their own crops with precision and ease.

1.2 Scope

- FarmBot control system, which includes a web-based interface for remotely controlling and managing the robotic farming operations.
- The FarmBot control system will allow users to plan, schedule, and monitor various farming tasks such as planting seeds, watering plants, monitoring plant health, and controlling other aspects of the growing process. It will provide a user-friendly interface accessible via web browsers on different devices.

- The application of the FarmBot control system involves automating and optimizing farming processes, aiming to make agriculture more accessible, efficient, and sustainable. By enabling remote control and automation of tasks, FarmBot reduces labor requirements, minimizes resource usage, and maximizes yields. The system's benefits include increased productivity, reduced manual labor, and improved precision in farming operations.

- The FarmBot control system will align with user needs and preferences, providing customizable features to accommodate various farming setups and requirements. Additionally, it will integrate with sensors and other hardware components as necessary to monitor and control the growing environment effectively.

1.3 Stakeholders and their concerns

The system has various stakeholders whose technical experiences and capabilities are ranging from basic users to experts. The stakeholders of the system are **end users (farmers, hobbyist), educational institutions, software developers, government agencies.**

- **End Users:** End users are the primary beneficiaries and users of the FarmBot system. Farmers utilize FarmBot to automate tasks such as planting, watering, and weeding on their farms. Hobbyists may also use FarmBot for personal gardening projects or educational purposes, exploring its capabilities in a smaller-scale setting. End users require user-friendly interfaces, reliable hardware, and accessible support for troubleshooting and maintenance to effectively integrate FarmBot into their agricultural practices or personal projects.

- **Educational Institutions:** Educational institutions, including universities and high schools, view FarmBot as a valuable tool for teaching subjects such as agriculture, and robotics. By integrating FarmBot into their curriculum, these institutions provide students with hands-on learning experiences and research opportunities. Additionally, FarmBot serves as a platform for exploring concepts such as automation, sustainability, and food production.

- **Government Agencies:** Government agencies such as the Ministry of Agriculture and Forestry are interested in sustainable agriculture and advancing technological solutions to address challenges in the agricultural sector. They recognize the potential of FarmBot to increase productivity, preserve resources, and improve food efficiency. Additionally, organizations like NASA may explore FarmBot technology for applications in space agriculture, supporting long duration space missions.

- **Software Developers:** Developers contribute to the FarmBot project as part of an open source community, collaborating to improve and expand the capabilities of the system. Developers from diverse backgrounds bring expertise in software development, robotics, and agriculture to the FarmBot ecosystem. They work on tasks such as developing control software, designing hardware components, and improving user interfaces.

2. References

This document is written with respect to the IEEE 29148-2018 standards:

29148-2018 - ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes – Requirements engineering

Aronson, R. L. (2013, September 19). *The FarmBot whitepaper*. FarmBot. https://farm.bot/pages/whitepaper

FarmBot. (n.d.). FarmBot. Retrieved May 15, 2024, from https://farm.bot

Rozanski, John, and Woods, Emily. Viewpoint Catalog. O'Reilly, 2024.

3. Glossary

AI Artificial Intelligence.

API Application Programming Interface.

CRUD Create Read Update Delete.

ML Machine Learning.

NLP Natural Language Processing.

4. Architectural Views

4.1 Context View

4.1.1 Stakeholders' uses of this view

There are 4 main stakeholders of FarmBot, which are end users, educational institutions, developers and government. Stakeholders may make use of this viewpoint in order to understand different perspectives and interactions between the website and any kind of stakeholders. The context diagram provides a visual representation of the system's external dependencies and interfaces, helping stakeholders understand the overall ecosystem in which FarmBot operates.

4.1.2 Context Diagram

The system context diagram for FarmBot shows the key components and interactions within the farming ecosystem. At its core is the FarmBot Information System, which interfaces with users to receive commands and provide feedback. The system also integrates with openfarm.cc, leveraging its API to access agricultural data such as planting guides and recommended schedules. Additionally, the Farm Environment interacts with the FarmBot system, transmitting real-time sensor data and logging farming activities. Together, these components enable efficient and optimized management of farming operations.

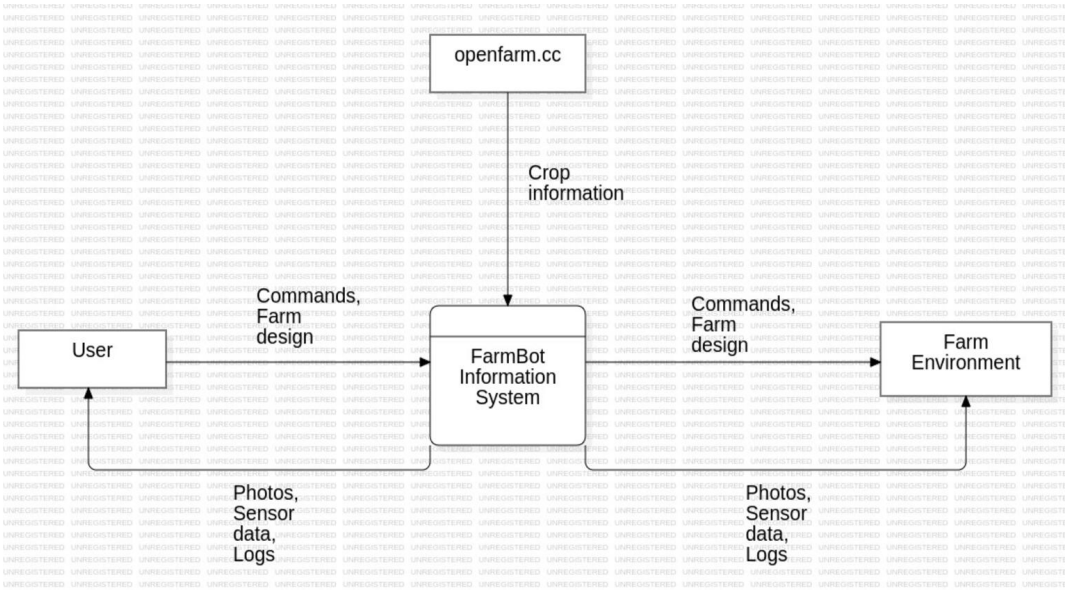

Figure 4.1: Context Diagram

- **User:** They have interactions with FarmBot system to send commands to farm environment and retrieve info about farming environment data and logs.
- **openfarm.cc**: This API enables FarmBot to retrieve data such as crop planting guides, watering plants, fertilizing plants and recommended planting dates.
- **Farm Environment:** It has interactions with FarmBot system to retrieve commands from user and send info about farming environment data and logs to user.

4.1.3 External Interfaces

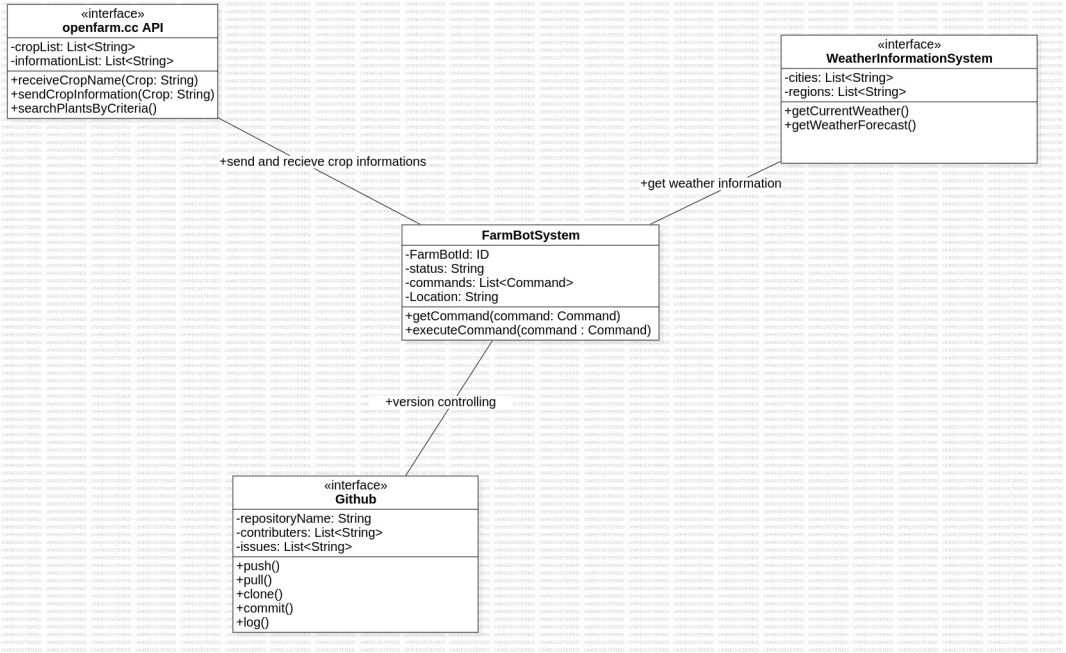

Figure 4.2: External Interfaces Class Diagram

- **FarmBotSystem:** This class represents the core system of FarmBot, responsible for orchestrating interactions with external interfaces. It provides methods for interacting with the external interfaces.

- **GitHub:** This class represents the GitHub interface, which FarmBot uses for version control and collaboration. It offers methods for fetching software updates, uploading data logs, and synchronizing configurations with GitHub repositories.

- **openfarm.cc:** This class represents the OpenFarm.cc interface, which FarmBot leverages for accessing plant-related information. It provides methods for retrieving detailed plant information and searching for plants based on specific criteria.

- **WeatherInformationSystem:** This class represents the Weather Information System interface, which FarmBot interacts with to obtain weather-related data. It offers methods for fetching current weather conditions and weather forecasts relevant to FarmBot operations.

Operation	**Description**
receiveCropName	Receives the name of a crop from an external source.
sendCropInformation	Sends information about a crop to an external source.
searchPlantsByCriteria	Searches for plants based on specified criteria from an external database.
getCommand	Retrieves commands from an external source for the FarmBot system.
executeCommand	Executes commands received from an external source by the FarmBot system.
push	Pushes changes from the local FarmBot system to an external repository.
pull	Pulls changes from an external repository to the local FarmBot system.
clone	Clones a repository from an external source to the local FarmBot system.
commit	Commits changes made in the local FarmBot system to an external repository.
log	Retrieves log information from an external source.
getCurrentWeather	Retrieves the current weather information from an external weather service.
getWeatherForecast	Retrieves the weather forecast from an external weather service.

Table 4.1: External Interface Operation Descriptions

FarmBot utilizes GitHub for managing software updates, logging data, and maintaining configurations. By interacting with GitHub, FarmBot can stay updated with the latest software releases, share data logs for analysis, and synchronize configurations

across different installations. FarmBot relies on OpenFarm.cc to access comprehensive information about plants, including planting instructions, growth requirements, and other agricultural data. This interface enables FarmBot to make informed decisions about planting schedules, nutritional needs, and cultivation techniques based on accurate plant-related information. FarmBot communicates with the Weather Information System to obtain real-time weather data and forecasts essential for efficient farming operations. By accessing weather information, FarmBot can adjust watering schedules, optimize planting times, and mitigate potential risks associated with adverse weather conditions.

4.1.4 Interaction scenarios

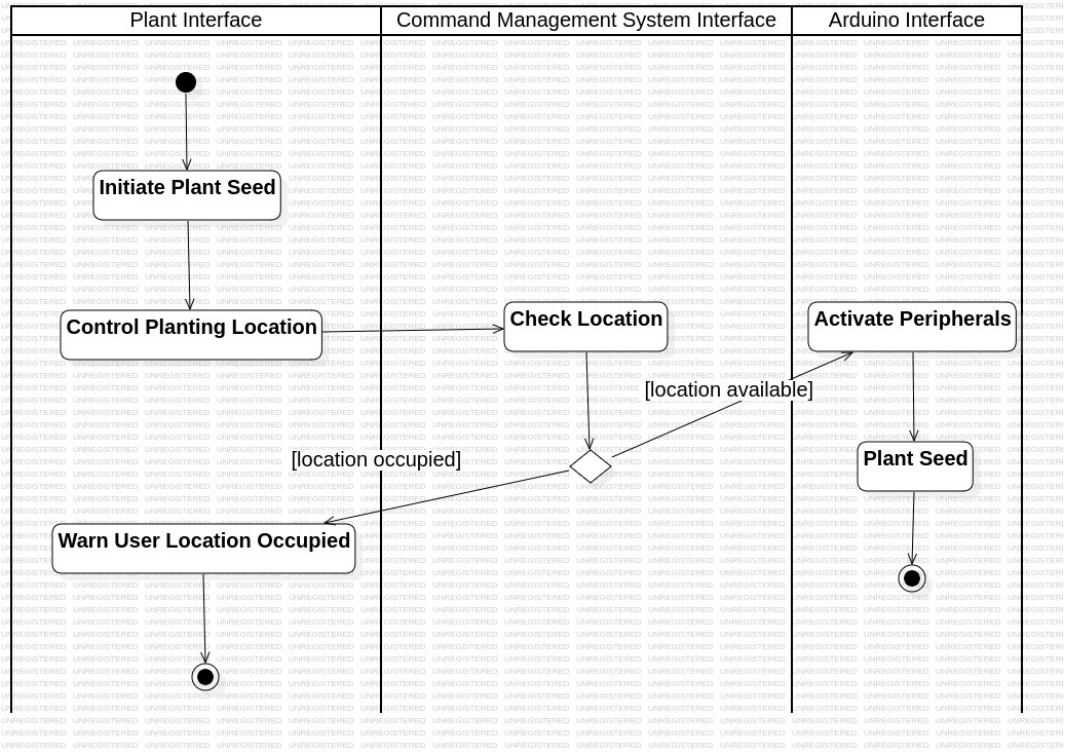

Figure 4.3: Plant Seed Activity Diagram

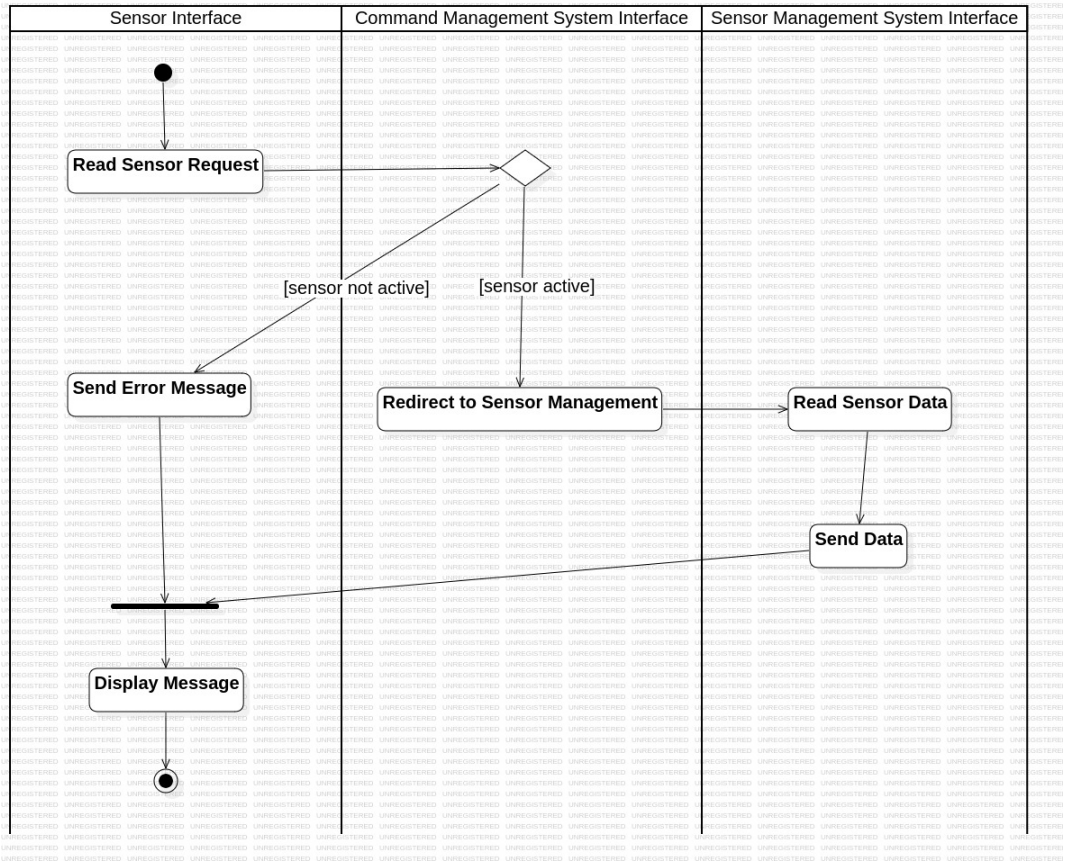

Figure 4.4: Read Sensor Activity Diagram

4.2 Functional View

4.2.1 Stakeholders' uses of this view

There are 4 main stakeholders of FarmBot end users, educational institutions, developers, and government. End users use this view to understand the capabilities and features of the FarmBot system. This helps them determine how they can use FarmBot for farming operations. Educational institutions use this view to evaluate how FarmBot can be integrated into research projects. Developers use the functional view to understand the functional requirements and interactions of different components within the FarmBot system. Government use this view to understand how FarmBot operates and the tasks it can perform, government agencies can assess its potential impact on

agricultural practices.

4.2.2 Component Diagram

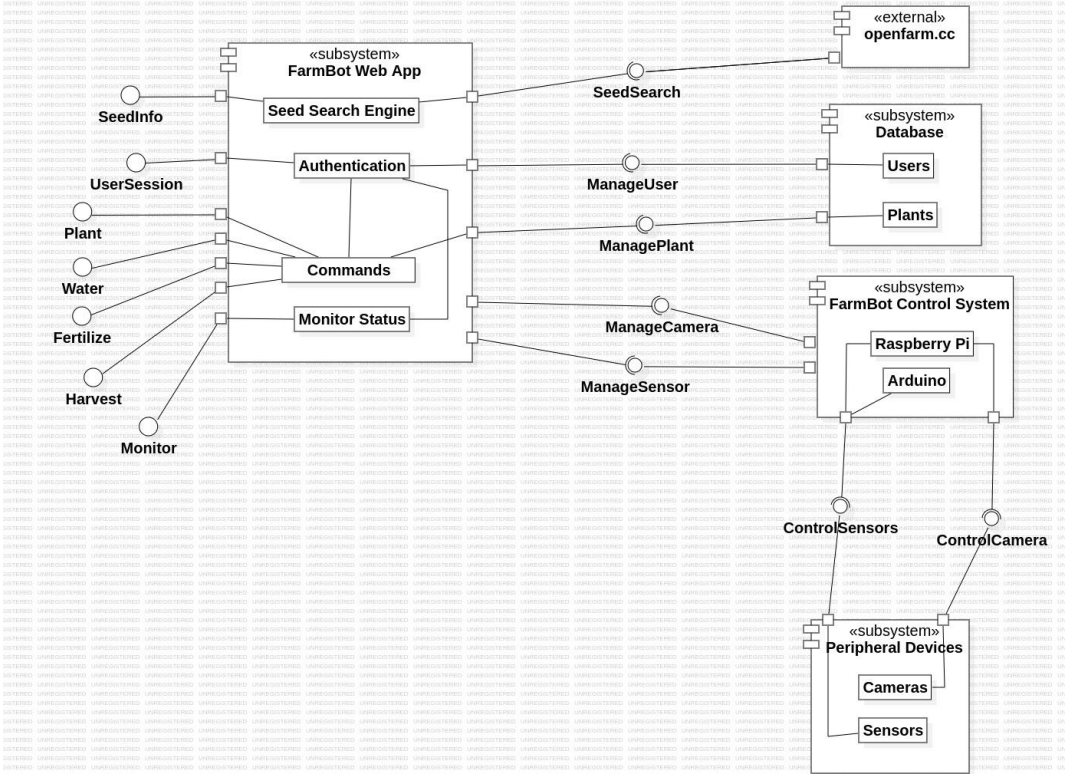

Figure 4.5: Component Diagram

- There are 4 subsystems in FarmBot's infrastructure, namely FarmBot Web App, FarmBot Control System, Database and Peripheral Devices. And one external component openfarm.cc

- Seed search engine part provides an interface to search for information about seeds. This interface allows users to access details such as planting guidelines, growth requirements, and recommended varieties for different crops.

- Authentication part provides a user session interface, allowing users to log in and authenticate themselves to access the system. This interface ensures that only authorized users can interact with the web app and perform actions.

- Commands part provides interfaces for sending commands to the FarmBot hardware. These interfaces include functionalities such as planting, watering, fertilizing, and harvesting crops.

- Monitor status part provides a monitor interface for users to track the status and progress of tasks performed by the FarmBot hardware. This interface displays real-time information about the FarmBot's operations, sensor readings, and any errors or alerts that occur during execution.

- External component openfarm.cc provides a interfaces to fetch detailed information such as planting guidelines, growth requirements, and recommended varieties for different crops.

- In database subsystem there exists user and plant data. The users part of the database subsystem stores information about the users of the FarmBot system. This includes data such as user profiles, authentication credentials, preferences, and permissions. The plants part of the database subsystem stores information about the plants managed by the FarmBot system. This includes data such as plant species, varieties, planting schedules, growth stages, and sensor readings.

- In FarmBot control system subsystem there exists raspberry pi and arduino parts. Raspberry Pi part of the FarmBot control system serves as a central processing unit responsible for coordinating the operation of the FarmBot hardware. It runs software applications and algorithms that control the movement of motors, manage sensor inputs, process data, and communicate with external devices. Arduino part of the FarmBot control system serves as a microcontroller unit responsible for interfacing with sensors and actuators connected to the FarmBot hardware. It executes firmware code that reads sensor data, controls motor movements, and activates actuators.

- In Peripheral devices subsystem there exists cameras and sensors part. Cameras part of the peripheral devices subsystem consists of hardware devices used for

capturing images and videos of the plants and surrounding environment. Sensors part of the peripheral devices subsystem consists of various types of sensors used for collecting data about the environment and plant conditions.

4.2.3 Internal Interfaces

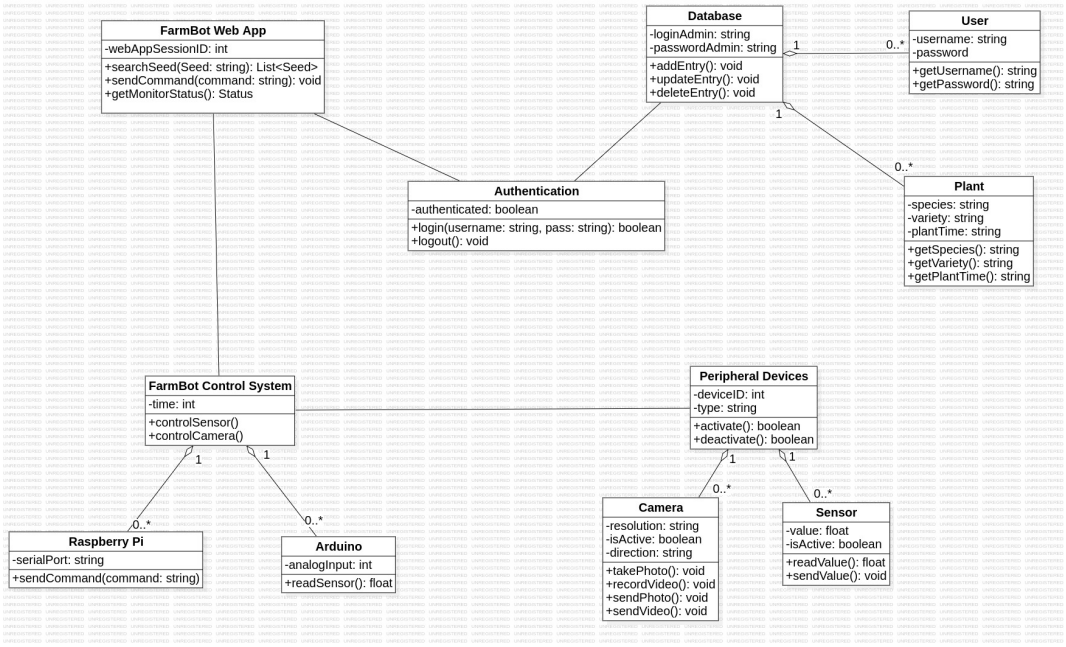

Figure 4.6: Internal Interfaces Class Diagram

- **FarmBot Web App:** This class represents the FarmBot web application and includes methods for searching seeds, user authentication, sending commands, and retrieving monitor status.
- **Database:** The Database class includes classes for Users and Plants, with attributes and methods for managing user accounts and plant information.
- **FarmBot Control System:** The RaspberryPi and Arduino classes represent the hardware components of the Farmbot control system. The RaspberryPi class includes a serialPort attribute and a method for sending commands. The Arduino class includes an analogInput attribute and a method for reading sensors.

- **Peripheral Devices:** The Camera and Sensor classes represent the peripheral devices used by the FarmBot system. The Camera class includes attributes for resolution and methods for taking photos and recording videos and sending them. The Sensor class includes an attribute for sensor value and methods for getting and sending sensor values.

Operation	**Description**
searchSeed	Searches for seeds in the database.
sendCommand	Sends a command to the FarmBot control system.
getMonitorStatus	Retrieves the current status of the FarmBot system.
controlSensor	Controls a sensor connected to the FarmBot system.
controlCamera	Controls a camera connected to the FarmBot system.
readSensor	Reads data from a sensor connected to the FarmBot system.
login	Logs a user into the FarmBot web application.
logout	Logs a user out of the FarmBot web application.
addEntry	Adds an entry to the database.
updateEntry	Updates an entry in the database.
deleteEntry	Deletes an entry from the database.
activate	Activates a component or feature within the FarmBot system.
deactivate	Deactivates a component or feature within the FarmBot system.

Table 4.2: Internal Interface Operation Descriptions

Operation	**Description**
takePhoto	Takes a photo using the FarmBot camera.
recordVideo	Records a video using the FarmBot camera.
sendPhoto	Sends a photo captured by the FarmBot camera.
sendVideo	Sends a video recorded by the FarmBot camera.
getUsername	Retrieves the username of the current user.
getPassword	Retrieves the password of the current user.
getSpecies	Retrieves the species of a plant.
getVariety	Retrieves the variety of a plant.
getPlantTime	Retrieves the time when a plant was planted.

Table 4.3: Internal Interface Operation Descriptions - continued

4.2.4 Interaction Patterns

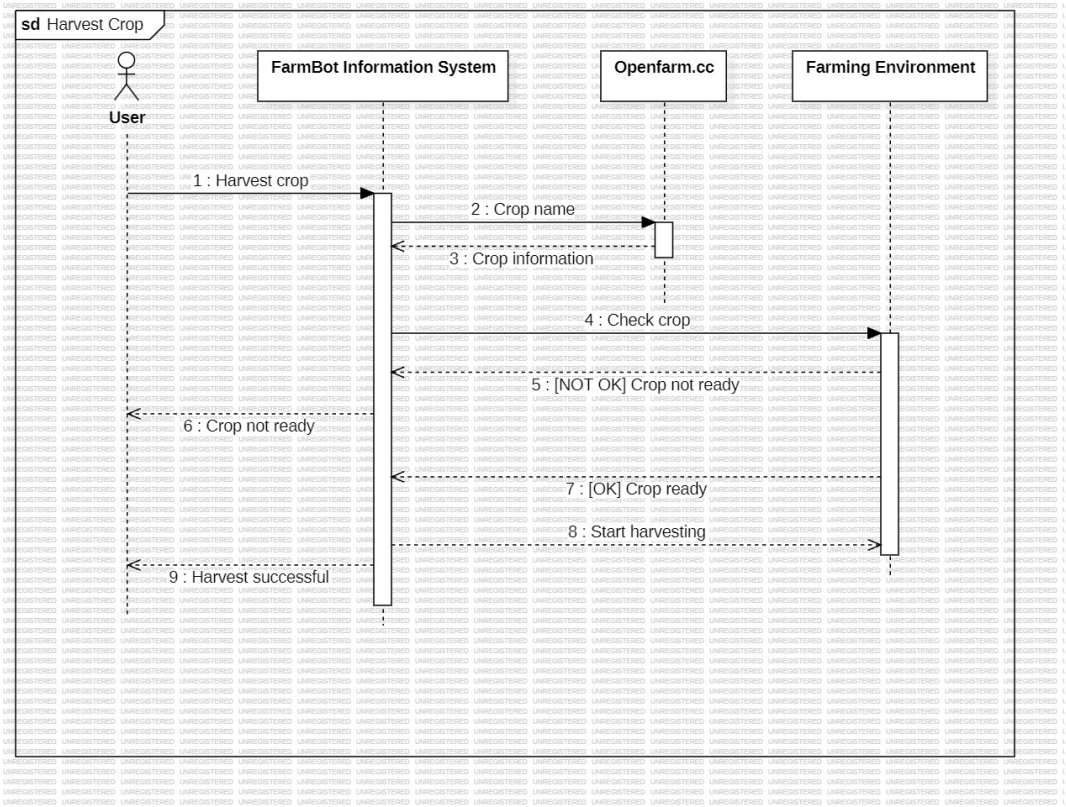

Figure 4.7: Harvest Crop Sequence Diagram

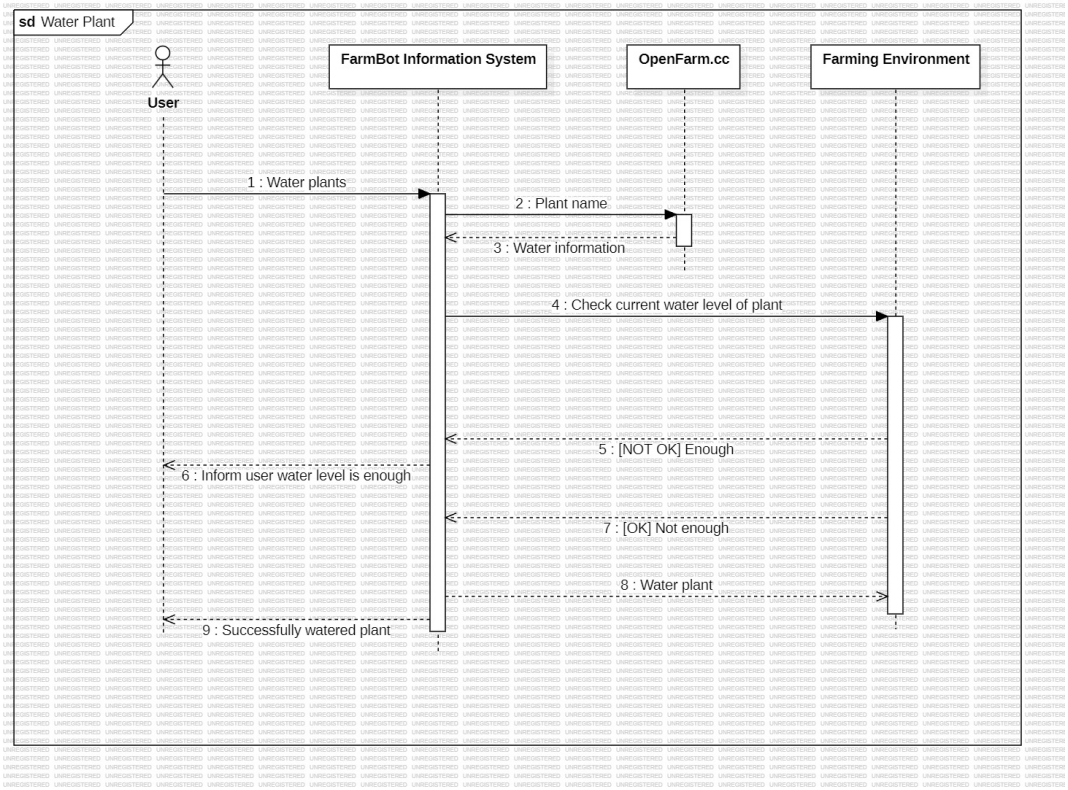

Figure 4.8: Water Plant Sequence Diagram

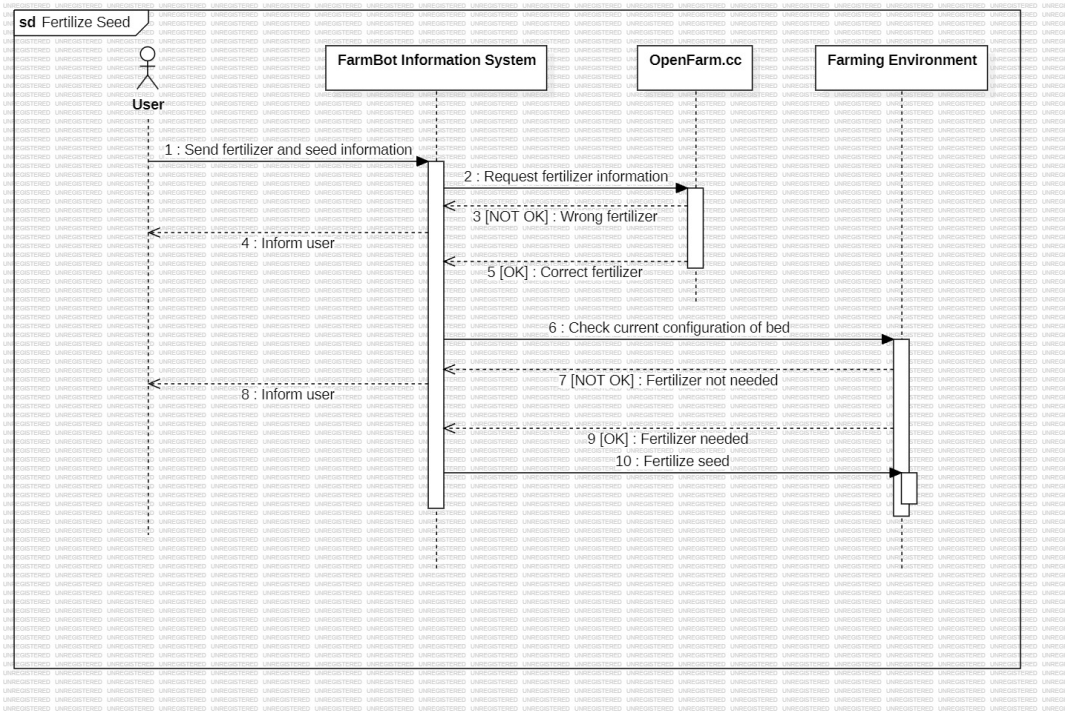

Figure 4.9: Fertilize Seed Sequence Diagram

4.3 Information View

4.3.1 Stakeholders' uses of this view

There are 4 main stakeholders of FarmBot end users, educational institutions, developers, and government. In the information view of FarmBot, stakeholders leverage the system's data management aspects. End users gain insights into how their data is handled, and efficient data management practices. Educational institutions use the system's handling of educational data to ensure alignment with curricular needs and data privacy regulations. Developers utilize this view to understand data structures and flows, make easier and effective software component design and implementation. Government agencies evaluate data management practices to address regulatory concerns. Understanding FarmBot's potential impact on agricultural practices through its information architecture.

4.3.2 Database Class Diagram

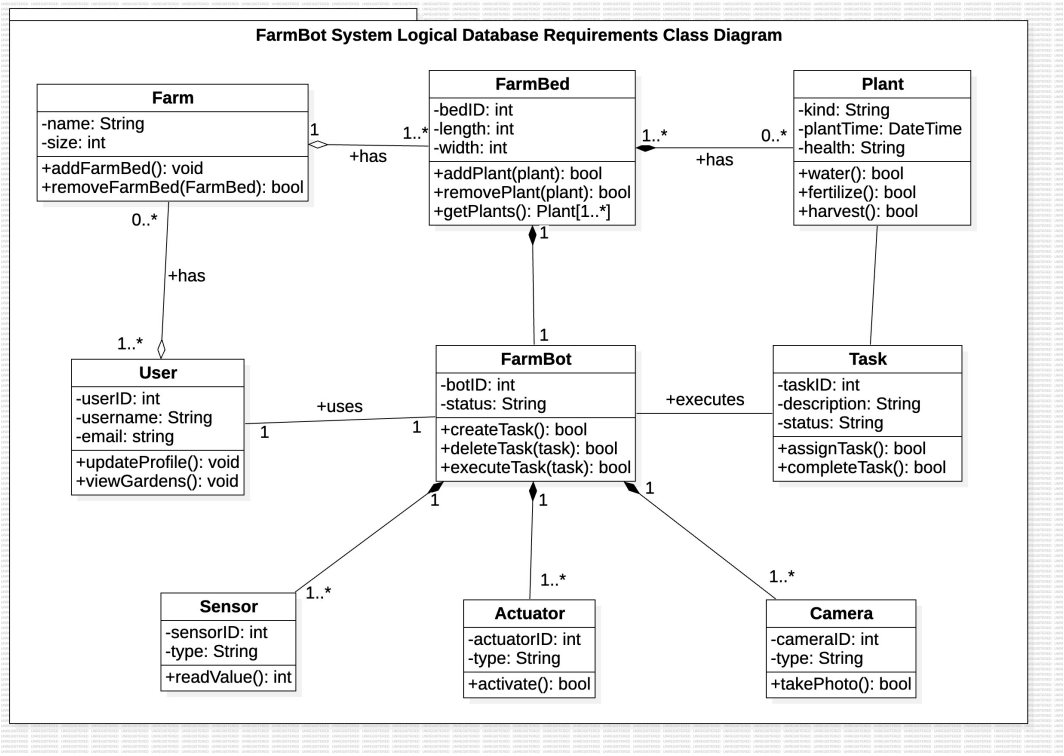

Figure 4.10: Database Class Diagram

4.3.3 Operations on Data

Operation	**CRUD (Create/Read/Update/Delete) Description**
addFarmBed	Create: Add a new farm bed to the system Read: Update: Delete:
removeFarmBed	Create: Read: Update: Delete: Remove specified farm bed from the system
addPlant	Create: Add a new plant to the system Read: Update: Delete:
removePlant	Create: Read: Update: Delete: Remove specified plant from the system
getPlants	Create: Read: Retrieve information about all plants Update: Delete:

Table 4.4: CRUD Operations

Operation	**CRUD (Create/Read/Update/Delete) Description**
updateProfile	Create: Read: Update: Update user profile information Delete:
viewGardens	Create: Read: View information about all gardens Update: Delete:
water	Create: Initiate watering for specified plant or farm bed Read: Update: Delete:
fertilize	Create: Apply fertilizer to specified plant or farm bed Read: Update: Delete:
harvest	Create: Harvest crops from specified plant or farm bed Read: Update: Delete:

Table 4.5: CRUD Operations - continued

Operation	**CRUD (Create/Read/Update/Delete) Description**
createTask	Create: Create a new task Read: Update: Delete:
deleteTask	Create: Read: Update: Delete: Delete specified task
executeTask	Create: Read: Update: Mark specified task as executed Delete:
assignTask	Create: Read: Update: Assign specified task to a user Delete:
completeTask	Create: Read: Update: Mark specified task as completed Delete:
readValue	Create: Read: Retrieve sensor reading Update: Delete:

Table 4.6: CRUD Operations - continued

Operation	CRUD (Create/Read/Update/Delete) Description
readValue	Create: Read: Retrieve recorded sensor reading Update: Delete:
activate	Create: Read: Update: Activate specified sensor Delete:
takePhoto	Create: Capture photo using system's cameras Read: Update: Delete:

Table 4.7: CRUD Operations - continued

4.4 Deployment View

4.4.1 Stakeholders' uses of this view

There are 4 main stakeholders of FarmBot, which are end users, educational institutions, developers and government. End users and educational institutions use this view to understand how they can use FarmBot system and hardware components. Developers use this view to understand interactions of physical and software components of FarmBot. The government uses this view to understand how the system is deployed in whole perspective.

4.4.2 Deployment Diagram

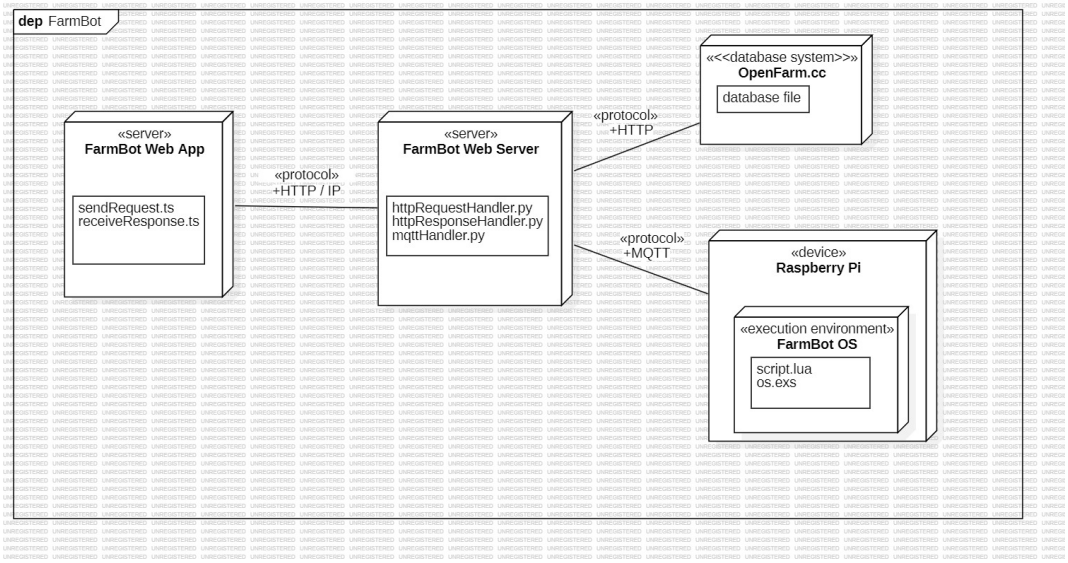

Figure 4.11: Deployment Diagram

- FarmBot Client Web App and FarmBot Web Server communicate with the protocols HTTP and IP.
- FarmBot Web Server and OpenFarm.cc communicate with the protocol HTTP.
- FarmBot Web Server and Raspberry Pi communicate with the protocol MQTT.
- sendRequest.ts and receiveResponse.ts are responsible for requests and responses at the FarmBot Web App.
- httpRequestHandler.py provides the implementation of the HTTP interface with the query parameters, path variables, request headers, and request body of the HTTP Request.
- httpResponseHandler.py provides the implementation of the HTTP interface for handling the responses.

- mqttHandler.py is responsible for communication with the protocol MQTT at the FarmBot Web Server side.
- os.exs provides the operating system's functionality in the Raspberry Pi device.
- script.lua provides the ability to run sequences for automation purposes at the Raspberry Pi.

4.5 Design Rationale

4.5.1 Context View

The rationale behind the context view is to provide a overall understanding of the FarmBot system, including its external interfaces, interactions, and overall impact. This view focuses on analyzing requirements, interactions with stakeholders such as farmers and agricultural experts, and identifying external dependencies. It aims to clearly define the system scope and discuss the impact of FarmBot on modern farming practices, emphasizing automation, efficiency, and sustainability in agriculture.

4.5.2 Functional View

The rationale behind the functional view is to depict the key functional components of the FarmBot system and how they interact. This view highlights the high-level functionality of system for that stakeholders, including farmers and system administrators, need to understand how FarmBot performs tasks such as planting, watering, and monitoring crops. It addresses concerns of scalability, modularization, and validation and verification of system requirements, ensuring the system can grow and adapt to different farming scenarios and needs.

4.5.3 Information View

The rationale behind the information view is to illustrate how data is processed within the FarmBot system. This view focuses on the structure and flow of data, in-

cluding interactions between different data elements, the transformation processes, and data persistence. It also touches on consistency, interoperability, integration, security, and privacy concerns of the system.

4.5.4 Deployment View

The rationale behind the deployment view is to provide a detailed overview of how FarmBot will be deployed in its target environment. This view focuses on the physical infrastructure required for deployment, explaining how the different modules, submodules, and components are distributed to make up the whole system. It addresses factors such as scalability, performance, availability, fault tolerance, communication between components, and resource management. This view ensures that FarmBot can be deployed reliably and maintain high performance and availability in real-world agricultural settings.

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

5.1 Context View

5.1.1 Stakeholders' uses of this view

This section presents the system's context view after considering the relevant suggestions. It provides a comprehensive overview of the actors and other related systems in a detailed manner. The primary focus is to facilitate the understanding of how the FarmBot functions and how external entities are utilized from a contextual perspective. By showing the interactions between FarmBot and external entities such as OpenAI API and Gmail API, stakeholders gain insights into the system's integration points and dependencies. This holistic view enables stakeholders to appreciate the ecosystem in which FarmBot operates

5.1.2 Context Diagram

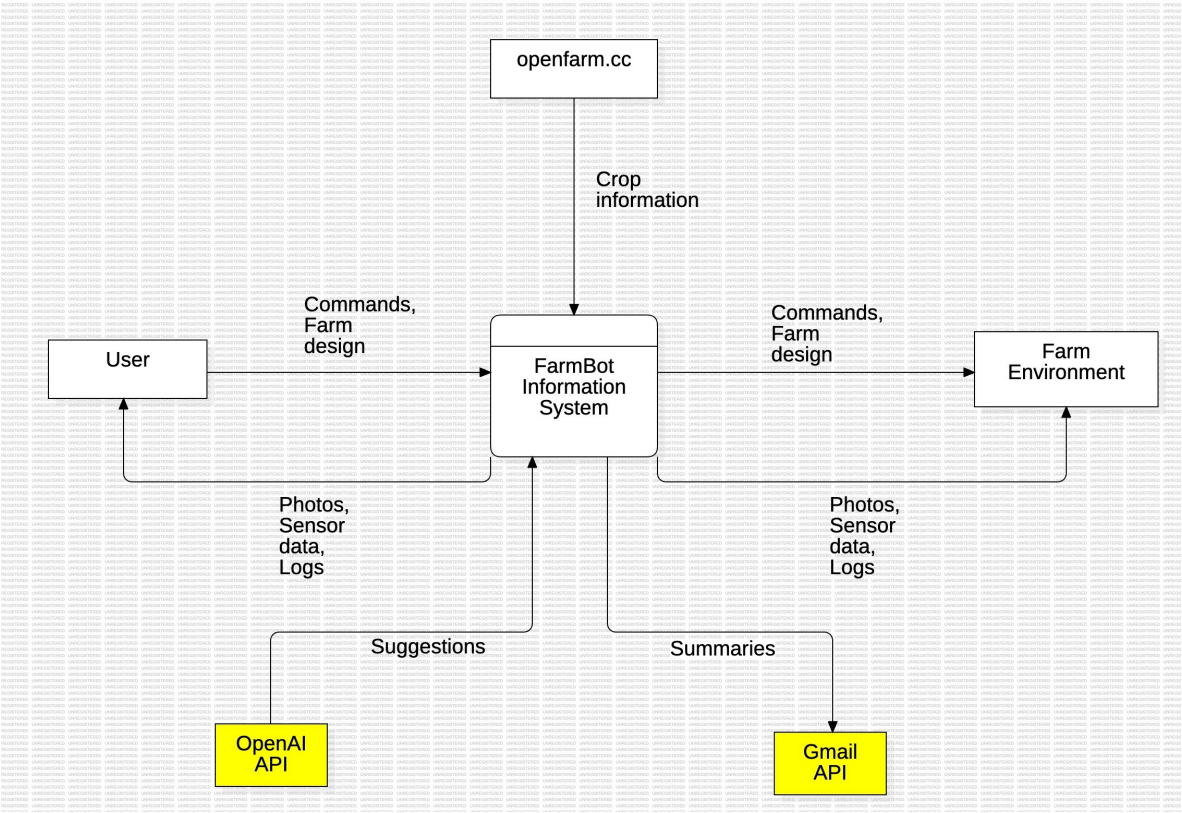

Figure 5.1: Context Diagram (Suggested)

- **OpenAI API:** FarmBot's integration with the OpenAI API enhances its capabilities by leveraging AI and natural language processing NLP technologies. With access to the OpenAI API, FarmBot can analyze and process textual data, enabling advanced features such as intelligent decision-making, language understanding, and predictive analytics.

- **Gmail API:** By integrating the Gmail API, FarmBot gains the ability to communicate and collaborate seamlessly with users via email. This integration allows FarmBot to send automated notifications, alerts, and status updates directly to users' email accounts, providing real-time information and facilitating remote monitoring and management of farming operations.

5.1.3 External Interfaces

This section should include an **External Interfaces Class Diagram for your suggestion**. Descriptions of the operations given in the external interface class diagram should also be given. **You should aim for 2 external interfaces.**

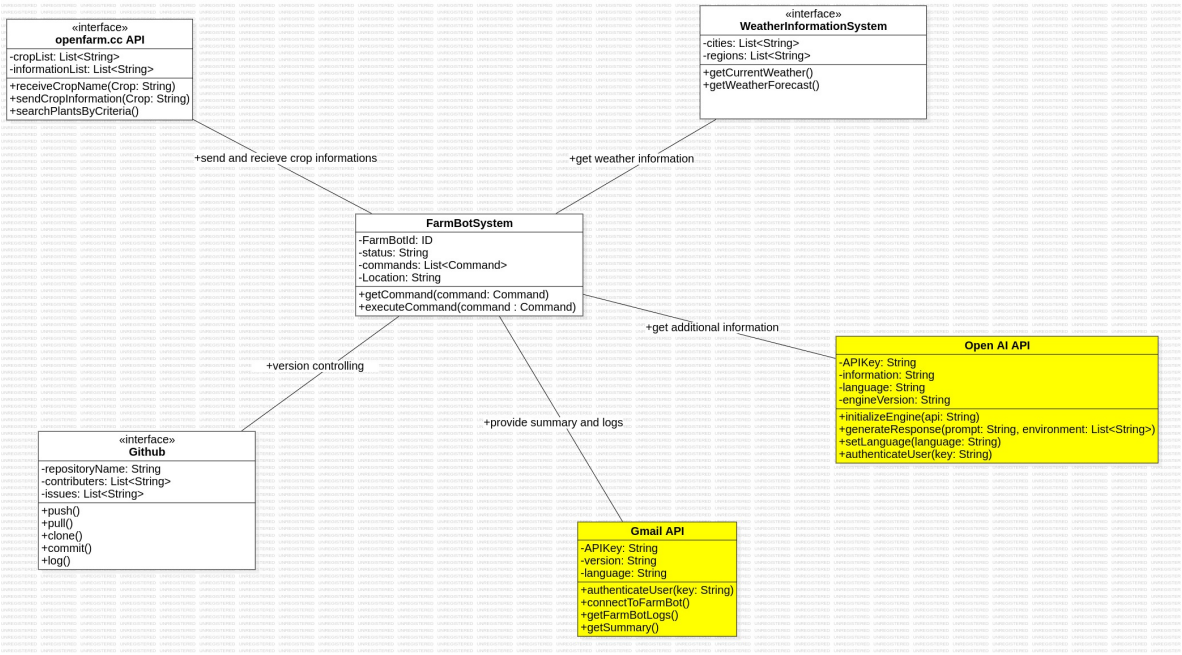

Figure 5.2: External Interfaces Class Diagram (Suggested)

- **OpenAI API:** Through this integration, FarmBot gains access to OpenAI's AI models. Users can interact with FarmBot using natural language queries, commands, and requests. This integration give users to personalized recommendations, obtain insights, and receive assistance on various farming tasks, such as crop selection, planting techniques, pest management, and soil health optimization.

- **Gmail API:** By using the Gmail API, FarmBot automates the process of sending weekly and monthly farm summary updates directly to user. These updates contain reports on farm activities, upcoming tasks, and personalized recommendations, empowering users to stay informed and manage their farms effectively from anywhere.

Operation	**Description**
authenticateUser	Authenticate the user's identity with API key.
connectToFarmBot	Establish a connection with the FarmBot system.
getFarmBotLogs	Retrieve logs and data generated by the FarmBot.
getSummary	Obtain a summary or overview of the current status.
initalizeEngine	Initialize the processing engine or AI system for executing commands.
generateResponse	Generate a response or output based on user inputs, system data.
setLanguage	Set the language or linguistic preferences for interactions with the FarmBot system.

Table 5.1: External Interface Operation Descriptions (Suggested)

5.1.4 Interaction scenarios

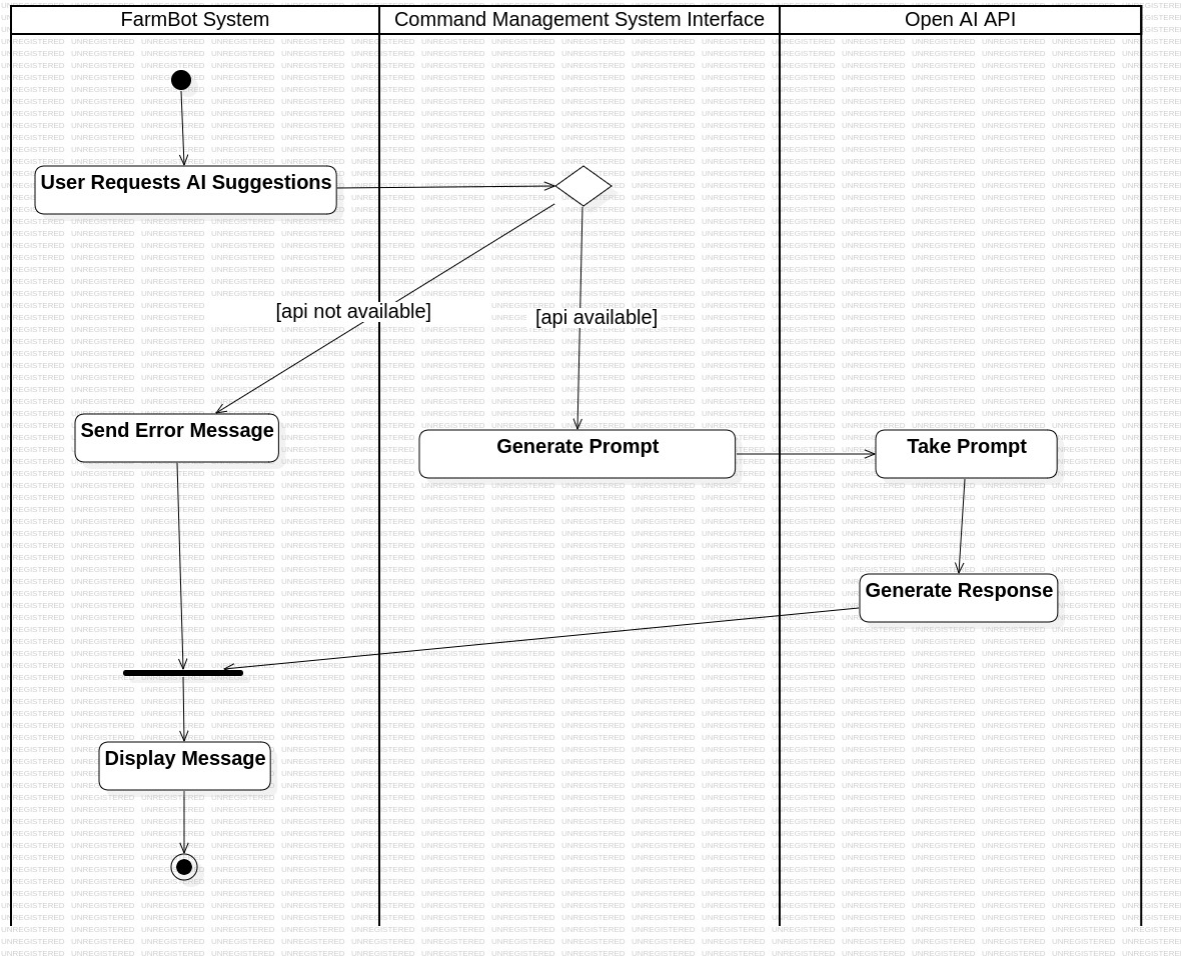

Figure 5.3: Take AI Suggestion Activity Diagram (Suggested)

5.2 Functional View

5.2.1 Stakeholders' uses of this view

There are 4 main stakeholders of FarmBot end users, educational institutions, developers, and government. End users use this view to understand the capabilities and features of the FarmBot system. This helps them determine how they can use FarmBot for farming operations. Educational institutions use this view to evaluate how FarmBot

can be integrated into research projects. Developers use the functional view to understand the functional requirements and interactions of different components within the FarmBot system. Government use this view to understand how FarmBot operates and the tasks it can perform, government agencies can assess its potential impact on agricultural practices.

5.2.2 Component Diagram

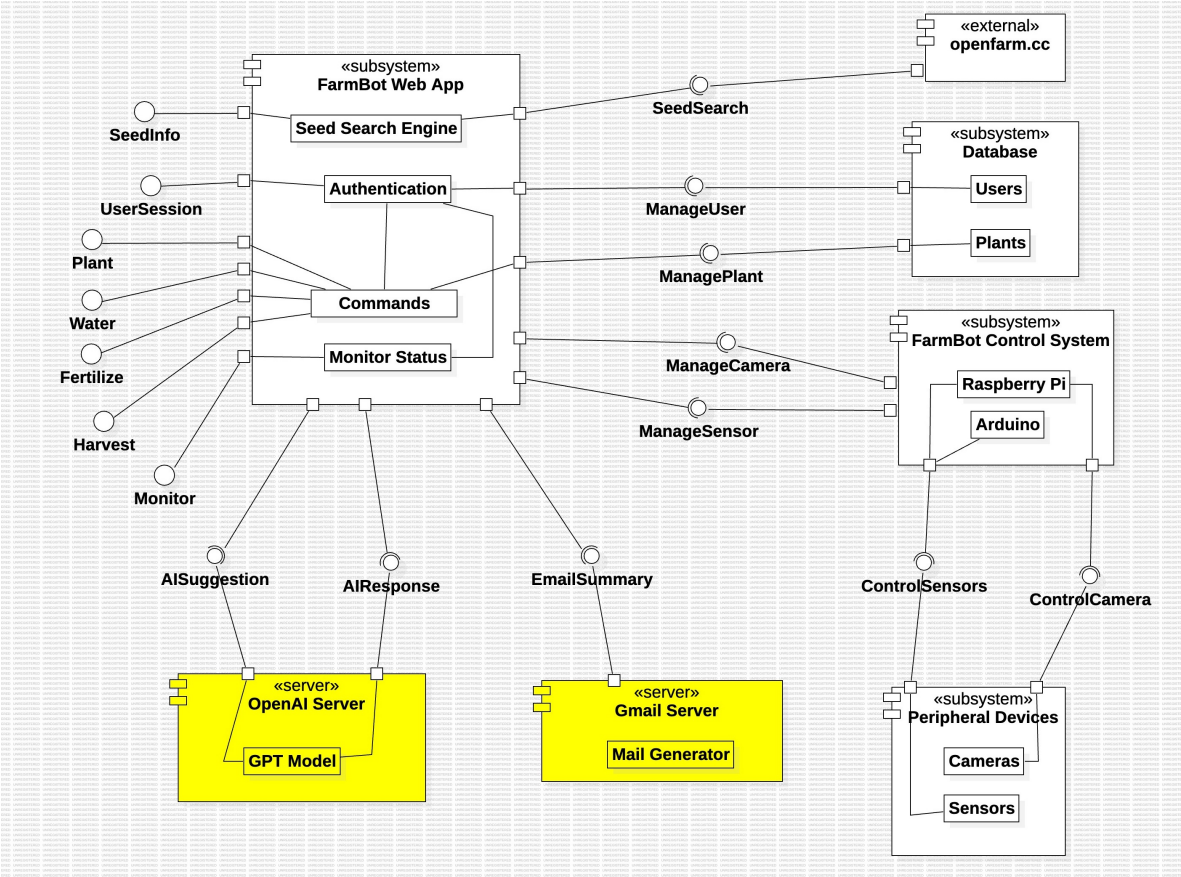

Figure 5.4: Component Diagram (Suggested)

- FarmBot ecosystem has two more new components, which are OpenAI Server and Gmail Server.
- The OpenAI Server component represents the server infrastructure used to interact with the OpenAI API for AI-related functionalities within the FarmBot

system.

- This component facilitates the integration of AI capabilities, such as NLP, ML, and predictive analytics, into FarmBot's operations.
- Key functionalities supported by the OpenAI Server include generating AI-driven recommendations, analyzing farming data, optimizing resource utilization, and improving overall farming efficiency.
- The Gmail Server component represents the server infrastructure utilized for integrating email communication capabilities into the FarmBot system via the Gmail API.
- Key functionalities supported by the Gmail Server include sending email notifications for task completion, weather alerts, system errors, and farm performance reports, enhancing communication and collaboration among farm operators and stakeholders.

5.2.3 Internal Interfaces

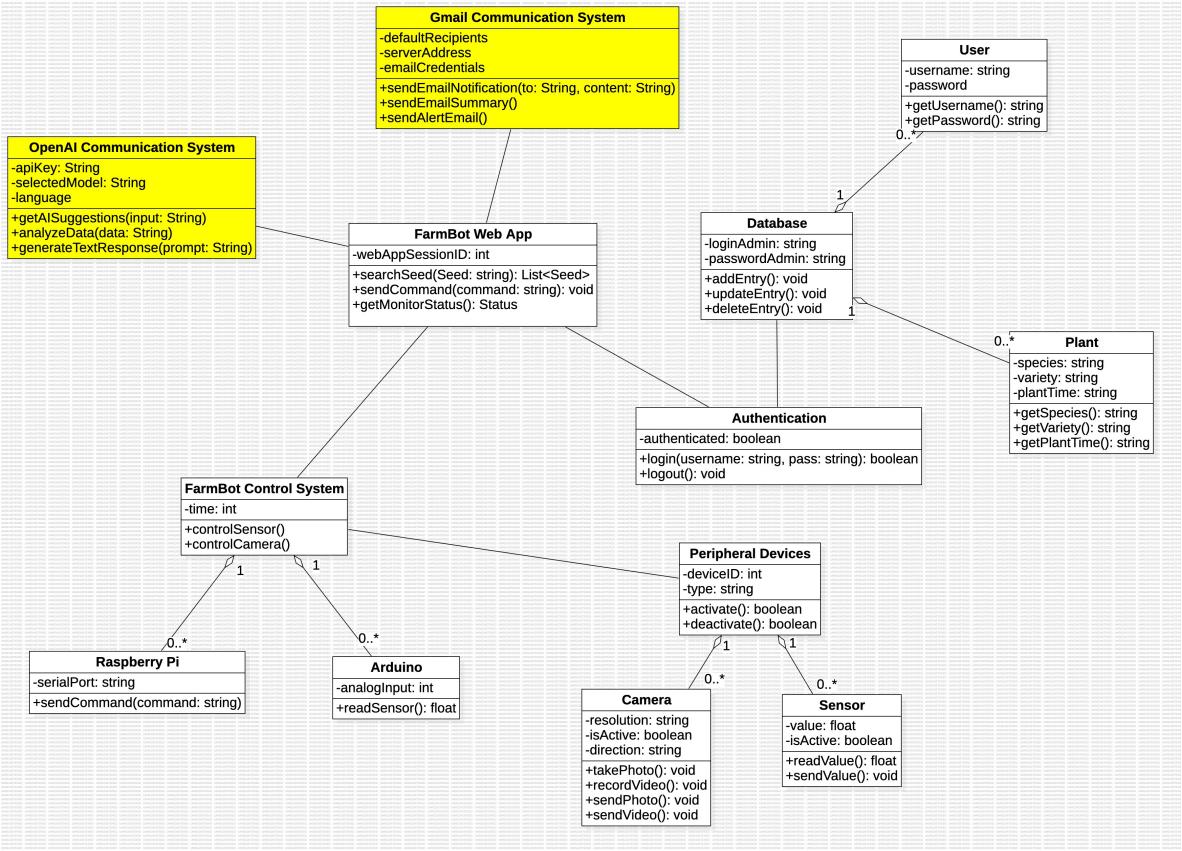

Figure 5.5: Internal Interfaces Class Diagram (Suggested)

- **Gmail Communication System:** This interface facilitates communication with the Gmail server for email-related functionalities within the FarmBot system. It includes methods for sending email notifications, summaries, and alerts, as well as configuring email server settings.

- **OpenAI Communication System:** This interface enables integration with the OpenAI server for AI-related tasks within the FarmBot system. It provides methods for retrieving AI-generated suggestions, analyzing data, generating text responses, and configuring AI server settings.

Operation	**Description**
getAISuggestions	Retrieves AI-generated suggestions based on user input.
analyzeData	Analyzes farming data using AI algorithms to provide insights and predictions.
generateTextResponse	Generates text responses or summaries based on the given prompt using AI models.
sendEmailNotification	Sends email notifications to specified recipients with the given subject and content.
sendEmailSummary	Sends email summaries containing farm activity summaries or reports to specified recipients.
sendAlertEmail	Sends alert emails for system errors, warnings, or critical events to specified recipients.

Table 5.2: Internal Interface Operation Descriptions (Suggested)

5.2.4 Interaction Patterns

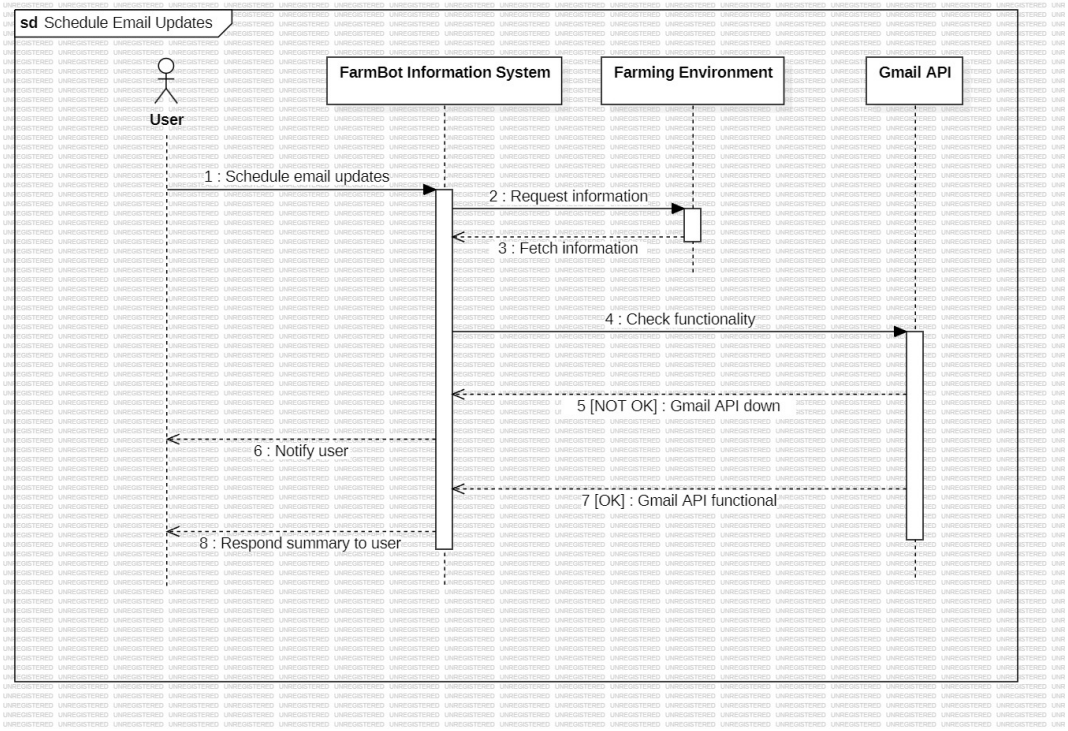

Figure 5.6: Schedule Email Updates Sequence Diagram (Suggested)

5.3 Information View

5.3.1 Stakeholders' uses of this view

There are 4 main stakeholders of FarmBot end users, educational institutions, developers, and government. In the information view of FarmBot, stakeholders leverage the system's data management aspects. End users gain insights into how their data is handled, and efficient data management practices. Educational institutions use the system's handling of educational data to ensure alignment with curricular needs and data privacy regulations. Developers utilize this view to understand data structures and flows, make easier and effective software component design and implementation. Government agencies evaluate data management practices to address regulatory con-

cerns. Understanding FarmBot's potential impact on agricultural practices through its information architecture.

5.3.2 Database Class Diagram

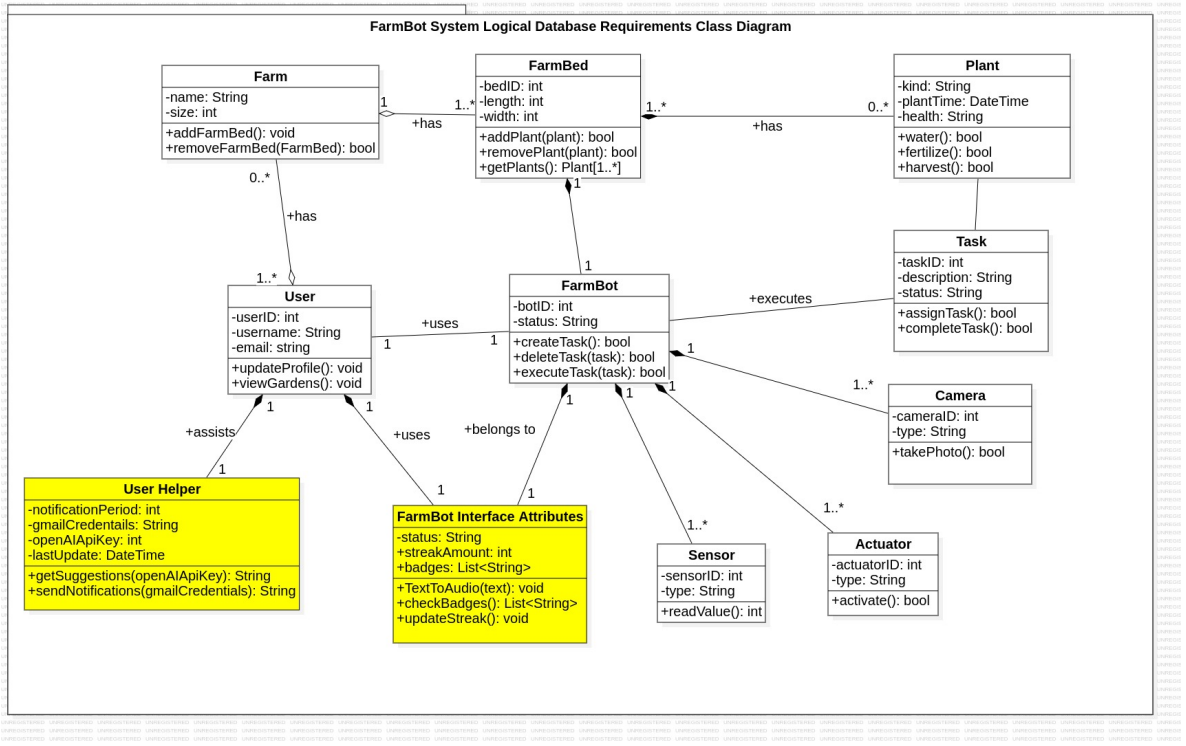

Figure 5.7: Database Class Diagram (Suggested)

5.3.3 Operations on Data

Operation	CRUD (Create/Read/Update/Delete) Description
getSuggestions	Create: Read: Retrieve AI-generated suggestions based on user input Update: Delete:
sendNotifications	Create: Create notifications to users regarding system events Read: Update: Delete:
textToAudio	Create: Convert text to audio for auditory feedback or alerts Read: Update: Delete:
checkBadges	Create: Read: Check and manage user achievements or badges Update: Delete:
updateStreak	Create: Read: Update: Update user's streak or consecutive activity count Delete:

Table 5.3: CRUD Operations (Suggested)

5.4 Deployment View

5.4.1 Stakeholders' uses of this view

There are 4 main stakeholders of FarmBot, which are end users, educational institutions, developers and government. End users and educational institutions use this view to understand how they can use FarmBot system and hardware components. Developers use this view to understand interactions of physical and software components of FarmBot. The government uses this view to understand how the system is deployed in whole perspective.

5.4.2 Deployment Diagram

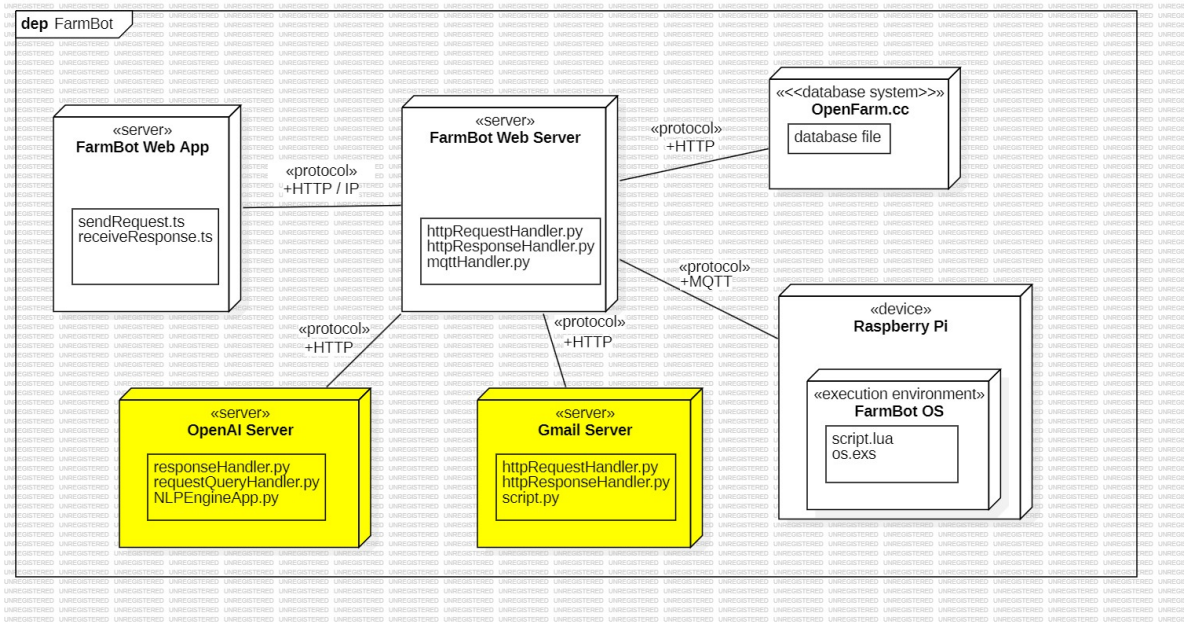

Figure 5.8: Deployement Diagram (Suggested)

- OpenAI Server and Gmail Server communicate with the FarmBot Web Server using HTTP protocol.
- Open AI Server has Response Handler and Request Query Handler files which

handle the user queries in the form of HTTP and returns a response with the answer to the user's query in the form of HTTP.

- Gmail Server has Response Handler and Request Handler files which handle the user queries in the form of HTTP and returns the translation of the user's input in the form of HTTP and the Script.py that is used by the Gmail server for custom email processing tasks, such as auto-responses, email filtering, or email forwarding rules..

5.5 Design Rationale

5.5.1 Context View

The rationale behind the context view is to provide a overall understanding of the FarmBot system, including its external interfaces, interactions, and overall impact. This view focuses on analyzing requirements, interactions with stakeholders such as farmers and agricultural experts, and identifying external dependencies. It aims to clearly define the system scope and discuss the impact of FarmBot on modern farming practices, emphasizing automation, efficiency, and sustainability in agriculture.

5.5.2 Functional View

The rationale behind the functional view is to depict the key functional components of the FarmBot system and how they interact. This view highlights the high-level functionality of system for that stakeholders, including farmers and system administrators, need to understand how FarmBot performs tasks such as planting, watering, and monitoring crops. It addresses concerns of scalability, modularization, and validation and verification of system requirements, ensuring the system can grow and adapt to different farming scenarios and needs.

5.5.3 Information View

The rationale behind the information view is to illustrate how data is processed within the FarmBot system. This view focuses on the structure and flow of data, including interactions between different data elements, the transformation processes, and data persistence. It also touches on consistency, interoperability, integration, security, and privacy concerns of the system.

5.5.4 Deployment View

The rationale behind the deployment view is to provide a detailed overview of how FarmBot will be deployed in its target environment. This view focuses on the physical infrastructure required for deployment, explaining how the different modules, submodules, and components are distributed to make up the whole system. It addresses factors such as scalability, performance, availability, fault tolerance, communication between components, and resource management. This view ensures that FarmBot can be deployed reliably and maintain high performance and availability in real-world agricultural settings.