ODTÜ
METU

Department of Computer Engineering

CENG350 Software Engineering

Software Requirements Specification for FarmBot

Group 55

By

Barış Can

Başar Yılmaz

Saturday 18th May, 2024

Contents

List of Figures

List of Tables

Revision History

Version	Description	Date	Author
v1.0	Finished until 1.3	14.03.2024	Barış, Başar
v1.1	Finished 3.2, added use case diagram and tabular descriptions	23.03.2024	Barış, Başar
v1.2	Finished 3.4, 3.5, 3.6	27.03.2024	Barış, Başar
v1.3	Section 4 finished	06.04.2024	Barış, Başar

Table 1: Revision Table

1. Introduction

This document is written to describe the Software Requirements Specification (SRS) [1] for the open-source project FarmBot. FarmBot is an open-source CNC (computer numerical control) farming project. It consists of a cartesian coordinate robot farming machine, software, and documentation, which includes a farming data repository.

1.1 Purpose of the System

The project aims to create open-source and accessible technology for people to grow food easily. In addition to that, the mission of the benefit corporation FarmBot Inc. is to build a community that develops open-source hardware plans, software, data, and documentation that is available to everyone for building their farming machine. The system, therefore, paves the way for decentralization and democratization of food production.

1.2 Scope

FarmBot is an open-source CNC farming project. The system aims to address the issues that have become prevalent with the ever-growing world population and its food demands. It significantly improves upon the conventional agriculture methods. The scope of the system consists of the following:

- Building a free and open database for farming and gardening knowledge

- A custom operating system, FarmBot OS, for maintaining connection and synchronization between the hardware and the web application using message brokers. Allowing scheduling of events, real-time control and uploading various sensor data and logs
- A web app for easy configuration and control of the FarmBot, featuring real-time manual control capabilities, logging, drag-and-drop farm designing and a routine builder for FarmBot to execute scheduled routines
- Scalability for all sizes of operation from home-use to industrial-use.
- Big data acquisition and analysis for data-driven agriculture.
- Providing the user with various farm design options with space efficiency in mind
- Fully automated and optimized farming operations such as planting, watering, spraying, weed detection, and seed spacing with mono-crop and poly-crop capabilities
- Offering customizability to the user in adding new sensors, adjusting parameters of their FarmBot to their liking

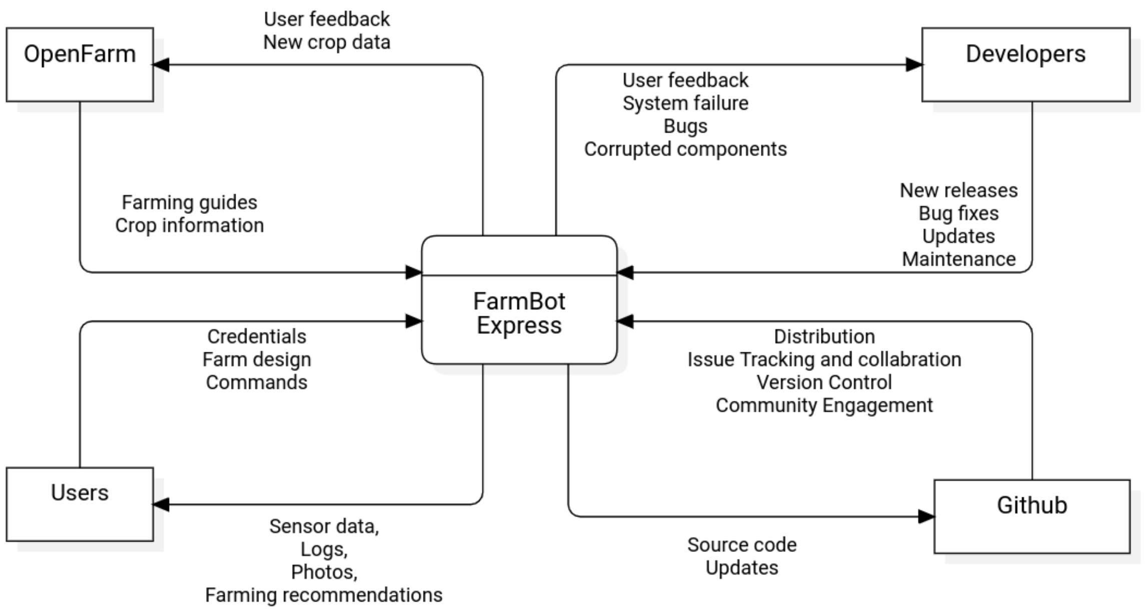


Figure 1.1: Context Diagram

1.2.1 System Perspective

FarmBot Express is a user-friendly, stand-alone automated gardening system designed to cater to a wide range of users, including beginners, students, researchers, home users, and even individuals with disabilities. This self-contained system leverages a robotic arm controlled by a central computer, a Raspberry Pi. The computer communicates wirelessly with FarmBot's onboard microcontroller, translating digital commands into precise movements for the motors. This enables the versatile 3-in-1 toolhead to perform tasks with high accuracy, such as planting seeds, delivering water directly to plant roots, and even managing weeds with compatible attachments. FarmBot Express integrates sensor systems to monitor environmental factors like soil moisture and temperature, allowing the system to adjust watering schedules for optimal plant growth. While some attachments are sold separately, the ability to detect and manage weeds through these attachments expands FarmBot's overall functionality.

Context Diagram Analysis:

The context diagram provides a high-level overview of how FarmBot Express interacts with its surroundings and the data that flows between them. This visual representation helps to understand the system's functionalities and its role in automating gardening tasks. The core entity is the FarmBot Express system itself, which interacts with three primary external entities:

User: This is the main entity that interacts with FarmBot Express. Users provide various inputs such as credentials for accessing the web interface, farm design data specifying the layout and plant types, and commands to direct the FarmBot's actions.

Developer: This external entitiy is responsible for developing the software and hardware components of FarmBot Express. Developers contribute to the system's functionality by creating new features, improving existing ones, and ensuring compatibility with external systems.

Github: Github is a cloud-based platform that hosts the open-source codebase for FarmBot Express. Developers use Github to collaborate on software development, track changes, and manage version control. The system interacts with Github to access the latest codebase, updates, and documentation.

OpenFarm: OpenFarm is an external database that provides crop information to FarmBot Express. The system communicates with OpenFarm.cc to access detailed plant data, including planting instructions, growth requirements, and other relevant information. This data helps users make informed decisions when designing their farms and scheduling planting tasks.

The data flow between these entities is crucial for the system's operation. Users send information to FarmBot Express, which in turn translates these instructions into actions within the farm environment. Sensor data from the farm is then relayed back to FarmBot Express, allowing the system to monitor and adjust its operations as needed. FarmBot Express may also provide data and recommendations back to the user.

1.2.1.1 System Interfaces

FarmBot Express relies on a network of non-user interfaces to facilitate communication and control between various hardware, software, and communication components.

These interfaces can be categorized as internal (within the FarmBot system) and external (between FarmBot and external systems). Communication between these interfaces occurs through established protocols, data formats, and a special operating system FarmBot OS.

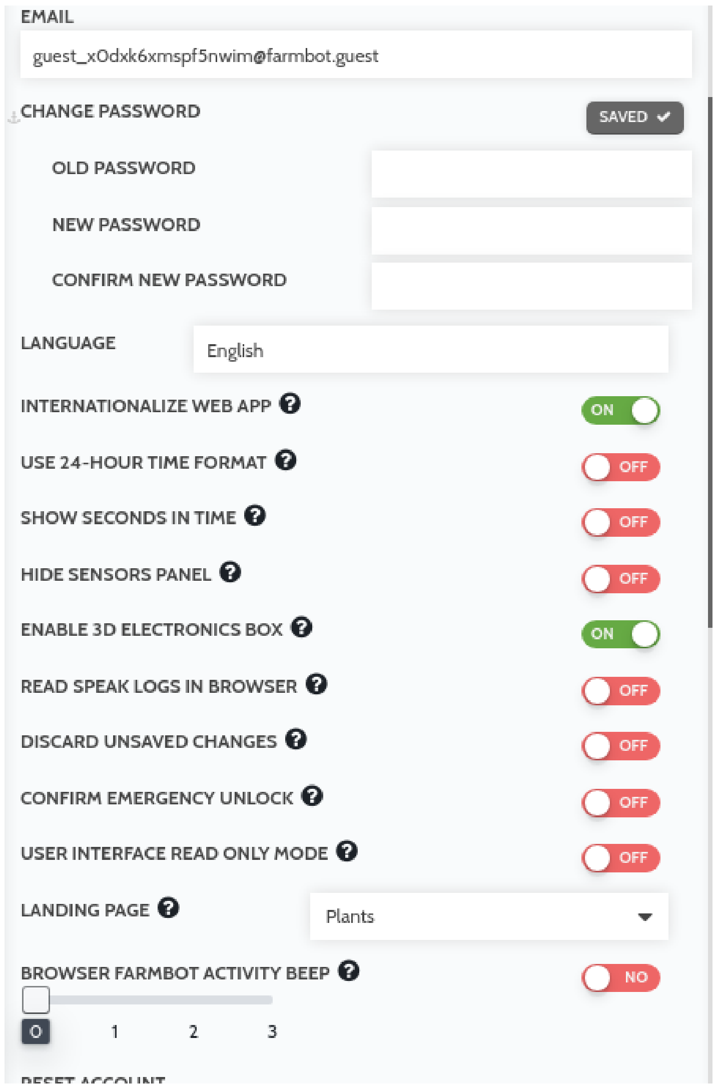
1.2.1.2 User Interfaces

FarmBot Express prioritizes user-friendliness by offering intuitive interfaces for interaction and control. These interfaces can be categorized as:

- **Web Interface:** This primary interface acts as the central hub for user interaction with FarmBot Express. Accessible through a web browser on a computer or mobile device, it mainly allows users to:

  Manage user accounts and profiles:

  The FarmBot Express Settings page lets users personalize their experience. They can change their password, language, and time format preferences. Additionally, users can choose what information to display in the interface, including sensor data. The page also offers options for managing saved changes, emergency actions, and user interface access restrictions.



EMAIL
guest_x0dxk6xmspf5nwim@farmbot.guest

CHANGE PASSWORD — SAVED ✓

OLD PASSWORD
NEW PASSWORD
CONFIRM NEW PASSWORD

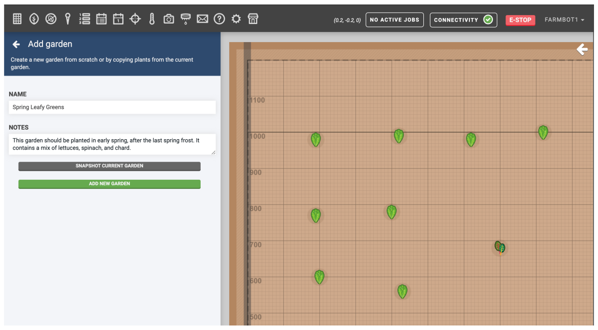
LANGUAGE: English

INTERNATIONALIZE WEB APP: ON
USE 24-HOUR TIME FORMAT: OFF
SHOW SECONDS IN TIME: OFF
HIDE SENSORS PANEL: OFF
ENABLE 3D ELECTRONICS BOX: ON
READ SPEAK LOGS IN BROWSER: OFF
DISCARD UNSAVED CHANGES: OFF
CONFIRM EMERGENCY UNLOCK: OFF
USER INTERFACE READ ONLY MODE: OFF
LANDING PAGE: Plants
BROWSER FARMBOT ACTIVITY BEEP: NO
0 1 2 3

Figure 1.2: Account Management

Create and manage farm profiles (size, layout, settings):

FarmBot Express's user interface empowers users to establish and manage detailed profiles for each of their farms. Within these profiles, users can define the farm's size, meticulously map out the garden layout, and configure various settings that optimize FarmBot's operation for that specific plot. This meticulous approach ensures precise automation and efficient resource utilization tailored to the unique needs of each individual farm.



Add garden

Create a new garden from scratch or by copying plants from the current garden.

NAME

Spring Leafy Greens

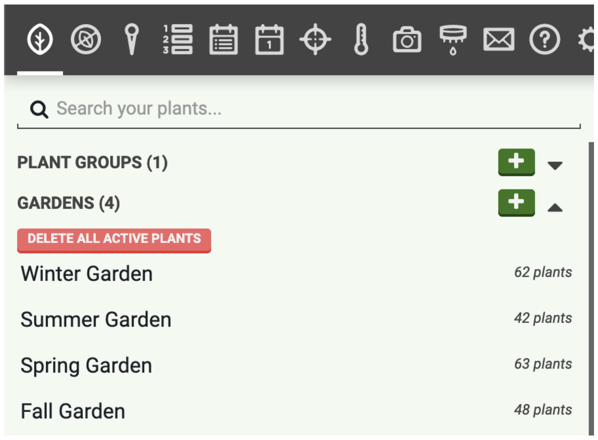
NOTES

This garden should be planted in early spring, after the last spring frost. It contains a mix of lettuces, spinach, and chard.

SNAPSHOT CURRENT GARDEN

ADD NEW GARDEN

Figure 1.3: Add Garden



Search your plants...

PLANT GROUPS (1)

GARDENS (4)

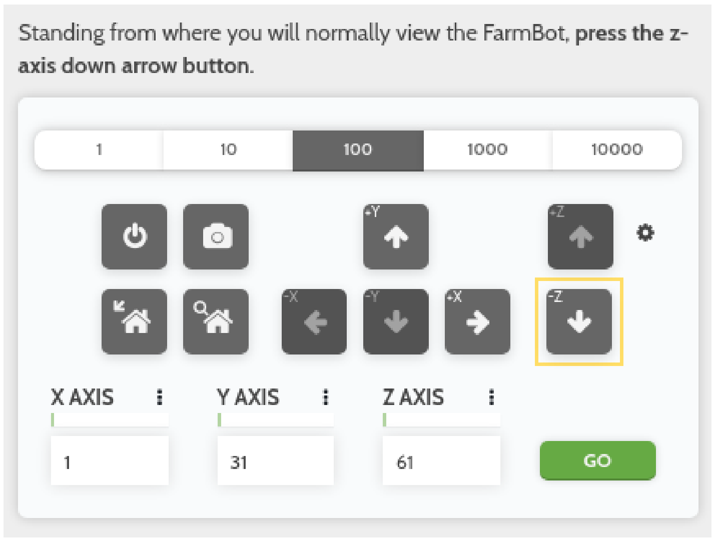
DELETE ALL ACTIVE PLANTS

Winter Garden	62 plants
Summer Garden	42 plants
Spring Garden	63 plants
Fall Garden	48 plants

Figure 1.4: Garden Profile

Manual Control:

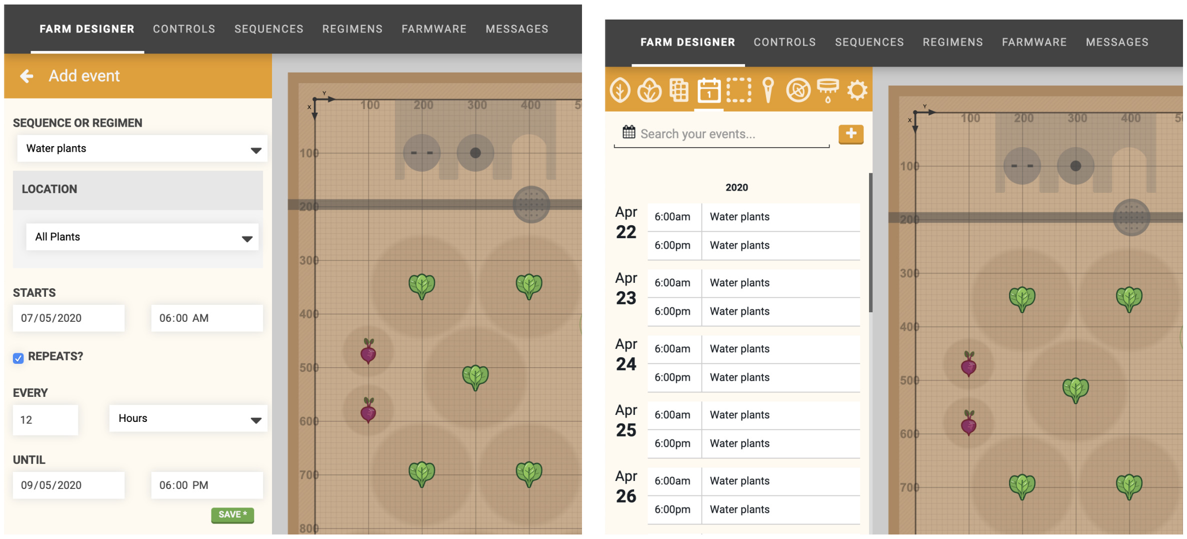
This interface serves as the manual control center for the FarmBot. Notice the directional arrows. One can use these controls to guide a robotic arm across your garden with pinpoint accuracy. The accompanying numbers (1, 10, 100, 1000, 10000) represent distance increments, allowing for precise movements tailored to the garden's layout and planting needs.


Standing from where you will normally view the FarmBot, **press the z-axis down arrow button.**

X AXIS	Y AXIS	Z AXIS
1	31	61

Figure 1.5: Manual Control

Schedule and monitor planting and watering tasks



SEQUENCE OR REGIMEN: Water plants

LOCATION: All Plants

STARTS: 07/05/2020 06:00 AM

[✓] REPEATS?

EVERY: 12 Hours

UNTIL: 09/05/2020 06:00 PM

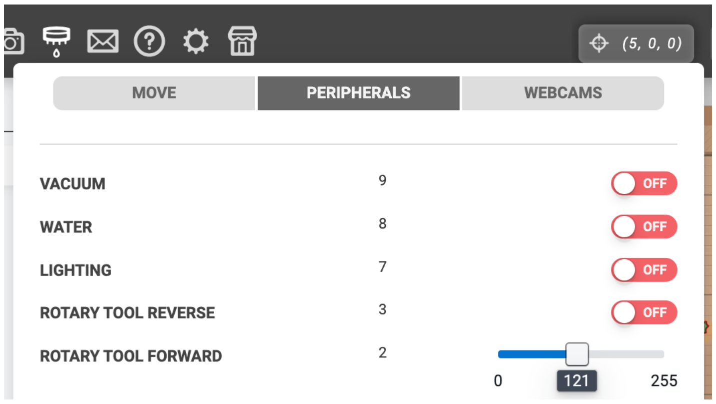
2020

Apr 22	6:00am	Water plants
	6:00pm	Water plants
Apr 23	6:00am	Water plants
	6:00pm	Water plants
Apr 24	6:00am	Water plants
	6:00pm	Water plants
Apr 25	6:00am	Water plants
	6:00pm	Water plants
Apr 26	6:00am	Water plants
	6:00pm	Water plants

Figure 1.6: Event Scheduling

The FarmBot Express Event Scheduler facilitates the creation and management of automated gardening tasks. Users can select the desired action type (e.g., watering, planting), specify the target location (entire garden layout or designated zones), define the start date and time for the event, and configure recurring schedules if necessary. By saving the event details, users can program FarmBot to perform these tasks autonomously, optimizing automated gardening routines.

Control Peripherals

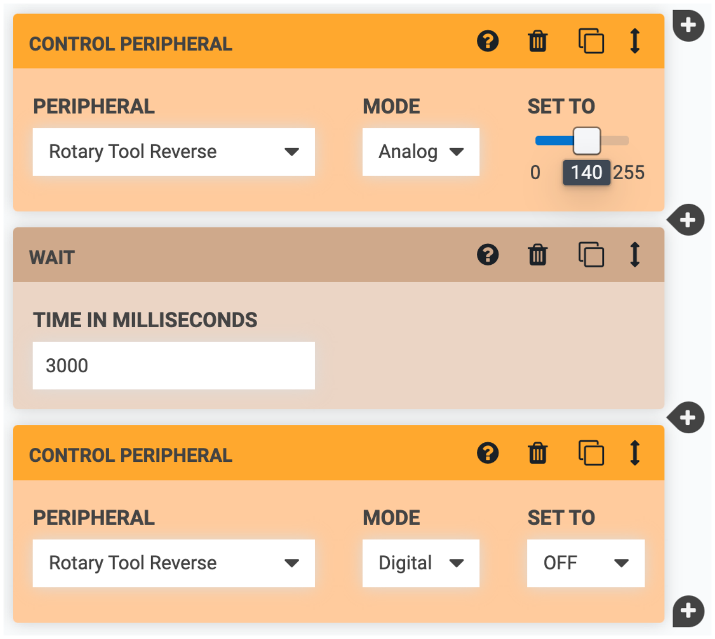


VACUUM 9 OFF
WATER 8 OFF
LIGHTING 7 OFF
ROTARY TOOL REVERSE 3 OFF
ROTARY TOOL FORWARD 2 0 121 255

Figure 1.7: Control Peripherals

The FarmBot Express software provides a dedicated ”Peripherals” tab within the control interface. This tab empowers users with real-time control and management of FarmBot’s peripheral devices.

Create sequential commands for various tasks



CONTROL PERIPHERAL
PERIPHERAL: Rotary Tool Reverse
MODE: Analog
SET TO: 0 140 255

WAIT
TIME IN MILLISECONDS: 3000

CONTROL PERIPHERAL
PERIPHERAL: Rotary Tool Reverse
MODE: Digital
SET TO: OFF

Figure 1.8: Sequential Command

FarmBot Express's web application prioritizes user control over every aspect of their automated gardening experience. However, manual oversight of each individual action becomes impractical for large-scale or complex tasks. Sequences address this challenge by enabling users to combine fundamental FarmBot commands, such as movement and peripheral control, into structured, multi-step programs. These programs, exemplified by a sequence that retrieves a watering nozzle, hydrates a specific plant and then returns the nozzle to its storage position, automate intricate tasks and optimize user time management within their flourishing gardens.

Visualize farm data and sensor readings

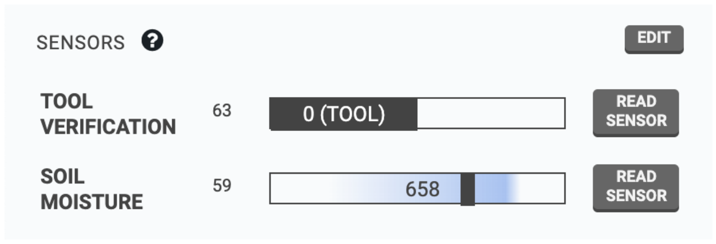


Figure 1.9: Sensor Configuration

Users can easily access the sensor data using the web application of the FarmBot Express. This interface enables users to handle many harsh situations by providing user and cloud components with real-time data.

1.2.1.3 Hardware Interfaces

Hardware interfaces define the physical connections between various hardware components within FarmBot Express. They specify the physical form (e.g., connectors, pins) and ensure proper electrical connections for data and power transfer. Examples include:

- **Tool Head Interface:** This interface defines the physical connection between the robotic arm and the 3-in-1 tool head, allowing for easy attachment and detachment. This is a proprietary FarmBot design with a standard electrical connector.

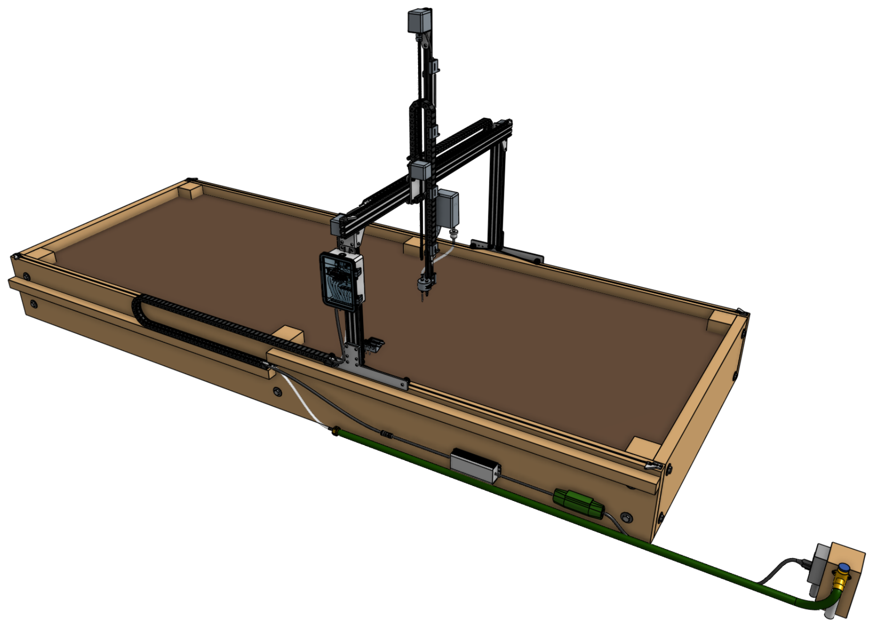


Figure 1.10: FarmBot Express v1.1 Hardware

- **Motor Control Interface:** This interface specifies the physical connection between the microcontroller and the FarmBot's motors. It ensures proper signal transmission for motor control.

FarmBot combines user-friendly input with a mechanical design. While FarmBot utilizes a robotic arm and various tools to automate tasks, the user interacts with the system through a user-friendly interface, likely a web application or mobile app. This interface allows users to plan their garden layout, schedule planting and watering tasks, and monitor plant growth. Essentially, the user interacts with FarmBot through a digital interface, while the physical actions in the garden are carried out by the mechanical components.

1.2.1.4 Software Interfaces

Software interfaces define the communication protocols and data formats used for software components to interact with each other. They specify the message structure, data types, and communication methods for seamless software interaction.

- **Microcontroller Software Interface:**

  This software interface defines how the central computer (often a Raspberry Pi) communicates with the onboard microcontroller. It uses an operating system named FarmBot OS developed for FarmBot to translate digital instructions into motor control commands. The FarmBot's onboard microcontroller utilizes its own embedded operating system to interpret numerical code received from the backend software and transmit sensor data back online. This embedded system includes functionalities for code interpretation, sensor data acquisition (taking photos), and data communication (accepting real-time commands and uploading photos to a web application).

- **Sensor Data Interface and Arduino Firmware:**

  Sensor Data Interface: This software interface defines the data format and communication protocol sensors use to transmit data (e.g., soil moisture, and temperature) to the central computer. However, the crucial role of the Arduino firmware comes into play before this data reaches the central computer.
  Arduino Firmware - The Bridge Between Sensors and FarmBot OS:

  The firmware flashed onto the Arduino or Farmduino microcontroller acts as the bridge between the physical sensors and the FarmBot OS software running on the Raspberry Pi. Here's a breakdown of its functionalities:

  Sensor Data Acquisition: The Arduino firmware actively reads sensor data (soil moisture, temperature, etc.) through the physical pins connected to the sensors. Data Formatting and Communication: The firmware translates the raw sensor data into a specific format (potentially adhering to established protocols) suitable for transmission. Communication with FarmBot OS: The formatted sensor data, along with readings from rotary encoders and other pins, is then sent back to the Raspberry Pi running the FarmBot OS software. This communication likely occurs using G and F codes.

- **Backend Interface:**

The Backend Interface serves as the backbone of FarmBot Express, facilitating seamless communication between the user-facing web interface and various internal software components. It acts as a central hub, managing essential data and functionalities:

User Management:

The interface handles user account creation, storage, and retrieval of user profile information. This ensures secure user authentication and facilitates personalized experiences within the FarmBot system. The more detailed version of user interaction and FarmBot's web application is argued in the User Interface part.

Farm Data Management:

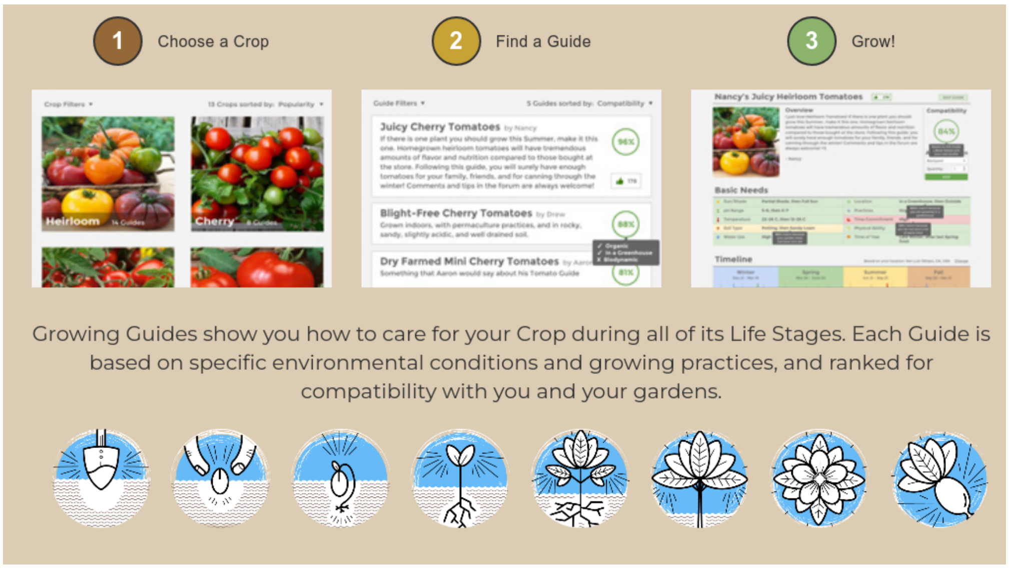
This aspect of the interface focuses on all data associated with a user's farm. It allows creation, editing, and access to farm profiles, which encompass details like size, plant layouts, settings, scheduled operations, statistics, data maps, and farm history. Essentially, the backend interface serves as the repository for all farm-specific information.

Equipment Profiles:

The interface maintains information about the user's FarmBot hardware, including its current configuration and operational status. This ensures compatibility between uploaded control code and the actual capabilities of the equipment. Additionally, the interface facilitates updates to equipment profiles as configurations change due to upgrades, maintenance, or other reasons.

Decision Support System Interaction:

The backend interface bridges the gap between the user-facing elements and the decision support system, an algorithmic component responsible for optimizing farm operations. It enables data exchange between the decision support system and other backend components. This allows the system to access relevant farm data (plant information, sensor readings, etc.) to make informed decisions regarding planting, watering, and other farm management tasks.



Juicy Cherry Tomatoes by Nancy
If there is one plant you should grow this Summer, make it this one. Homegrown heirloom tomatoes will have tremendous amounts of flavor and nutrition compared to those bought at the store. Following this guide, you will surely have enough tomatoes for your family, friends, and for canning through the winter! Comments and tips in the forum are always welcome!

Blight-Free Cherry Tomatoes by Drew
Grown indoors, with permaculture practices, and in rocky, sandy, slightly acidic, and well drained soil.

Dry Farmed Mini Cherry Tomatoes by Aaron
Something that Aaron would say about his Tomato Guide

Growing Guides show you how to care for your Crop during all of its Life Stages. Each Guide is based on specific environmental conditions and growing practices, and ranked for compatibility with you and your gardens.

Figure 1.11: OpenFarm.cc

Data Integration

While not explicitly mentioned as a core function of the Backend Interface, FarmBot Express might leverage cloud services to retrieve additional crop information from external databases like OpenFarm.cc. This integration enriches the decision support system by providing users access to a wider range of plant data for informed decision-making.

1.2.1.5 Communication Interfaces

Communication interfaces govern the exchange of data between FarmBot and external systems. They define the communication protocols and data formats used for internet connectivity, sensor data transmission, and potential future integrations.

- **Wi-Fi Interface:** This communication interface enables wireless communication between the central computer and the internet using the IEEE 802.11 standard (e.g., Wi-Fi b/g/n). This allows for remote user interaction and potential software updates.

Communication Mechanisms:

Communication between these interfaces occurs through the established protocols and data formats mentioned above. Here's a breakdown of some key communication mechanisms:

- **Serial Communication:** This is a common method for communication between the central computer and the microcontroller. It involves sending data one bit at a time over a dedicated serial communication line. There exists a USB communication in between Farmduino to Raspberry Pi.
- **Network Communication:** The Wi-Fi interface utilizes network communication protocols defined by the IEEE 802.11 standard to connect FarmBot to the internet thanks to the operating system specially designed for the FarmBot Express Project, named FarmBot OS. This sophisticated mechanism provides the user many opportunities to interact with his farm. Furthermore, the Raspberry Pi has an integrated Wi-Fi radio antenna and an ethernet connector module.

Standardization and Benefits:
By adhering to established communication protocols and utilizing standardized interfaces whenever possible (e.g., I2C, SPI, Wi-Fi), FarmBot ensures compatibility with readily available components. This facilitates future expansion with additional sensors or tools, allowing users to customize their FarmBot experience. Additionally, standardized communication protocols simplify troubleshooting and system maintenance.

1.2.1.6 Memory Constraints

Technical Feasibility: FarmBot Express leverages established technologies (Raspberry Pi, sensors, open-source software) and cloud integration, making it technically achievable for users with moderate technical knowledge. The cloud-based deployment of the decision support system significantly reduces memory constraints on the Raspberry Pi for FarmBot Express. The FarmBot software can focus on core functionalities with a moderate memory footprint, while the cloud handles intensive computations and data storage.

Economic Feasibility: While FarmBot Express offers automation and convenience, its cost-effectiveness depends on the user. It might be more suitable for hobbyists or small-scale gardeners who value these benefits. Large-scale farms would need to assess cost savings (labor, yield) against traditional methods.

Social Feasibility: FarmBot Express might require some technical knowledge for setup and operation, potentially limiting accessibility for non-tech-savvy users. However, it promotes water conservation and potentially reduces reliance on chemical pesticides through precise watering and weed management (with compatible attachments).

Overall, FarmBot Express is technically feasible, but economic and social factors require consideration. It caters well to home users and small-scale gardeners seeking automation, while larger operations need a cost-benefit analysis.

1.2.1.7 Operations

- **System Operations:**
  - Sow and water the seeds
  - Monitoring the seeds
  - Get crop data from database
  - Guide for optimal design
  - Scan garden and detect weeds
  - Display logs
  - Display external data
  - Authentication
- **User Operations:**
  - Create custom farm design
  - Create custom sequence models
  - Log in to web application

- Save farm designs
- Customize external tools

1.2.2 System Functions

Table 1.1 below, provides the system functions in tabulated manner.

Function	Description
Sow seeds, water and monitor	The system sows seeds, waters them individually and monitors the garden.
Customize farm design	The system allows user to create custom farm designs through the web app via a simple drag-and-drop designer.
Acquire crop data via external crop database	The FarmBot Cloud Service communicates with the external crop database OpenFarm.cc, which provides crop information to the web app.
Guide for optimal farm design	The system guides the user for the most optimal farm design using the crop data available.
Create custom sequences of farming actions	The system allows the user to create custom sequences to run on the device, through the web app with a custom sequence editor.
Scan garden and detect weeds	The system scans the garden using cameras and computer vision algorithms, detecting weeds in a timely manner.
Display system logs	The system displays to the user, the logs of the device through the web app.

Continued on next page

(Continued)

Display sensor data	The system displays to the user, the sensor data coming from the device through the web app.
Display photos	The system displays to the user, photos taken by the device's camera through the web app.
Authenticate user	The system authenticates the user by passing its web app credentials
Save garden designs	The system saves designs created by the user for reusability.
Customization of sensors, tooling	The system firmware allows users to implement custom sensors and tooling on the device.

1.2.3 Stakeholder Characteristics

The stakeholders of the system are range from students, researchers, robotic artwork creators, to home users, people with disabilities. Characteristics of the stakeholders are explained below:

- **Students:** For students in any level of study from K12 (Kindergarten and grades 1-12) to university education, FarmBot provides a practical, fun and hands-on learning experience for STEM (Science, Technology, Engineering and Mathematics) learning objectives such as: **robotics**, **coding**, **soil science**, **biology**, and **much more**. They can be called novices in the domain of FarmBot. They do not have the necessary knowledge to fully comprehend the hardware-software interactions. They are naturally curious which encourages them to come up with new features, ways to use the device. They want to have the freedom to use their creativity. In addition to that, the students do not have a long attention span and it might be hard to keep them engaged.

- **Researchers:** For researchers, there are various ways to take advantage of the FarmBot. It offers repeatability, scale, speed, and low cost to researchers.

  - **Repeatability**: The human error factor needs to be eliminated which leads to more accurate and repeatable results while experimenting. In addition, they want to schedule events (experiments) easily.
  - **Scale**: Researchers want to easily test multiple groups of crops and run multiple experiments concurrently.
  - **Speed**: Researchers want to create their own sequences and collect data automatically at any frequency, run experiments 24/7.
  - **Cost**: Researchers want to avoid high labor costs

  This class of stakeholders is highly educated therefore can capitalize on FarmBot being highly customizable, integratable. They would like to create custom tooling (eg. brush heads for coral farming), custom regimens (eg. for speeding up coral growth) for their experiments. Collecting accurate data in a timely fashion is of utmost importance to them. Precise and continous data flow from sensors, cameras free the researchers from labor intensive tasks and potential for error. This class uses FarmBot for **phenotyping research**, **photogrammetry**, **off-world farming research** and such.

- **Robotic Artwork Creators:** This class of stakeholders approach to using FarmBot is different from the others. Their perspective is more creative and artistic. They use the various sensors and cameras of the device, feed the data from them into different algorithms such as **stable diffusion** to create art pieces. For this class, ability to customize farm designs and sequence customizing are crucial features. They have, similar to the researchers class, the required knowledge of technology and they are generally well educated.

- **Home User:** The home user class is usually driven to use FarmBot because of emerging problems in the world such as climate change, environmental pollution due to plastic packaging and excessive pesticide use. This class takes pride in being able to grow its food itself and being able to grow challenging plants. They are not experts on the technology, they have general knowledge. They usually use their backyards, small gardens for farming. They generally do not have a lot of spare time to tend to their garden each day.

- **People with disabilites:** These people suffer from various physical and mental limitations in their day-to-day lives. They are not able to farm on their own using conventional farming techniques (eg. can't clean the garden from weeds). Centres for disabled people can use FarmBot for horticultural therapy (the art or practice of garden cultivation and management) or vocational training. This would help them integrate with the society and give them a sense of autonomy. They do not have the technical expertise due to physical and mental handicaps.

1.2.4 Limitations

- **Regulatory Requirements:** The FarmBot project is 100% open-source. Hardware designs, software source code, and documentation are openly shared. However, technical support for DIY builds, customizations are not to be provided. In addition, FarmBot must comply with the local farming laws and regulations.

- **Hardware Limitations:** The custom electronics board needs a dedicated processor in order to monitor the rotary encoders at high speed and a Raspberry Pi 3 to be used as the brain communicating through the web. In addition, FarmBot has four stepper motors and needs to provide a tested rotary encoder for closed-loop feedback control. Lastly, FarmBot needs to be able to work with custom tooling.

- **Interfaces to Other Applications:** FarmBot must be able to communicate with external resources through its cloud services for crop information. Interacting with FarmBot from the web must be done using FarmBot.JS or FarmBot.Py and requests must be done according to the provided API.

- **Parallel Operation:** Parallel operation is crucial to FarmBot. It needs to be able to handle multiple crops of various types and sizes. Also, FarmBot needs to be able to handle data flow from sensors and cameras in parallel.
- **Audit Functions:** FarmBot web app handles user credentials and wifi credentials of the user. The way this data is stored and protected may require auditing from relevant organizations.
- **Control Functions:** Since FarmBot is an open-source project, the control function does not exist. In contrast to being central, it is distributed to the community.
- **Higher-order Language Requirements:** The frontend for the web application is developed using TypeScript and ReactJS UI library. The backend is developed using Ruby on Rails, and PostgreSQL. Google Cloud is used for file storage. The message broker sub-component is built as a custom RabbitMQ instance. CeleryScript is used for handling data exchange between systems in an asynchronous and predictable way. Elixir is used to develop FarmBot OS to handle low level details. Lastly, the firmware for Arduino is written in C++ to communicate effectively with the hardware.
- **Signal Handshake Protocols:** HTTPS is used for communication between client and server, therefore SSL Handshake Protocol is used.
- **Quality Requirements:** Precision, scalability, automation, space efficiency, continuous land use, ease of customization are important quality requirements of FarmBot.
- **Criticality of the application:** The FarmBot system is not critical. The effects of system failure would not be drastic.
- **Safety and Security Considerations:** The user and wifi credentials must be stored safely. Also, the device shall not harm the user while executing actions.

- **Physical/Mental Considerations:** The user must be able to at least interact with the drag-and-drop no-code programming environment and must have the basic physical and mental capability to set up the device.
- **Limitations that are sourced from other systems, including real-time requirements from the controlled system through interfaces:** The FarmBot system communicates with the crop database OpenFarm as an external resource. This information shall be updated regularly. In addition, the system shall also communicate with user devices such as laptops, mobile phones and data flow must be real-time. For example, taking pictures from the camera or sensor data logs.

1.3 Definitions

- HTTPS: Hypertext Transfer Protocol Secure (HTTPS) is an extension of the Hypertext Transfer Protocol (HTTP). It uses encryption for secure communication over a computer network, and is widely used on the Internet.
- SSL: SSL, or Secure Sockets Layer, is an encryption-based Internet security protocol.
- API: A set of functions and procedures allowing the creation of applications that access the features or data of an operating system, application, or other service.
- I2C: Inter-Integrated-circuit (I2C) is a simple, bidirectional two-wire synchronous serial bus that requires only two wires to transmit information between devices connected to the bus.
- SPI: Serial Peripheral Interface (SPI) is a de facto standard (with many variants) for synchronous serial communication, used primarily in embedded systems for short-distance wired communication between integrated circuits.

2. References

[1] Iso/iec/ieee international standard - systems and software engineering – life cycle processes – requirements engineering. *ISO/IEC/IEEE 29148:2018(E)*, pages 1–104, 2018.

3. Specific Requirements

3.1 External Interfaces

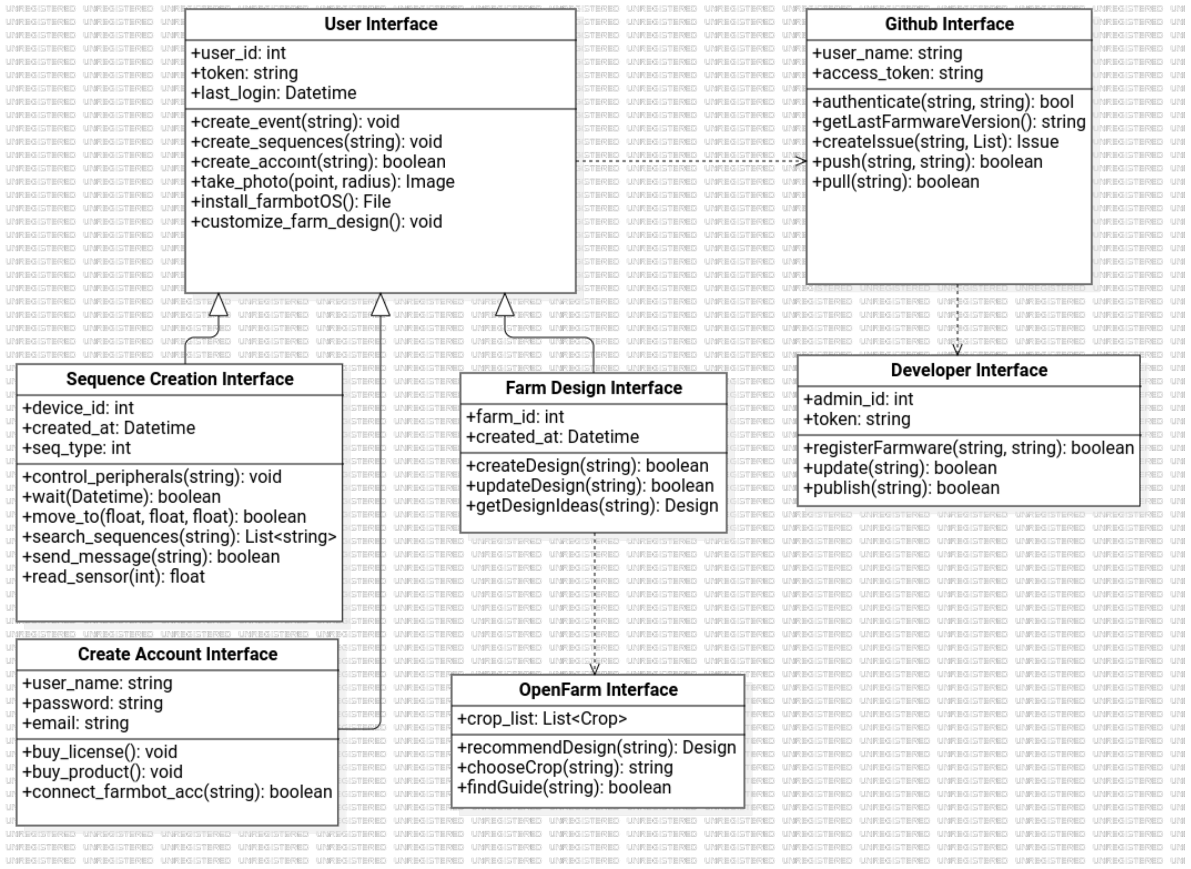

User Interface
+user_id: int
+token: string
+last_login: Datetime
+create_event(string): void
+create_sequences(string): void
+create_accoınt(string): boolean
+take_photo(point, radius): Image
+install_farmbotOS(): File
+customize_farm_design(): void
Github Interface
+user_name: string
+access_token: string
+authenticate(string, string): bool
+getLastFarmwareVersion(): string
+createIssue(string, List): Issue
+push(string, string): boolean
+pull(string): boolean
Sequence Creation Interface
+device_id: int
+created_at: Datetime
+seq_type: int
+control_peripherals(string): void
+wait(Datetime): boolean
+move_to(float, float, float): boolean
+search_sequences(string): List<string>
+send_message(string): boolean
+read_sensor(int): float
Farm Design Interface
+farm_id: int
+created_at: Datetime
+createDesign(string): boolean
+updateDesign(string): boolean
+getDesignIdeas(string): Design
Developer Interface
+admin_id: int
+token: string
+registerFarmware(string, string): boolean
+update(string): boolean
+publish(string): boolean
Create Account Interface
+user_name: string
+password: string
+email: string
+buy_license(): void
+buy_product(): void
+connect_farmbot_acc(string): boolean
OpenFarm Interface
+crop_list: List<Crop>
+recommendDesign(string): Design
+chooseCrop(string): string
+findGuide(string): boolean

Figure 3.1: External Interfaces Class Diagram

- **User Interface** The user interface is the main way for the user to interact with the system. Every user has his own token and ID. Users can do some of the basic operations here, such as sequence creation, creating an account, and farm design, which can be considered as sub-interfaces of the user interface.

- **OpenFarm Interface** The OpenFarm is a free and open database for farming and gardening knowledge. The OpenFarm API is used to fetch crop data and guides for crops. The system uses OpenFarm API to get crop data and guides for crops.
- **GitHub Interface** GitHub is a code hosting platform for version control and collaboration. The system uses GitHub API to fetch the latest version of the software. It can be easily seen that the user depends on this interface to get the latest version of the software.
- **Developer Interface** The developer interface is used to interact with the system for development purposes. Developers can create new features, fix bugs, etc. Developers use this interface to contribute to the system. With further investigation, one can see that users depend on this interface to get the latest version of the software since the maintenance of the GitHub interface depends on developers.

3.2 Functions

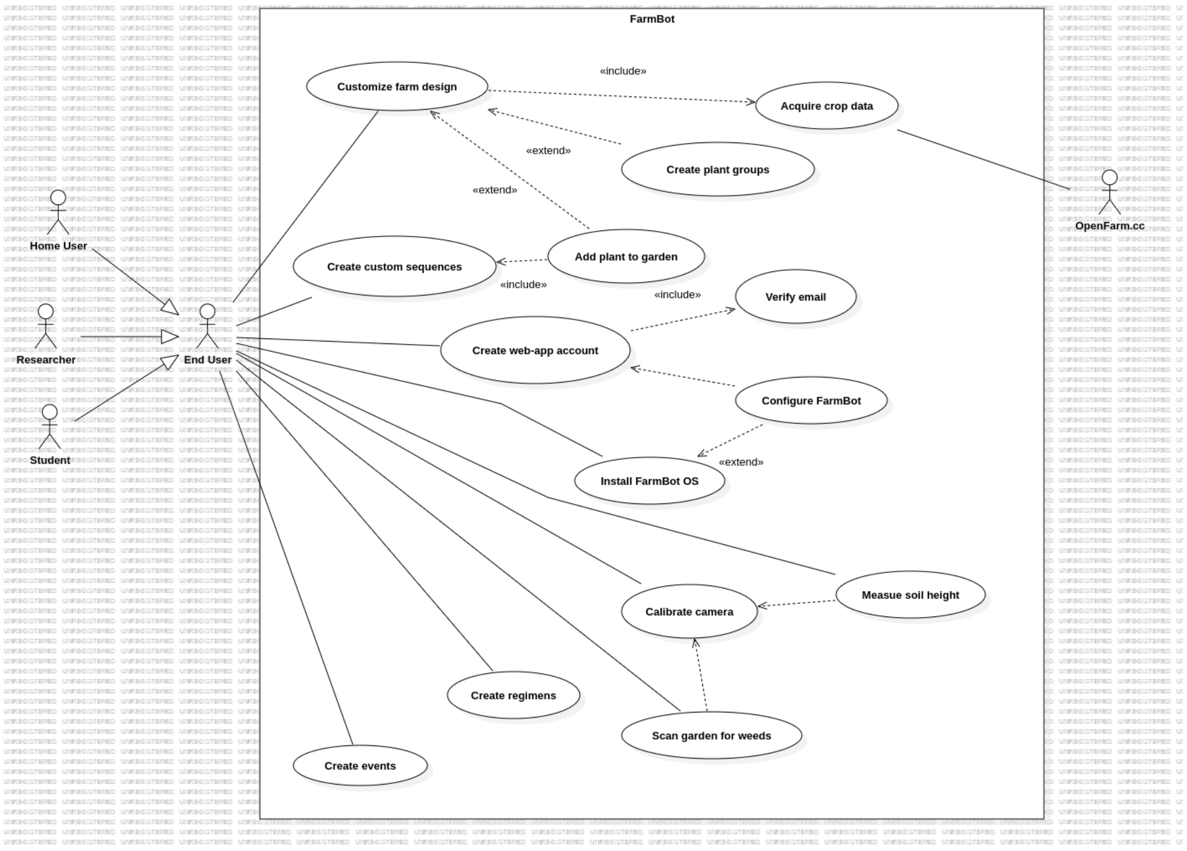


Figure 3.2: Use Case Diagram

Use case name	Create web-app account
Actors	End User
Description	The end user creates an account on the FarmBot Web App.
Preconditions	The end user must have a valid e-mail address. The user must accept the privacy policy and terms of use.
Data	User credentials: e-mail, name, password
Response	E-mail verification link which logs user into the app
Stimulus	The user interacts with the "CREATE AN ACCOUNT" widget.
Normal Flow	1. User goes to my.farm.bot 2. Enters name, e-mail and password into the widget 3. User checks to agree the privacy policy and terms of use 4. User clicks "CREATE ACCOUNT" button 5. User clicks on the verification link sent to their e-mail.
Alternative Flow	-
Exception Flow	4. If user leaves any of the fields blank, an error pop-up appears alerting that these fields cannot be left blank.
Post Conditions	The users account must be confirmed and user must be taken to the message center.
Comments	-

Table 3.1: Tabular description of create web-app account case

Use case name	Install FarmBot OS
Actors	End User
Description	The user installs FarmBot OS on the Raspberry Pi
Preconditions	The user has a microSD card with sufficient capacity, a card reader and has downloaded Raspberry Pi Imager.
Data	-
Response	User should see a solid red LED and a a steadily flashing green LED on the Raspberry Pi.
Stimulus	User goes to os.farm.bot and clicks on the download button for respective model of FarmBot Express
Normal Flow	1. User downloads the latest FarmBot OS image 2. User writes the FarmBot OS onto the microSD card using Raspberry Pi Imager 3. User inserts the microSD card into the Raspberry Pi 4. User plugs in the power source
Alternative Flow	-
Exception Flow	2. microSD cards larger than 32GB will not work. Drag and drop or copy-paste will not work. Raspberry Pi Imager must be used.
Post Conditions	LED lights should be lit, indicating that Raspberry Pi has adequate power and is busy booting up.
Comments	-

Table 3.2: Tabular description of install farmbot os case

Use case name	Configure FarmBot
Actors	End User
Description	The user configures FarmBot to connect to WiFi network and web-app account
Preconditions	The user must have a web app account User has flashed FarmBot OS onto microSD card inserted into Raspberry Pi User has an ETHERNET or WIFI connection
Data	WiFi network password and name, web-app credentials
Response	Status ticker at my.farm.bot shows that FarmBot has come online and is sending messages.
Stimulus	Initial boot-up of FarmBot
Normal Flow	1. User connects to configurator 2. User chooses a network type (ETHERNET or WIFI) 3. User enters their web app credentials 4. User submits configuration 5. User makes sure that FarmBot is online
Alternative Flow	1. If captive portal does not automatically open up, user should try navigating to setup.farm.bot or 192.168.24.1 on a web browser. 2. If user needs to know the network interface name or MAC address, presses the circular ”i” icon. 3. If user is self-hosting the web app, should go to their custom server URL
Exception Flow	1. If user is using a smartphone, they might need to disable cellular data in order to connect to the configurator. 5. If there is a problem with the configuration, the Configurator will restart within 5 minutes.
Post Conditions	Connection between FarmBot OS and user device is terminated. Status ticker is indicating that FarmBot is online.
Comments	-

Table 3.3: Tabular description of configure FarmBot case

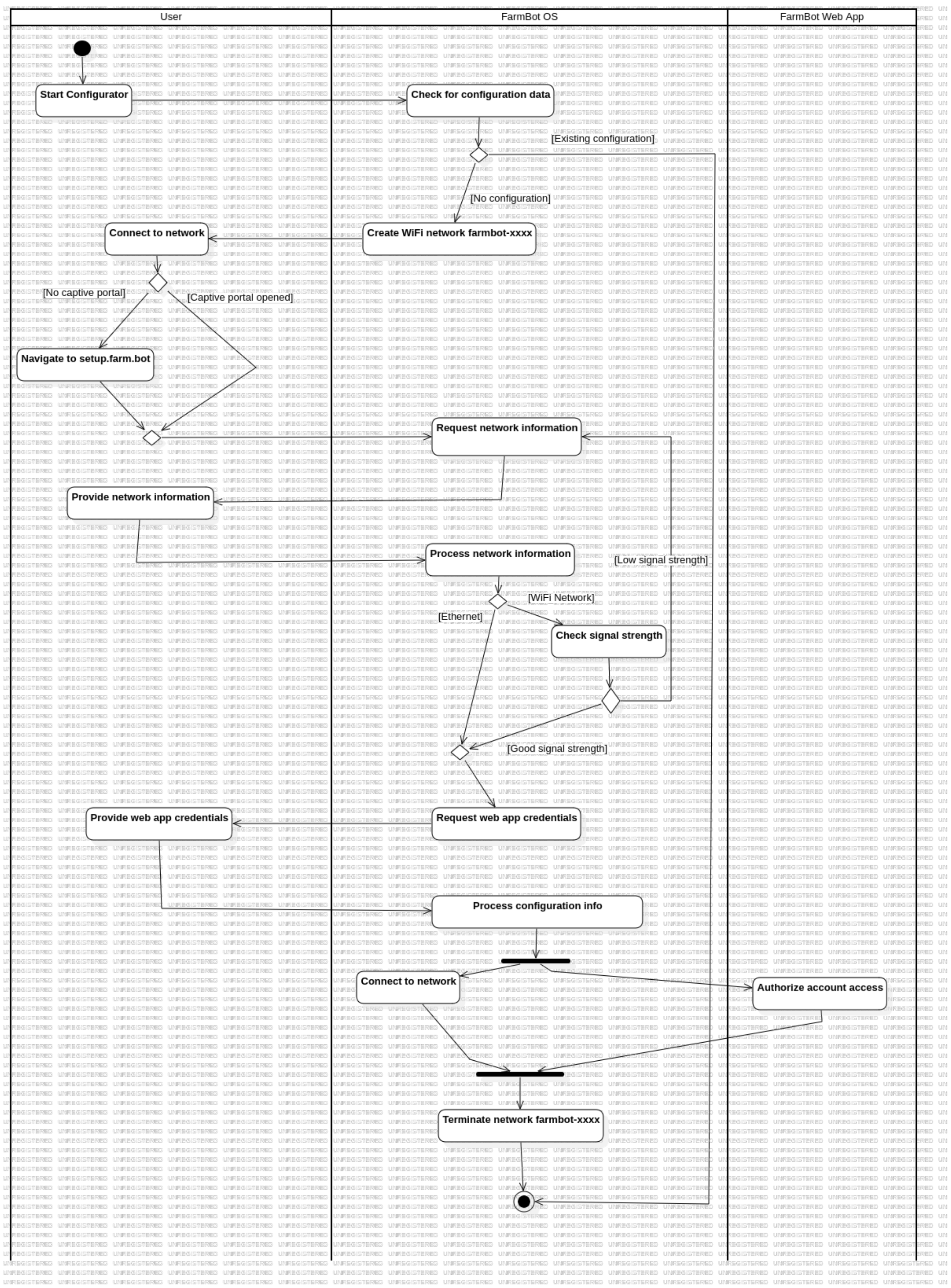


Figure 3.3: Activity Diagram for "Configure FarmBot" use case

Use case name	Create events
Actors	End User
Description	User schedules to initiate sequences automatically
Preconditions	The user has a verified web-app account New events must be scheduled at least one minute into the future FarmBot is not performing an update
Data	Sequence or regimen data
Response	The event will show up in the agenda.
Stimulus	User navigates to the events panel on the farm designer page and presses the ”+” button for creating an event.
Normal Flow	1. User presses the ”+” button in the events panel 2. User chooses the sequence or regimen to be executed 3. User provides a START date and time. 4. User presses the save button to save the event.
Alternative Flow	3. By default current date and current time + 3 minutes will be used. 3. Sequence events can be set to REPEAT EVERY custom interval and UNTIL a particular stop date and time
Exception Flow	3. New events must be scheduled at least one minute into the future, input is rejected. 3. Since FarmBot performs updates at 3AM, scheduling events between 2AM and 4AM is discouraged.
Post Conditions	The agenda is showing the created event and its occurrences into the future.
Comments	-

Table 3.4: Tabular description of create events case

Use case name	Create regimens
Actors	End User
Description	User creates a regimen to easily take care of a plant throughout its entire life and allows reuse of ”recipe” season after season.
Preconditions	User has created sequences to add to the regimen
Data	Sequence data, date and time
Response	The regimen will be visible on the scheduler page
Stimulus	User presses the ”+” button on the Regimens panel
Normal Flow	1. User gives a descriptive name to the regimen loaded into the panel 2. User presses the ”SCHEDULE ITEM” button 3. User selects sequence to add to the regimen. 4. User picks a TIME and DAYS (relative to plant) for the sequence to run 5. User presses ”SAVE” button to save the regimen.
Alternative Flow	1. User can optionally assign it a color 3. User can delete falsely added sequences by clicking the trash can icon
Exception Flow	4. Since FarmBot performs updates at 3AM, scheduling events between 2AM and 4AM is not allowed.
Post Conditions	The scheduler panel shows the regimen. Success pop-up is shown.
Comments	-

Table 3.5: Tabular description of create regimens case

Use case name	Create custom sequences
Actors	End User
Description	User combines the basic commands of FarmBot into more complex actions with multiple steps.
Preconditions	User has a verified web-app account and has created a farm.
Data	Commands, step parameters
Response	The green save button will turn gray and will have its text set to ”SAVED”.
Stimulus	User presses the ”+” button on the Sequences panel
Normal Flow	1. User presses the ”+” button to create a new sequence 2. User enters a unique name for the sequence 3. User adds a command by pressing a ”+” button using the command palette 4. User defines step parameters 5. User names each step 6. User saves the sequence
Alternative Flow	1. User can select an existing sequence to edit 1. User can use the search box to find a sequence 2. User can optionally assign a color to the sequence 3. User can enter full-screen editor, where commands can be dragged-and-dropped. 5. User can take advantage of AI to write the sequence name and description
Exception Flow	2. If the name is not unique, it is rejected.
Post Conditions	The green ”SAVE” button turns gray and its text is set to ”SAVED”.
Comments	-

Table 3.6: Tabular description of create custom sequences case

Use case name	Add plant to garden
Actors	End User
Description	User adds plant to their garden
Preconditions	User has a verified web-app account and has completed FarmBot setup
Data	Crop data from OpenFarm
Response	The farm layout view will show the plant
Stimulus	User navigates to the plants panel and presses a button
Normal Flow	1. User presses the ”+” button in the plants panel. 2. User chooses a crop from panel 3. User drags and drops the crop image into the map
Alternative Flow	2. User can check out crop details 3. User can click ”ADD TO MAP” button and then drag the crop onto the map
Exception Flow	3. If user wants to delete a plant, clicks on it and presses the ”DELETE” button on the panel that popped up
Post Conditions	The map shows a crop icon at the desired location by the user
Comments	-

Table 3.7: Tabular description of add plant to garden case

Use case name	Acquire crop data
Actors	OpenFarm.cc
Description	Information about different crops are fetched
Preconditions	For certain requests, tokens may be needed
Data	Garden data, crop data, guides for crops
Response	A JSON payload
Stimulus	User chooses a crop on the plants panel
Normal Flow	1. A request to the token endpoint is sent with user credentials 2. Token is inserted to the request to be made 3. GET/POST/PUT/DELETE request is sent with required payload to the desired endpoint 4. The response from the server is parsed
Alternative Flow	1. Some requests can be made without a token
Exception Flow	4. The API responds with an error JSON payload in the case something went wrong
Post Conditions	-
Comments	-

Table 3.8: Tabular description of acquire crop data case

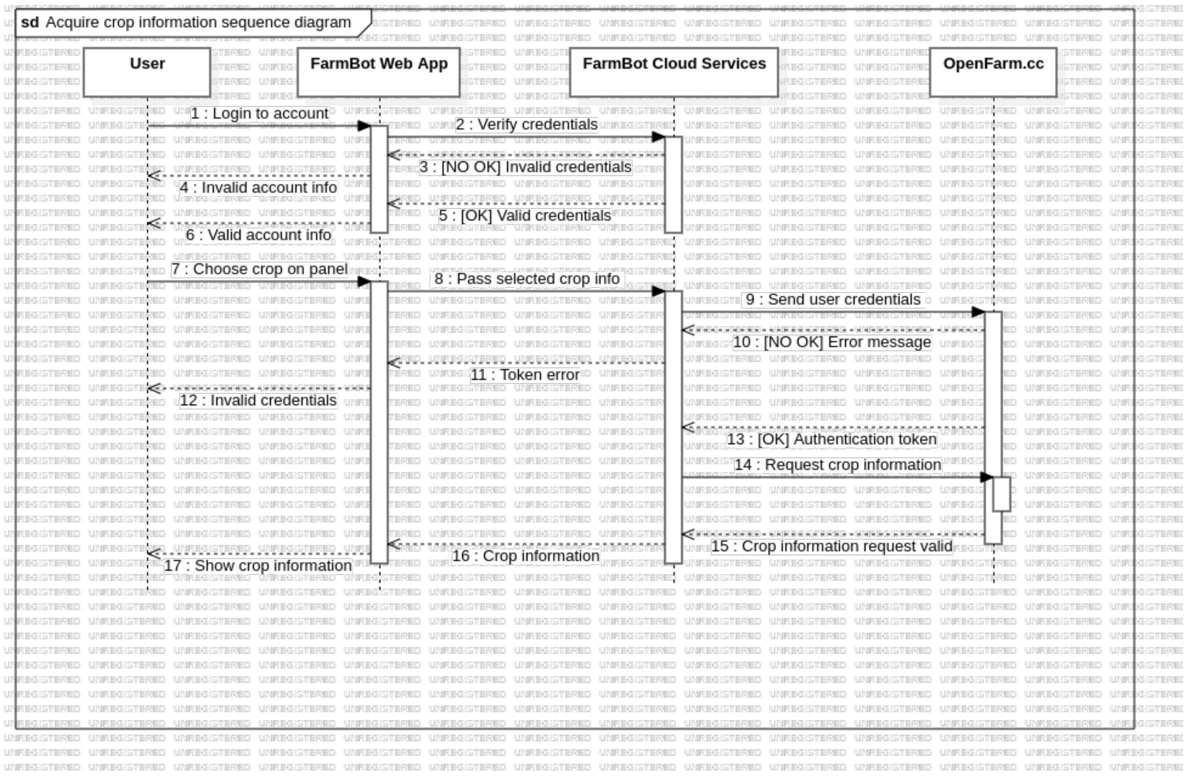

sd Acquire crop information sequence diagram
User
FarmBot Web App
FarmBot Cloud Services
OpenFarm.cc
1 : Login to account
2 : Verify credentials
3 : [NO OK] Invalid credentials
4 : Invalid account info
5 : [OK] Valid credentials
6 : Valid account info
7 : Choose crop on panel
8 : Pass selected crop info
9 : Send user credentials
10 : [NO OK] Error message
11 : Token error
12 : Invalid credentials
13 : [OK] Authentication token
14 : Request crop information
15 : Crop information request valid
16 : Crop information
17 : Show crop information

Figure 3.4: Sequence Diagram for "Acquire crop data" use case

Use case name	Create plant groups
Actors	End User
Description	User groups plants, weeds or points in the garden
Preconditions	User has a verified web-app account and has completed setup
Data	-
Response	Group is shown on the map
Stimulus	User navigates to the GROUPS panel and presses a button
Normal Flow	1. User opens the PLANT GROUPS, WEED GROUPS or POINT GROUPS section of respective panels 2. Edit group panel will be loaded with all plants, points, weeds selected 3. User applies filters to narrow down selection 4. User enters a name for the group 5. User presses the back arrow button to save
Alternative Flow	2. User can select the multi-select mode in the farm designer and press ”CREATE GROUP”
Exception Flow	-
Post Conditions	User must be able to use the groups within the sequence builder
Comments	-

Table 3.9: Tabular description of create plant groups case

Use case name	Customize farm design
Actors	End User
Description	User designs the layout of their garden graphically
Preconditions	User has a verified web-app account and has completed setup
Data	Crop data
Response	Layout shown on the map
Stimulus	User navigates to the GROUPS panel and presses a button
Normal Flow	1. The user sets their origin for the map 2. User chooses a crops to add 3. User drags and drop the crops image to the map 4. User moves the crops to the desired location
Alternative Flow	-
Exception Flow	-
Post Conditions	The plants should be at least 10mm away from each other
Comments	-

Table 3.10: Tabular description of customize farm design case

Use case name	Calibrate camera
Actors	End User
Description	User calibrates camera to allow images to be rotated, positioned and calibrated
Preconditions	User has the camera calibration card
Data	-
Response	Photos taken from the garden are shown in the farm designer with calculated values
Stimulus	User presses the "CALIBRATE" button
Normal Flow	1. User places the calibration card in the bed 2. User moves FarmBot that the camera is positioned directly over the card 3. User raises the z-axis as high as it can be raised 4. User opens the photos panel 5. Expands camera calibration section and presses "CALIBRATE" button 6. FarmBot takes photos moving in different directions and upload them to web-app
Alternative Flow	-
Exception Flow	1. If card is placed outside of the field of view, FarmBot does not detect it 1. If poor lighting, user should try toggling FarmBot's LED light 6. If card cannot be detected, problematic photos are uploaded. FarmBot returns to start
Post Conditions	The photos taken of the garden must line up with the grid in the farm designer.
Comments	-

Table 3.11: Tabular description of calibrate camera case

Use case name	Scan garden for weeds
Actors	End User
Description	User scans the whole garden for weeds
Preconditions	User has calibrated FarmBot's camera
Data	-
Response	The results are shown on the farm designer as photos
Stimulus	User presses "RUN" button
Normal Flow	1. User creates a point grid 2. User groups the points 3. User creates sequences for scanning 4. User presses RUN and starts scan
Alternative Flow	-
Exception Flow	4. After run, if the output is problematic, locations, spacing and number of points in the grid may be adjusted 4. Color range and other weed detection parameters may be adjusted
Post Conditions	The photos of the garden line up in the farm designer and weeds are indicated
Comments	-

Table 3.12: Tabular description of scan garden for weeds case

Use case name	Measure soil height
Actors	End User
Description	Using computer vision software, FarmBot detects average z-axis height of soil
Preconditions	User has calibrated FarmBot's camera
Data	-
Response	Soil height values
Stimulus	User presses "MEASURE" button
Normal Flow	1. User presses the "MEASURE" button 2. FarmBot takes a photos from different directions 3. The photos are processed to determine the soil height 4. Processed images are uploaded and soil height point is created
Alternative Flow	-
Exception Flow	2. If photos are problematic, user needs to recalibrate camera 4. If there are outliers in the dataset, user needs to remove them.
Post Conditions	There are soil height points with measured heights across the farm designer map
Comments	-

Table 3.13: Tabular description of measure soil height case

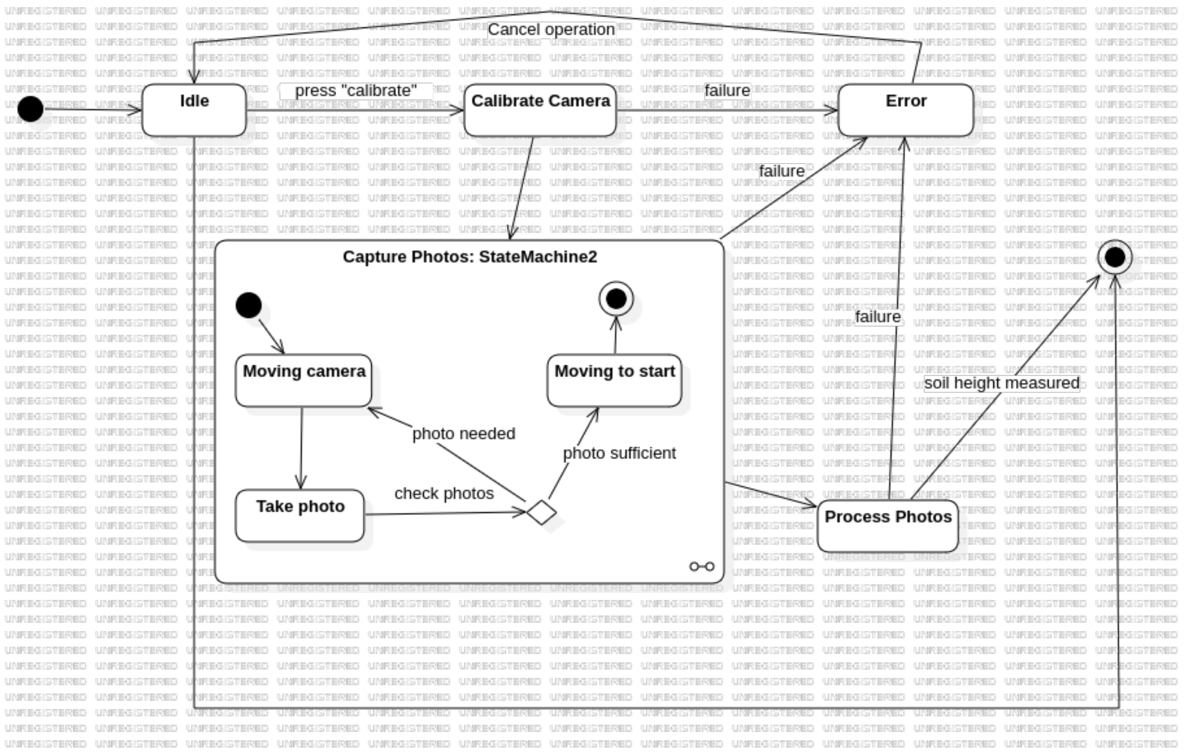


Figure 3.5: State Diagram for ”Measure soil height” use case

3.3 Logical Database Requirements

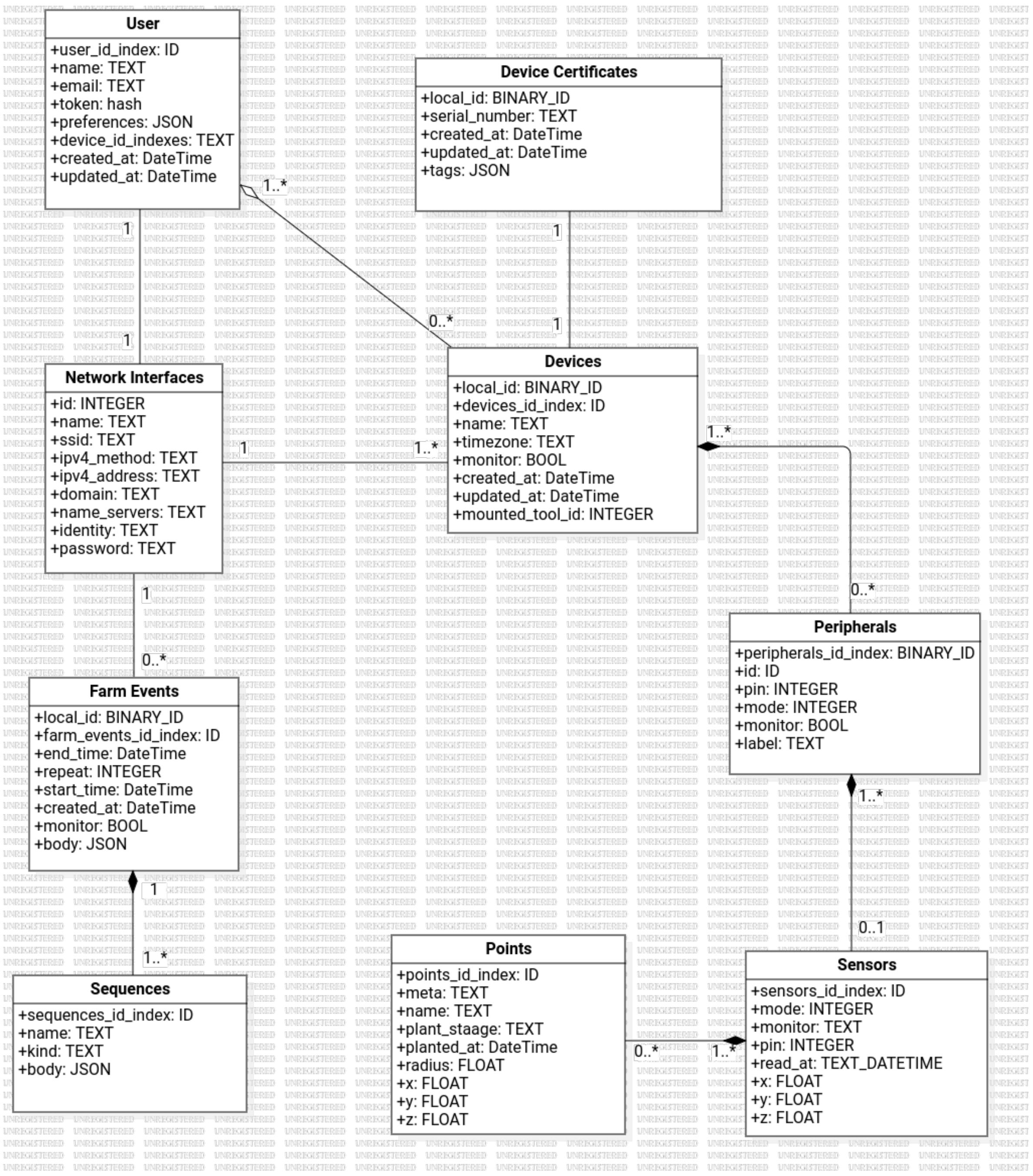


Figure 3.6: Logical Database Requirements Class Diagram

The logical database requirements diagram can be seen in Figure 3.6. The dataflow diagram shows the relationships between the different entities in the database.

1. **User** - The user entity stores the user's information, such as their name, email, password, and role. The user can have multiple farm devices. The user can create multiple farm events using the network interface of the web app.

2. **Devices** - The farm device entity stores the device's information, such as its name, device type, and status. The device can access multiple farm events using the web app network interface. The farm device can have multiple farm sequences, too. Furthermore, the device can have multiple peripherals with external sensors. Finally, every device has its own device certification.

3. **Network Interface** - The network interface entity stores the network interface's information, such as its name, type, and status. The network interface basically a bridge between the device and the web app. Here the protocols, IP addresses, and ports are stored.

4. **Farm Events** - The farm event entity stores the event's information, such as its name, type, and status. The farm event can have multiple farm sequences. It is a user-created event that can be scheduled to run at a specific time.

5. **Sequences** - The farm sequence entity stores the sequence's information, such as its name, type, and status. The farm sequence can have multiple farm commands.

3.4 Design Constraints

FarmBot software is an open-source development project. Therefore clear guidelines for contribution, documentation and version control is needed. The software is licensed with the MIT license approved by the Open Source Initiative and a free license by Free Software Foundation. The documentation is licensed under the CCO Public Domain Dedication.

FarmBot collects user data such as their web-app credentials, photos from their gardens, their plant schedules. The software comply with the data privacy regulations GDPR (General Data Protection Regulation), CCPA (California Consumer Privacy Act).

FarmBot software has certifications to prove that it is compliant with user safety regulations. The Raspberry Pi comply with RoHS (Restriction of Hazardous Substances), FCC (Federal Communications Comission) certifications. Farmduino is compliant with CE certification meaning that the product conforms European health, safety, environmental protection standards and is compliant with RoHS certification.

3.5 System Quality Attributes

3.5.1 Usability Requirements

- The system should be easy to set up, operate and maintain in order to be accessible to people with minimal technical knowledge.
- The software should be intuitive for learning basic gardening principles and experimentation for a good learning experience for students. Data visualizations and analysis components must be easy to understand and use since student's knowledge is minimal.
- The system should allow for precise control over planting parameters, data collection and export of data for research to be done.
- The interfaces should be accessible, working well with screen readers and feature adjustable font sizes to accommodate different users needs.
- The system should allow for automation of repetitive tasks so that researchers can focus on more important tasks.
- The user interface should be visually appealing, user-friendly, and must provide clear feedback on system status and actions.

- The system should offer a sense of accomplishment and provide a fun farming experience for home users to take part in as a family.

3.5.2 Performance Requirements

- FarmBot supports two user interfaces (terminals):
  1. FarmBot Web App (Browser)
  2. FarmBot Configurator (Raspberry Pi)
- Sensor data, camera data must be transferred between web-app and device in real-time. The messaging queue must process a million MQQT messages per second providing miliseconds-level of latency.

3.5.3 Reliability

- Transfer of sensor, camera data between FarmBot device and web-app shall be accurate, without loss and real-time.

3.5.4 Availability

- FarmBot should be available to execute custom sequences, events, and/or regimens 91.67 percent of the day. It performs updates between 3AM - 5AM.
- If the device attempts to connect to the message broker more than 20 times in a 10 minute period, it will be temporarily blocked for 10 minutes.

3.5.5 Security

- FarmBot device logs and web-app logs must be maintained to track user activity, system events and potential security breaches.
- Communication between critical software componenets are restricted via the use of tokens while using the APIs.

- Data validation and error checking mechanisms are implemented to ensure accuracy of user data and inputs.

3.5.6 Maintainability

- Modular design and well-defined interfaces between different components are crucial for independent development, testing and debugging.
- A clear and concise documentation is needed for future maintenance.
- A version control system must be utilized for tracking changes and rolling back if necessary.
- Unit tests and integration tests must be created to realize and fix problems efficiently.

3.5.7 Portability

- The web-app must allow using modern web standards to be compatible with different devices and browsers.

3.6 Supporting Information

FarmBot is an open-source and scalable automated precision farming machine. FarmBot automates repetitive tasks such as planting seeds, watering, and applying nutrients. It allows precise control over various planting parameters like seed spacing, watering schedules. It provides a data-driven approach to farming using various sensor data and crop information from OpenFarm. OpenFarm is a free and open database for farming and gardening knowledge. FarmBot can be a valuable tool for students and educators interested in learning about different STEM objectives. Robotics, agriculture and automation are some of them.

4. Suggestions to Improve The Existing System

4.1 System Perspective

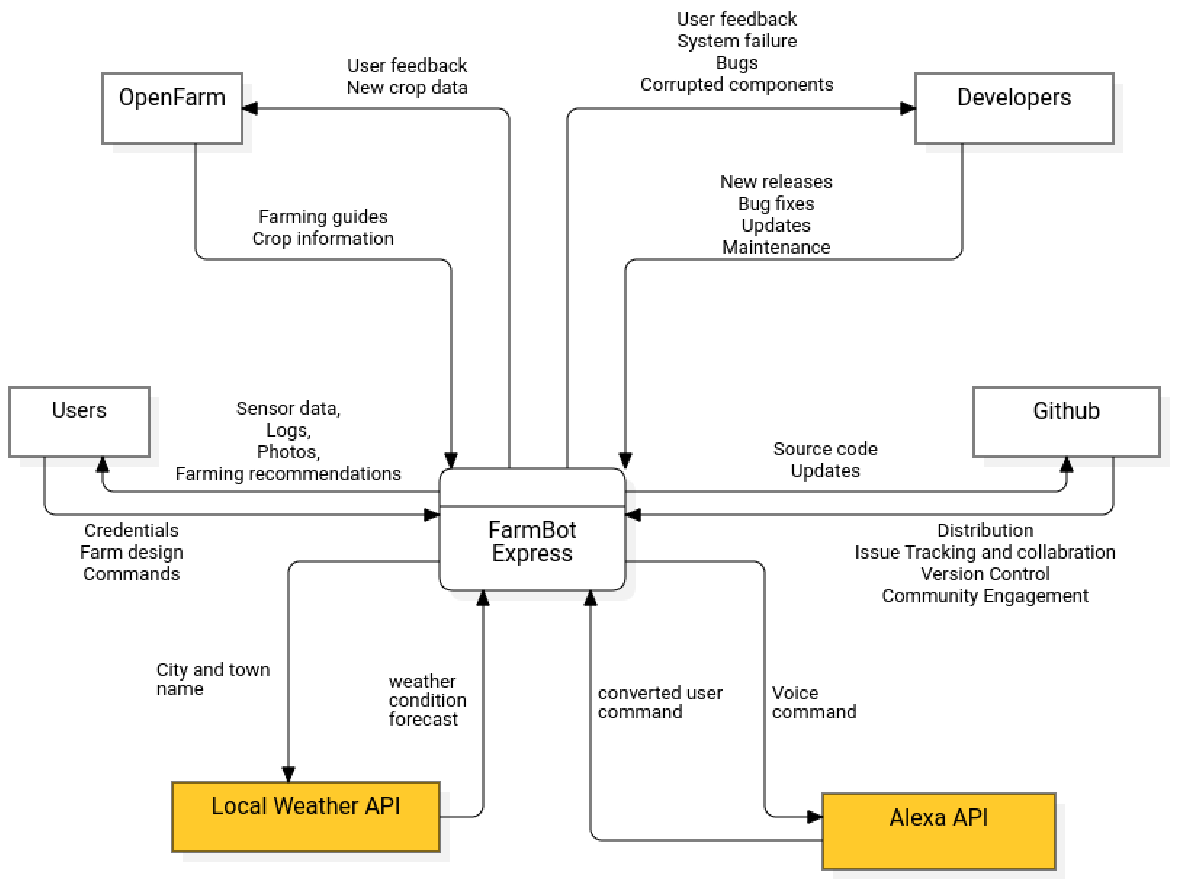


Figure 4.1: Improved Context Diagram

The improved context diagram is shown in Figure 4.1. It now has Local Weather API and Alexa API as new external entities. The system can get weather data from **Local Weather API** and add this data to its smart decision-making process. Also, the system can communicate with **Alexa API** to provide voice command support.

4.2 External Interfaces

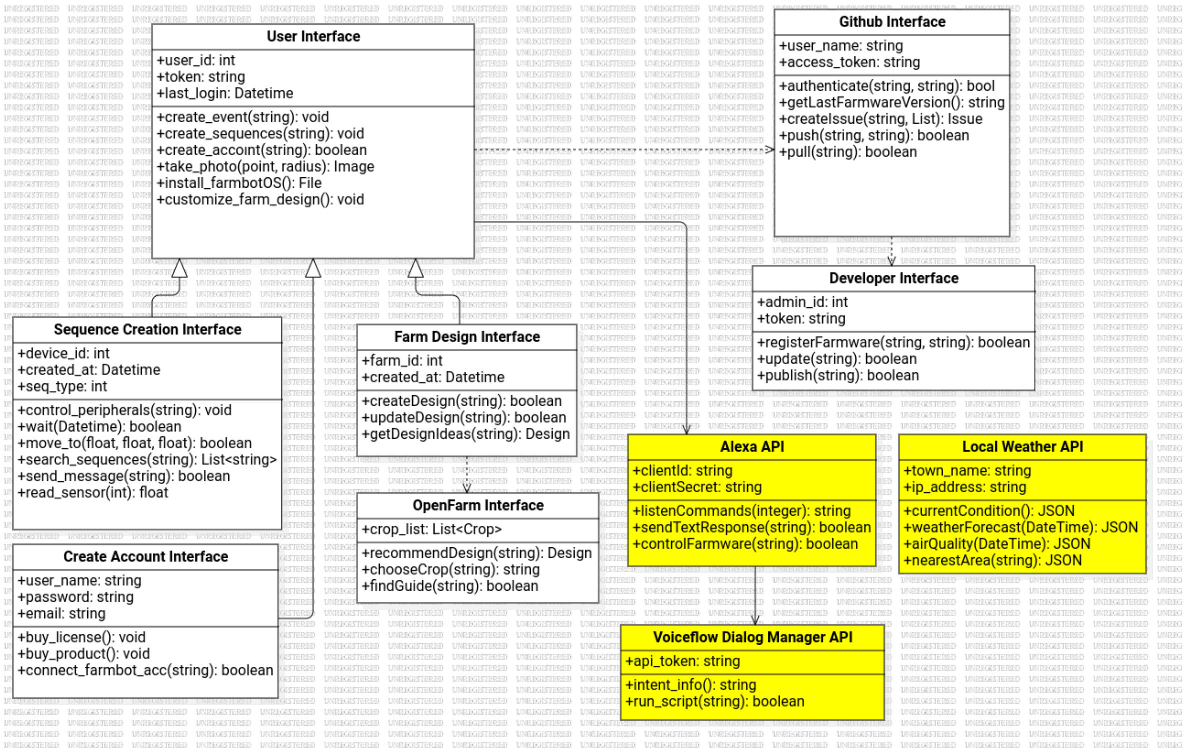

User Interface
+user_id: int
+token: string
+last_login: Datetime
+create_event(string): void
+create_sequences(string): void
+create_accoint(string): boolean
+take_photo(point, radius): Image
+install_farmbotOS(): File
+customize_farm_design(): void
Github Interface
+user_name: string
+access_token: string
+authenticate(string, string): bool
+getLastFarmwareVersion(): string
+createIssue(string, List): Issue
+push(string, string): boolean
+pull(string): boolean
Developer Interface
+admin_id: int
+token: string
+registerFarmware(string, string): boolean
+update(string): boolean
+publish(string): boolean
Sequence Creation Interface
+device_id: int
+created_at: Datetime
+seq_type: int
+control_peripherals(string): void
+wait(Datetime): boolean
+move_to(float, float, float): boolean
+search_sequences(string): List<string>
+send_message(string): boolean
+read_sensor(int): float
Farm Design Interface
+farm_id: int
+created_at: Datetime
+createDesign(string): boolean
+updateDesign(string): boolean
+getDesignIdeas(string): Design
OpenFarm Interface
+crop_list: List<Crop>
+recommendDesign(string): Design
+chooseCrop(string): string
+findGuide(string): boolean
Create Account Interface
+user_name: string
+password: string
+email: string
+buy_license(): void
+buy_product(): void
+connect_farmbot_acc(string): boolean
Alexa API
+clientId: string
+clientSecret: string
+listenCommands(integer): string
+sendTextResponse(string): boolean
+controlFarmware(string): boolean
Local Weather API
+town_name: string
+ip_address: string
+currentCondition(): JSON
+weatherForecast(DateTime): JSON
+airQuality(DateTime): JSON
+nearestArea(string): JSON
Voiceflow Dialog Manager API
+api_token: string
+intent_info(): string
+run_script(string): boolean

Figure 4.2: Improved External Interface Diagram

- **Local Weather API:** The system can get the current weather, temperature, humidity, and wind speed data from this API. The data can be used to make better decisions for the farm.
- **Alexa API:** The system can communicate with this Alexa API (provided by Amazon) to provide voice command support. The user can give voice commands to the system, and the system can respond to these commands. The vocal input

goes into the Alexa API and redirects to the Voiceflow API. The Voiceflow API processes the vocal input and sends the processed data to the system.

- **Voiceflow Dialog Manager API**: This newly introduced API gets its input from Alexa API and runs JavaScript code triggering the commands.

4.3 Functions

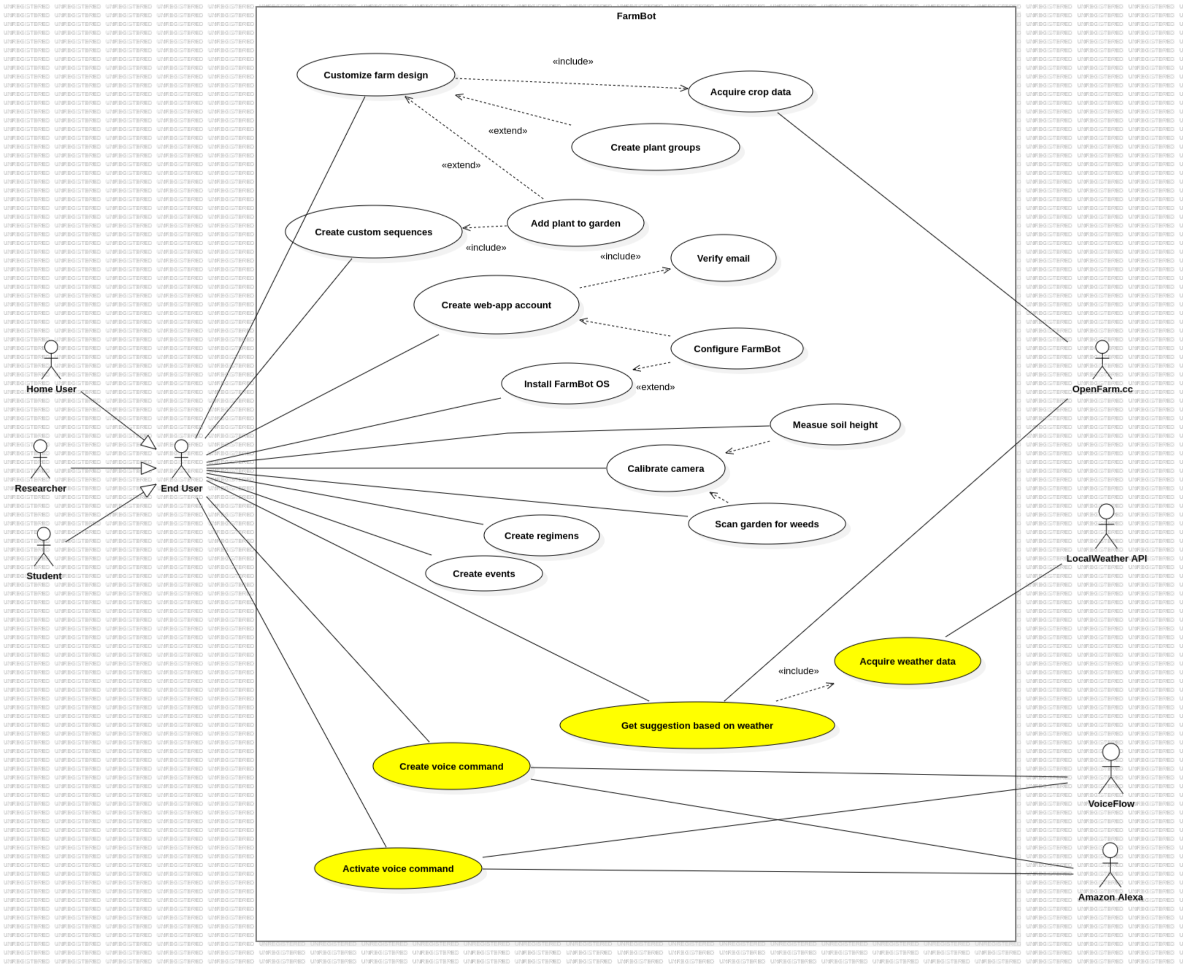


Figure 4.3: Improved Use Case Diagram

Use case name	Activate voice command
Actors	End User, VoiceFlow, Amazon Alexa
Description	End user talks with the virtual assistant Alexa integrated with VoiceFlow to run farming sequences
Preconditions	User has a Amazon Alexa device User is logged on the device User is able to talk User has created sequences on FarmBot web-app
Data	User voice, user FarmBot sequences
Response	FarmBot starts executing sequence
Stimulus	User says the word "Alexa" to wake device up
Normal Flow	1. User tells the sequence they want to execute e.g. "FarmBot, run watering sequence." 2. Alexa processes the command and sends it to VoiceFlow 3. VoiceFlow sends request to execute sequence to FarmBot web-app 4. FarmBot starts executing sequence
Alternative Flow	-
Exception Flow	3. If sequence does not exist, Alexa notifies the user and FarmBot does nothing.
Post Conditions	Sequence must have started executing
Comments	-

Table 4.1: Tabular description of Activate voice command use case

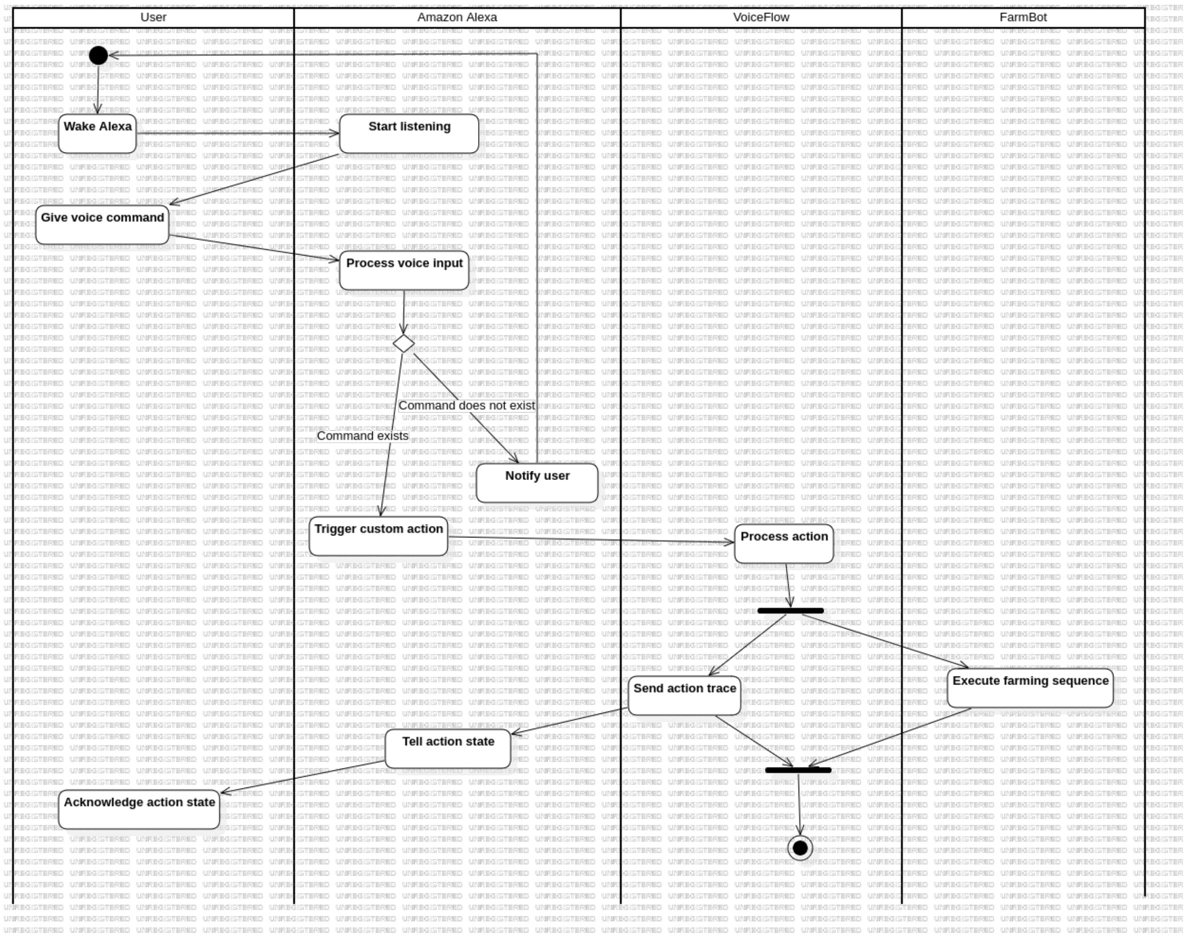


Figure 4.4: Activity Diagram for "Activate voice command" use case

Use case name	Create voice command
Actors	End User, VoiceFlow, Amazon Alexa
Description	User creates commands that execute farming sequences using Alexa integrated with VoiceFlow
Preconditions	User has an account on VoiceFlow User has an account on Alexa Developer platform
Data	NLU information (intents, entities)
Response	Amazon Alexa responds and executes the correct sequence
Stimulus	-
Normal Flow	1. User creates a VoiceFlow project 2. User creates intents within the project eg. ”water_intent” 3. User creates an Alexa Skill on Alexa Developer platform 4. User replicates the VoiceFlow NLU (Natural Language Understanding) in the Alexa skill
Alternative Flow	-
Exception Flow	-
Post Conditions	Alexa and VoiceFlow must be integrated and commands must be executing
Comments	By integrating Alexa with VoiceFlow users can activate voice commands in order to execute certain farming sequences

Table 4.2: Tabular description of Create voice command use case

Use case name	Acquire weather data
Actors	Local Weather API
Description	Weather data is acquired from Local Weather API including current weather, temperature, humidity, and wind speed data to use for farming action suggestions
Preconditions	FarmBot has city and town name configuration set FarmBot is authenticated to Weather API
Data	Current weather, temparature, humidity, wind speed
Response	Weather information is returned
Stimulus	-
Normal Flow	1. Web-app provides location data (city, town name) 2. Web-app provides number of days of forecast 3. Web-app makes a request to the Local Weather API 4. Weather information is returned
Alternative Flow	2. If number of days is not provided, the request is defaulted to monthly average
Exception Flow	1. If location data is not provided, request is rejected.
Post Conditions	Weather information must be available to the web-app
Comments	-

Table 4.3: Tabular description of Acquire weather data use case

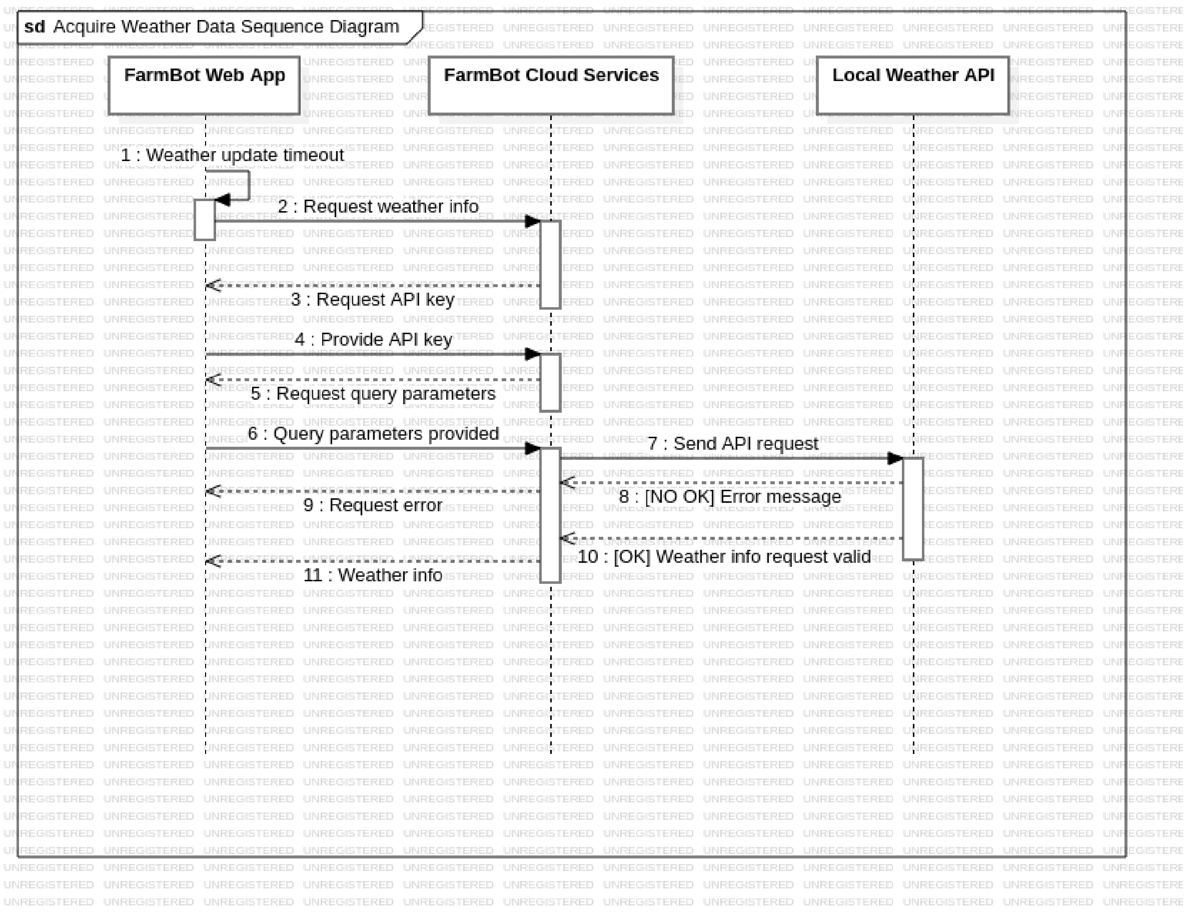

sd Acquire Weather Data Sequence Diagram
FarmBot Web App
FarmBot Cloud Services
Local Weather API
1 : Weather update timeout
2 : Request weather info
3 : Request API key
4 : Provide API key
5 : Request query parameters
6 : Query parameters provided
7 : Send API request
8 : [NO OK] Error message
9 : Request error
10 : [OK] Weather info request valid
11 : Weather info

Figure 4.5: Sequence Diagram for "Acquire weather data" use case

Use case name	Get suggestion based on weather
Actors	OpenFarm.cc, End User, Local Weather API
Description	Farming action suggestions are provided to the user based on the acquired weather data, and available crop information
Preconditions	Weather API is not down OpenFarm is not down User has authentication to Weather API User has authentication to OpenFarm
Data	Crop info, weather info
Response	Farming actions to take, sequences to execute are suggested to the user
Stimulus	Automatically run each day
Normal Flow	1. Web app checks current weather info 2. Web app checks currently planted crop information 3. Suggestions based on information are created 4. A notification for farming action suggestions are shown to the user on the web-app.
Alternative Flow	-
Exception Flow	1. If weather API cannot be reached, try again after a short interval 2. If crop information is non-existent, try to acquire crop info
Post Conditions	A notification must be displayed to the user with suggested farming actions.
Comments	-

Table 4.4: Tabular description of Get suggestion based on weather use case

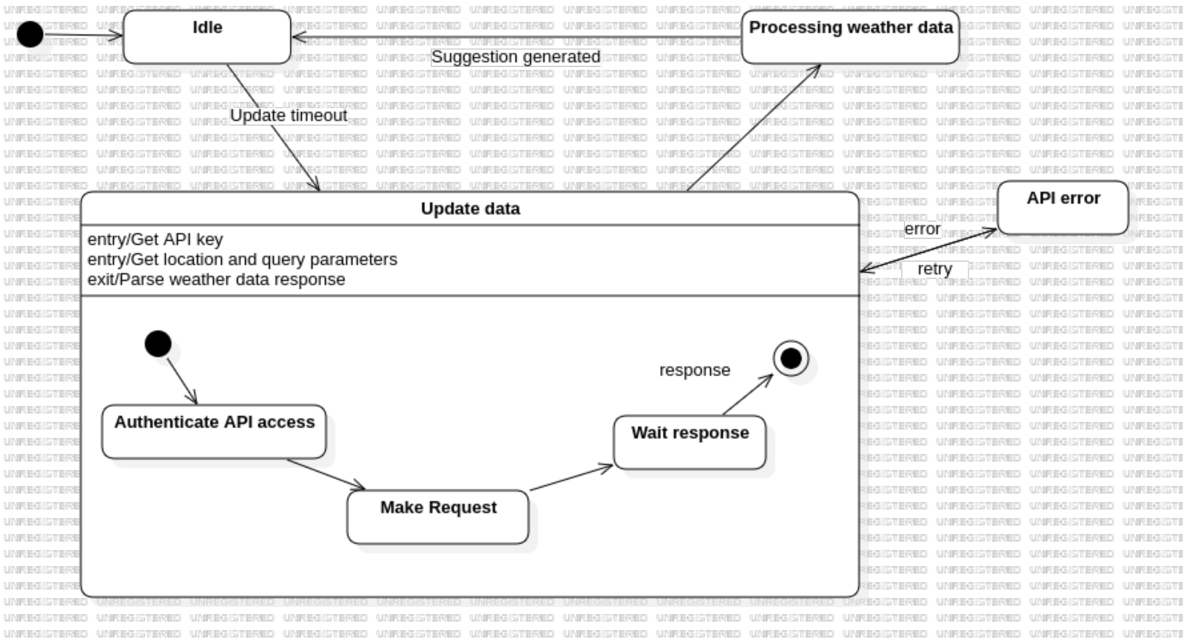


Figure 4.6: State Diagram for "Get suggestion based on weather" use case

4.4 Logical Database Requirements

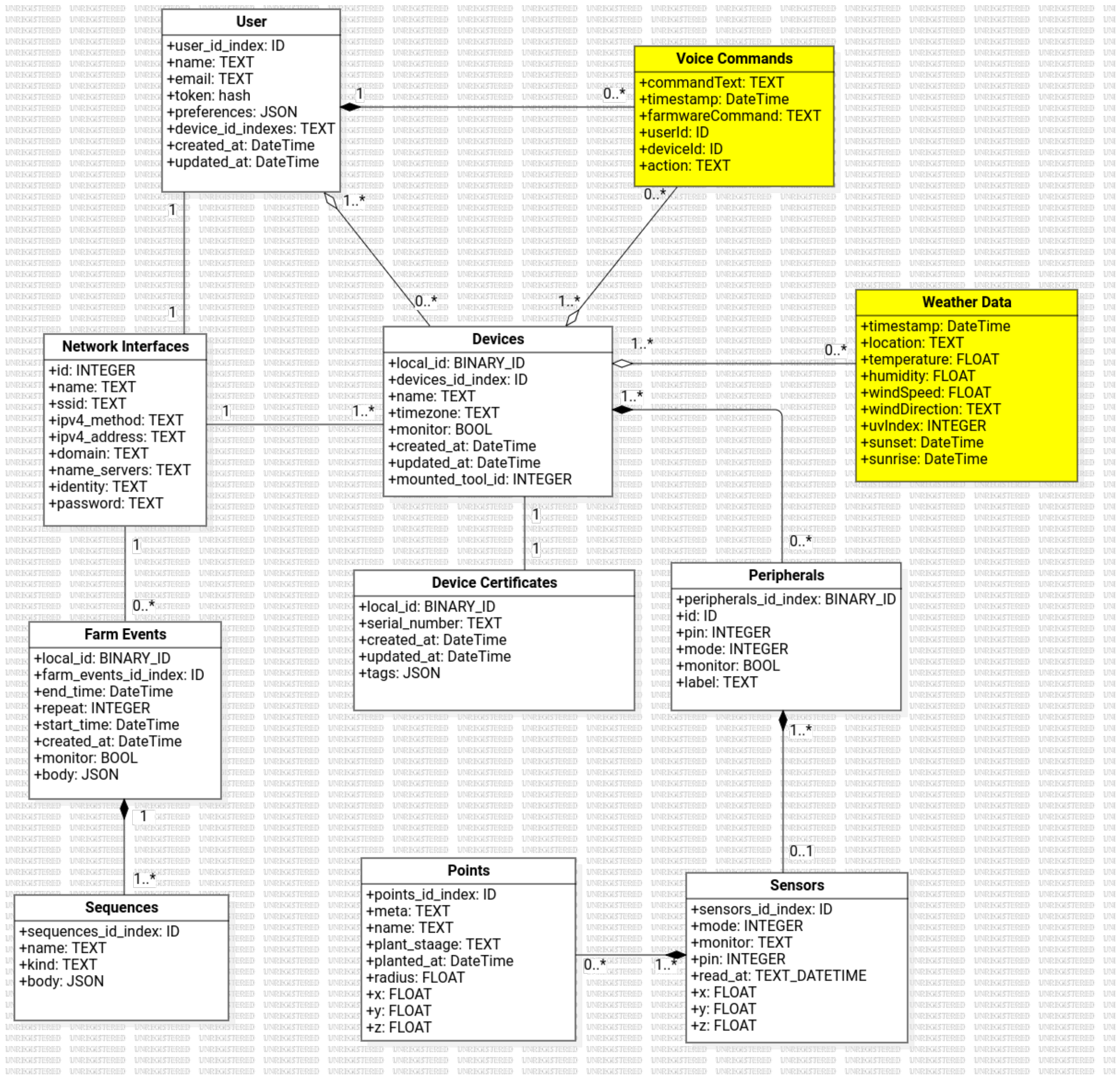


Figure 4.7: Improved Logical Database Requirements Diagram

The improved logical database requirements diagram is shown in Figure 4.7. The system now has a new table named **Weather Data** to store weather data. This table has columns for weather data such as temperature, humidity, and wind speed. The system can store the weather data obtained from the Local Weather API in this table. Any device that has access to the database can read this data and use it to make better decisions for the farm.

The system also has a new table named **Voice Commands** to store voice commands processed by the Voiceflow API. This table has columns for voice command data such as command text and action. The system can store the voice commands created by the user in this table. The system can read this data and execute the corresponding action when the user gives the voice command. A user may form their own voice commands and add them to the Voice Commands table and devices of this user can take action using the data stored in the Voice Commands table.

4.5 Design Constraints

- FarmBot must ensure that the voice data taken from end user must be processed and used in a compliant manner with the data privacy regulations GDPR (General Data Protection Regulation), CCPA (California Consumer Privacy Act).
- FarmBot must ensure that data privacy regulations GDPR and CCPA are not breached during the acquisition of weather data from the Local Weather API by using users location information.
- Access to reliable, real-time weather data requires subscription to a trusted weather service, API.
- Development resources and expertise in speech recognition technology might be limited for the FarmBot project.
- Secure communication protocol SSL/TLS must be used to ensure user data protection and prevention of breaches while connecting to Alexa, Voiceflow and Weather API.

4.6 System Quality Attributes

4.6.1 Usability Requirements

- Users should be able to successfully execute desired FarmBot actions using clear and concise voice commands.
- The voice recognition system should understand commands quickly and minimize the need for repeated attempts or corrections.
- The voice control interface should be intuitive and enjoyable to use, reducing reliance on manual controls.
- Farming action suggestions should be clear, actionable, and relevant to the user's specific farm context
- Weather data and suggestions should be presented concisely and easily understandable within the FarmBot interface.
- Users should find the suggestions helpful for making informed decisions about their farming actions.
- Voice control should support a variety of user accents and speaking styles.

4.6.2 Performance Requirements

- Average time to execute the sequence indicated by the voice command shall not exceed 5 seconds to ensure user satisfaction.
- User shall not need to have to repeat their voice command more than 2 times.
- Voice command interpretation accuracy must be at least 95
- Weather forecasts for the coming day must be acquired daily at 2AM-4AM (FarmBot update hours) with a delay of no more than 10 minutes.

4.6.3 Reliability

- Speech recognition must be accurate.
- Potential misinterpretations of voice commands must be handled gracefully, prompting for confirmation or offering alternative options must be implemented.
- Weather data source must have a high uptime and provide accurate weather information.

4.6.4 Availability

- Voice control functionality must be highly available, downtime due to recognition engine issues must be minimized.
- Despite weather data feed not being available, system should remain functional and give suggestions based on historical data or user preferences.

4.6.5 Security

- User voice data of any kind must be anonymized and opt-out option to users must be presented.
- Secure communication protocols such as HTTPS, SSL/TLS must be followed.
- FarmBot must not endanger its users due to misinterpreted voice commands

4.6.6 Maintainability

- Voice control and weather modules must be designed as seperate, well-documented modules to facilitate future improvements or integrations with different speech recognition engines.

4.6.7 Portability

- Speech recognition libraries, APIs used must be protable accros different platforms to maintain compatibility with potential hardware upgrades.
- Established libraries must be utilized for accessing and parsing weather data formats to ensure compatibility with various weather service APIs.

4.7 Supporting Information

In order to implement these suggestions the following choices are made:
Amazon Alexa is chosen as the hardware for capturing user voice and sending it to VoiceFlow for processing it. It supports various languages: English, German, Indian, Japanese, French, Italian, and Spanish ensuring accessibility for different cultures.
World Weather API's, "Local Weather API" is selected for the weather data source. It includes various metrics about the current weather at user's location such as temperature, humidity, and wind speed. The data is updated daily and it costs $120 monthly for a million requests per day to the API. JSON format is used for acquiring and parsing the weather data.