Department of Computer Engineering

CENG350 Software Engineering

Software Requirements Specification for FarmBot

Group 52

By

Emre Çam

Tufan Özkan

Friday 29th March, 2024

Contents

List of Figures

List of Tables

Revision History

Name	Date	Change Reason	Version
SRS First Draft	March 29, 2024	Initial Commit	1.0.0

Table 1: Change History

1. Introduction

This document provides the Software Specification Requirement (SRS) of an open-source precision agriculture CNC farming project that lets the user grow food from anywhere with a web app. The website for this system is https://farm.bot

1.1 Purpose of the System

The purpose of the FarmBot system is to revolutionize agricultural practices through automation. FarmBot aims to regularize farming methods, enhance crop yields and minimize resource waste. Additionally; FarmBot provide a platform to manage and monitor their crops and agricultural operations efficiently. Through automation and monitoring, the system aims to:

- Automate planting, watering, weeding and harvesting processes to reduce human work and improve efficiency.
- Provide precise control over environmental factors such as soil moisture, and harmful weeds.
- Facilitate real-time monitoring from web applications to enhance crop health.
- Provide an environment to design a layout for crops and implement that design in the real world.

Despite the many functionalities it offers, FarmBot's main purpose is not large-scale agriculture. The initiative to introduce the FarmBot technology aims to:

- Increase interest in gardening and plant cultivation.
- Teach people, especially students, to discover the needs of plants and how to observe those needs.
- Make a tool to teach robotic and computer programming skills.

1.2 Scope

In the scope of this system, Farmbot is humanity's open-source CNC farming machine that automatically grows food right in your backyard keeping you in complete control, as opposed to the current food production system which no longer has control over how the food is produced.

- Farmbot is provided with 95 pre-assembled elements to be set up quickly and easily and perform the basic functions needed to grow a garden.
- Raspberry Pi computers operate the system with its hardware components (webcam, Arduino firmware microcontroller combined with powerful stepper motors and dynamic devices, ph sensors) and user application requests.
- Farmduino microcontrollers allow positioning of the tool head with millimetre accuracy for sowing, and watering plants in any pattern and frequency with given built-in features of plants based on their type, age, and soil conditions. The sensors also send the condition of soil and weather data of pH, temperature, and moisture.
- The Onboard camera system allows user to monitor their garden, and capture images, and the farmbot can take action by detecting the weeds utilising advanced computer vision.
- The web and mobile application allows users to control the garden remotely, and configure or update the software of Raspberry Pi computers without any need for hardware change.

- The farm designer and sequence editors allow users to lay out their plants with built-in data for plants, creating optimal layouts each season, and taking care of their plants in the way they want without requiring any coding.
- The farmbot software development allows professionals to modify and use it for any purpose (e.g. education, business) in hardware tools or develop completely new versions of the farmbot based on the existing library of components.

1.3 System Overview

This section of the document will provide detailed information about the system including all components.

1.3.1 System Perspective

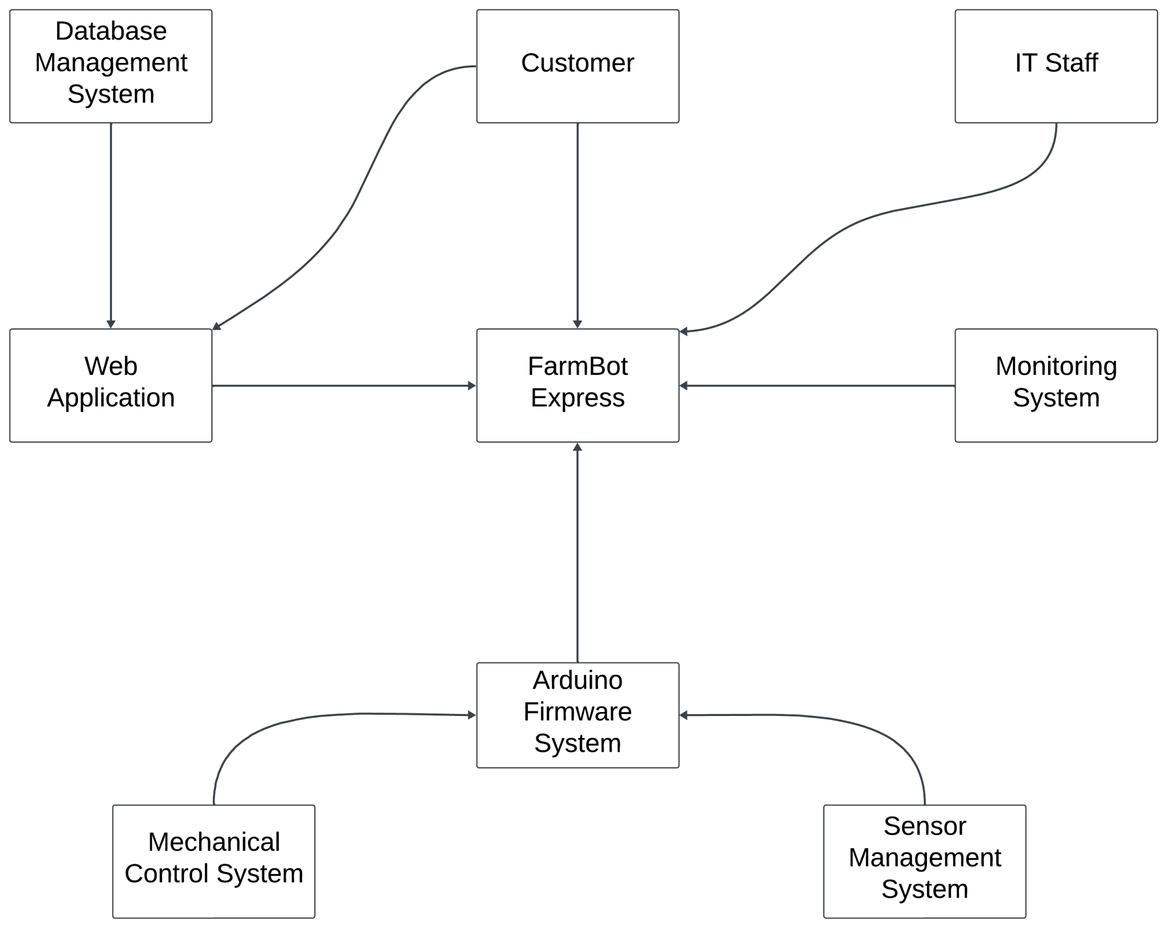


Figure 1.1: Context Diagram of FarmBot Express

FarmBot is not part of a large system nor necessarily requires the download of an application. Hence customers shall go into the web browser of the app and configure their FarmBot using their laptop, tablet, or smartphone. The web application features real-time manual controls and logging, a sequence builder for custom routines for FarmBot to execute.The web application keeps the data and receives the built-in data for plants from the database management system. The machine maintains a connection and synchronises with the web application via the message broker. Besides that, Farmbot communicates with the Arduino firmware system over a serial connection to send commands for motor and tool positioning and receive collected from sensors and rotary encoders. Moreover, the FarmBot is connected to a monitoring system through a webcam to provide real-time view, image-capturing, and detecting weeds, taking actions according to it. Furthermore, the system provides an IT staff interface in case of loggings, errors and customary configurations to help customers.

1.3.1.1 System Interfaces

- **Monitoring System:** In the FarmBot Express, the camera is provided within the kit. The kit is connected to Raspberry Pi through a USB cable or serial connection. The monitoring system receives data from the camera and using built-in computer vision functions, it detects the weeds and layout of the garden in general. According to the status of the garden, it takes actions autonomously or alerts the customer through the web application using the MQTT Gateway service. Moreover, it provides a real-time view or image-capturing from recorded data.

- **Arduino Firmware System:** The system consists of two subsystems named Mechanical Control System and Sensor Management System. The firmware that is flashed onto the Arduino or Farmduino microcontroller is responsible for physically operating FarmBot's motors, tools, sensors, and other electronics. It receives G and F codes from FarmBot OS, and then moves the motors and reads and writes pins accordingly. It also sends collected data from the rotary encoders and pin reads back to the Raspberry Pi. The processed data returns as a command from the main system to position tool heads to set light, sow weeds, and water plants.

- **Mechanical Control System:** This system is primarily constructed from V-slot aluminium extrusions and aluminium plates and brackets. They are driven by four NEMA 17 stepper motors, the Farmduino Express microcontroller, and a Raspberry Pi Zero 2 W computer. The bots roll directly on the wood-raised bed. FarmBot Express features a three-in-one tool head and does not require any additional precision tool. The gantry allows the tool head to be moved in the X-direction and serves as a guide for the Y and Z directions. The gantry also supports the structure for mounting the electronics box and seed troughs.

- **Sensor Management System:** This system has soil moisture, temperature, light, pH, humidity, and weather sensors. The abrupt measurements are sent to the Farmduino Express microcontroller. The firmware and Raspberry Pi computers process data and take actions autonomously or if it requires further investigation, it sends messages to the web server.

- **Web Application:** This system allows users to control and configure their FarmBot from most electronic devices. It does not require installation since it is a web-based application. The core features include the farm designer, event scheduler, sequence editor, regimen editor, controls, management of FarmBot's tools and seed containers, management of FarmBot's settings, storage and viewing of logs and sensor readings. Moreover, the application cloud is connected to the crop database.

- **Database Management System:** This system serves as a comprehensive repository of information about botanics and personal gardens. The general built-in data details are planting information, growing requirements, watering needs, nutritional needs, companion planting, pest and disease management, and harvesting and maintenance. Overall, the crop database is a valuable resource for users, so that they do not have to search for plants or to be experts.

1.3.1.2 User Interfaces

FarmBot Express users and IT staff interact with the system primarily through a web-based application accessible via computer and mobile device browsers. The interface facilitates various functionalities essential for managing and monitoring farm operations. Furthermore; embedded operating system of the FarmBot Express is also available as open-source code. So a customer or an IT member can access the operating system code and configure the embedded system of the FarmBot Express by using means of embedded microcontroller chip. The following parts constitute the user interface:

- **Customer Interface:** Users are required to create an account by providing necessary details such as name, email address, and password. After signing, users can log in using their credentials to access their accounts. Upon logging in, users are greeted with a dashboard displaying key information such as current garden layout, planted crops and real photos of the layout. The dashboard provides tabs of the farming operations that allow users to navigate to different sections of the application. These tabs can be grouped as below:

– **Layout Screen:** Shows the layout of the garden. It shows real photos of the garden, harmful weeds, the spread area of the planted crops, condition of the soil etc. directly on the garden layout.

– **Plants:** In this tab, user can view and manage all of the plants in the garden. Plants can be dragged and dropped to the garden layout to make a design.

– **Weeds:** This tab stands for viewing and managing the weeds in the garden. Weeds that are detected by FarmBot Express appear in this tab. User can select which weeds to be removed and what is the period of the removal.

– **Points:** This tab provides information about soil in the different point of the garden. User can select points to monitor soil condition.

– **Curves:** This tab provides information about water, spread and height curves for the plants. User can determine amount of water, spread and height of the plants in this tab.

– **Sequences:** In this tab, user can configure sequences by putting some steps in order like first plant the seed, water it and then take a photo of it.

– **Regimens:** This tab stands for scheduling sequences based on the age of the plants.

– **Events:** This tab provides an overview of schedulend sequences and regimens for the user.

– **Sensors:** This tab provides information about historical sensor readings. It also allows user to view current sensor readings and manage sensors.

  - **Photos:** In this tab, user can view and manage photos of the garden taken by FarmBot Express. It also allows user to calibrate camera and weed detection settings of the FarmBot Express.
  - **Tools:** This tab allows user to view detailed information about and manage tools, seed containers, and slots in the garden.
  - **Messages:** This tab includes messages and announcements for the user.
  - **Help:** In this tab, user can see the documentation of the web-application and can get support from IT members.
  - **Settings:** This tab is the settings tab for web-application. Mainly for configuring accounts of the user.
  - **Contols:** This tab provides a controller interface for user to control movements of FarmBot Express manually.
  - **Jobs and Logs:** In this tab, user can view running and completed jobs.
  - **Connectivity:** This tab provides information about the connection between the web browser, FarmBot Express, and the FarmBot web-application servers.

  Besides this web-application user interface, the web-application and the embedded operating system of the FarmBot Express are open-source. So advanced users can configure the application and operating system as they want. Last thing to mention is that, FarmBot provides a REST API to edit information such as sequences and the gardan layout without using web-application directly.

- **IT Staff Interface:** While there isn't an interface exclusively designed for IT staff within the FarmBot Express system, IT members have access to the source code of both the web-application and the embedded operating system of the FarmBot Express as they are open source. This access enables them to support users and perform configurations as needed.

1.3.1.3 Hardware Interfaces

The system requires a borescope camera, Raspberry Pi computer, Farmduino, several sensors, four motors, rotary encoders, Vacuum Pump, audio jack, micro SD Card, Wi-Fi radio antenna, Ethernet connector, several USB cables, UTM cable, power supply, LED light strip, water solenoid valve and the universal tool mount. The camera supports advanced computer vision technologies. The Raspberry Pi computer, Farmduino and many mechanical devices are usually connected through USB cables. To receive data and send commands, the vacuum pump, water solenoid valve, encoders, motors, and light strip are integrated into the Raspberry Pi. Another hardware requirement is an electronic device that has a connection to the web server. Hence the Ethernet connector or the Wi-fi radio antenna must be implemented into the Farmduino firmware. Overall, the Raspberry Pi serves as a communication node between the Web App and the Farmduino via message broker, MQTT gateway. This communication allows Raspberry Pi to get instructions, customary configurations, and modifications manually from the web app to the Farmduino and hence to other hardware products.

1.3.1.4 Software Interfaces

- **Database:** OpenFarm which is a free and open source database for farming and gardening knowledge is used for this service. This service stores and provides crop and growing information to the web app for a streamlined user experienced. Any information in the tabs mentioned in the User Interface stored in the database. This information includes data accumulated by the sensors of the FarmBot Express such as moisture of the soil, temperature; data obtained from the photos provided by the webcam of the FarmBot Express such as weeds in the garden; data related to planned garden layout; data of jobs and scheduled sequences and much more.
  - Name: OpenFarm
  - Version Number: Last stable version

  – Source: https://openfarm.cc/

1.3.1.5 Communication Interfaces

The communication protocol between FarmBot Express and the web application is established using the MQTT Gateway protocol. MQTT is a lightweight messaging protocol ideal for communication between constrained devices and network servers. MQTT operates on a publish-subscribe messaging pattern, allowing FarmBot Express and the web application to exchange messages asynchronously. Messages are published by either FarmBot Express or the web application and subscribed to by the respective counterpart to facilitate bidirectional communication. The message content includes commands, sensor readings, job logs, data obtained by the hardware and photos taken by the webcam.

1.3.1.6 Memory Constraints

The system's application is web-based, therefore there is no requirement to have storage in the user's device. On the other hand, the Farmbot Express is a microcontroller after all, hence it has built-in preserved area storage for program and data memories. Hence there is a memory constraint to keep webcam images, recordings, and data of several sensors. The data should be sent to web cloud services before overwriting. Moreover, it is an open-source system, hence the functions, OS and more can be configured by customers which affects the preserved program memory.

1.3.1.7 Operations

The operations of the Farmbot Express system can be partitioned into:

User-Initiated Operations:

- Customising garden
- Planting the garden
- Mounting and dismounting tools

- Measuring soil moisture
- Performing actions on many plants
- Using Farmbot's buttons
- Scanning the garden for weeds
- Using the rotary tool

Periods of Interactive Operations and Periods of Unattended Operations:

- Optimised Events
- Automated Planting Schedule
- Scheduled Sowing
- Scheduled Watering
- Real-time Viewing
- Capturing Images
- Connecting Farmbot to the Internet
- Connecting Farmbot to the Web App

Data Processing Support Functions

- Sensor Data Analysis
- Image Processing
- Logging
- Rotary Encoder Data Analysis
- Wi-fi and Web App credentials configuration

1.3.2 System Functions

Connection of the FarmBot to the Web App	The system establishes a communication with the user through an application to send and receive messages, allows the user to control remotely, schedule tasks and app to analyse and store data in the crop database. .
Axis Setup	The system sets the axis up and home position to generate a map on the web app interface. Hence, the user can use the map to achieve various tasks such as planting, watering, sowing, and harvesting. Moreover, the system itself recognises the garden and can give commands to Arduino firmware correctly.
Mounting Tool	The system selects the correct tool out of the tool slot and mounts it to succeed in various tasks with the correct equipment.
Dismounting Tool	The system dismounts the tool from the tool head to mount new tools since it has to manage different sequence commands with different tools.
Planting Weed	The system mounts the needle and turns the vacuum pump on to suck and held the seed. In the suction-held position, the system moves the tool head to the correct position utilising stepper drivers. The system lets the seed off.

Dispensing Water	The system analyses the plants concerning their species, age and many various factors. Then, it sends commands to the firmware to distribute and water plants accordingly.
Measuring Soil Moisture	The system checks the soil to ensure that its environment is optimal for the plants. The system utilises the soil moisture sensor and then, it notifies the user or takes initiative and applies essential commands.
Scanning the Garden for the Weeds	The system checks the garden and scans it with the webcam to have information about the garden and notify the user with relevant image data. Utilising advanced computer vision technologies, the system takes actions as it sees fit if necessary or applies user-generated commands.
Mowing all weeds	The system mows all weeds respectively after mounting the rotary tool.
Emergency Locking and Unlocking	When the system encounters issues such as motor stalling unexpectedly or due to an event that triggers the error mechanism, the system shuts down the firmware and its connected motors and sensors.

1.3.3 Stakeholder Characteristics

- **Botanic Students**
  - **Objective**: Utilize FarmBot Express as an educational tool to learn about

botany.

  - **Characteristics**:
    - Interested in studying plant biology, growth patterns, and environmental interactions.
    - Seek practical applications to extend understanding of botany.
    - May have varying levels of prior knowledge in botany.

- **Engineering Students**
  - **Objective**: Utilize FarmBot Express as an educational tool to learn about robotics and coding.
  - **Characteristics**:
    - Seek hands-on learning experiences to deepen understanding of robotics and coding.
    - May have varying levels of prior knowledge in robotics and computer science.
- **Scientists**
  - **Objective**: Utilize FarmBot Express for conducting experiments with plants and conducting research.
  - **Characteristics**:
    - Engaged in scientific research related to plant biology and environmental science.
    - Require precise control over experimental conditions, including watering, light exposure, and soil nutrients.
    - May have advanced knowledge of botany, molecular biology, or related fields.
- **Garden Enthusiasts**

  - **Objective**: Enhance gardening skills and optimize plant growth using automated techniques.
  - **Characteristics**:
    * Passionate about gardening and plant cultivation.
    * Desire to leverage technology for improving crop yields and efficiency.
    * May have varying levels of experience in traditional gardening practices.

- **Educators**
  - **Objective**: Integrate FarmBot Express into educational purposes to teach agricultural concepts and engineering skills.
  - **Characteristics**:
    * Professionals in the field of education with a focus on robotics or agricultural instruction.
    * Seek innovative teaching tools to engage students and enhance learning outcomes.

- **Home Gardeners**
  - **Objective**: Simplify and automate gardening tasks for hobby or sustenance gardening.
  - **Characteristics**:
    * Enjoy gardening as a leisure activity or means of producing food.
    * Value convenience and time-saving solutions for maintaining garden plots.
    * May have experience in traditional gardening methods and programming.

1.3.4 Limitations

- **Regulatory Policies:** The system's web app holds personal information, like name, email address, telephone number or credit card to store with the account.

Moreover, it collects all FarmBot web application data (e.g. photos, garden layout, weeds), device, log, location pieces of information, and local storage. Hence the system must hold data as encrypted.

- **Hardware Limitations:** The system is kept all day and night outside with the rain, sun, bugs, birds, dirt and different temperatures. Hence, it is suggested to inspect the hardware for loose screws, visible damage such as corrosion, bending, or cracking of components every three months. These are important to receive and process the correct image, sensor, and encoder data to send to the cloud and apply built-in or customary instructions.
- **Interfaces to other applications:** FarmBot Express must be compatible with web app services, Arduino firmware system, and monitoring system. Furthermore, the operating system and version of the customer's device must be compatible with FarmBot Express.
- **Parallel Operation:** FarmBot Express is a standalone system controlled by a web app registered by a particular account. Hence parallel operation is not required.
- **Audit Functions:** The system is an open-source, hence it can be configured by IT staff or customers. However, the inventory of weeds, water and the maintenance of the Arduino firmware system and the system's hardware shall be checked manually in person.
- **Control Functions:** The system has built-in advanced control-automation technologies. However, it is an open-source project and the user can customise and configure the system freely.
- **Higher-order Language Requirements:** The system and its Farmduino system are written in microprocessor-specific assembly. The system's firmware can be configured with C++, and its web app is written with Ruby on Rails, ReactJS, and TypeScript.

- **Signal Handshake Protocols:** The communication protocol between the FarmBot Express and its web services is satisfied by MQTT Gateway.
- **Quality Requirements:** The maintenance of the hardware, robots, bed, and sensors shall be checked physically regularly.
- **Criticality of the application:** The system is a single unit used by specific people. Hence the failure of the machine affects only one user.
- **Safety and security considerations:** The system's web app stores personal data. Hence the IT Staff shall check security regularly.
- **Physical/Mental considerations:** The system's web application does not require any coding and is easy to use. The built-in crop database provides information about Botanics and there is no physical action required after assembling the FarmBot Express. Hence physically disabled, young or old, expert or not expert, anyone can use it.

1.4 Definitions

You should add acronyms and abbreviations here. Refer to *(Clause 9.6.7)*

For any citation, refer to it as [1].

2. References

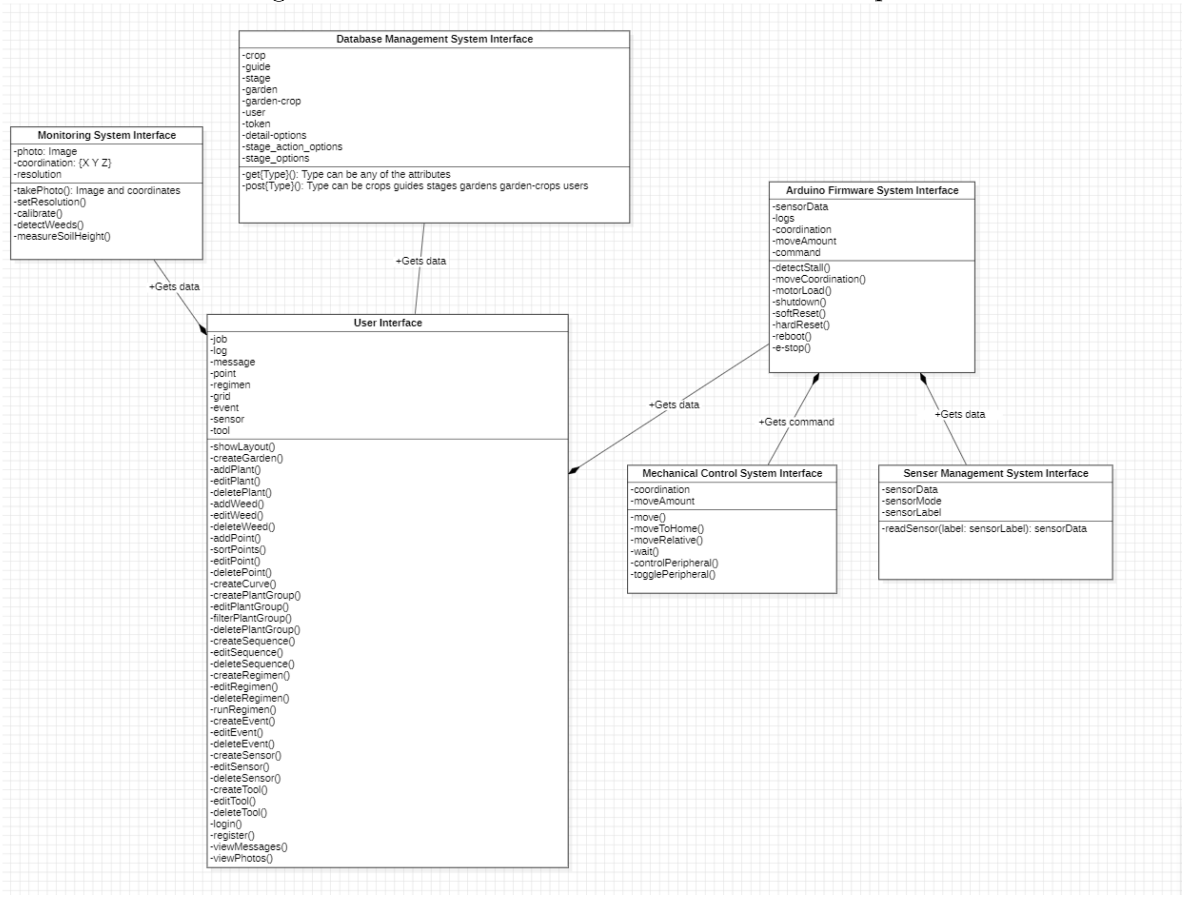
[1] M. C. Younis and H. Abuhammad, “A hybrid fusion framework to multi-modal biometric identification,” *Multimedia Tools and Applications*, vol. 80, no. 17, pp. 25799–25822, 2021.

3. Specific Requirements

3.1 External Interfaces

The following class diagram represents the relationship between interfaces and their functionalities

Figure 3.1: External interfaces of FarmBot Express



3.2 Functions

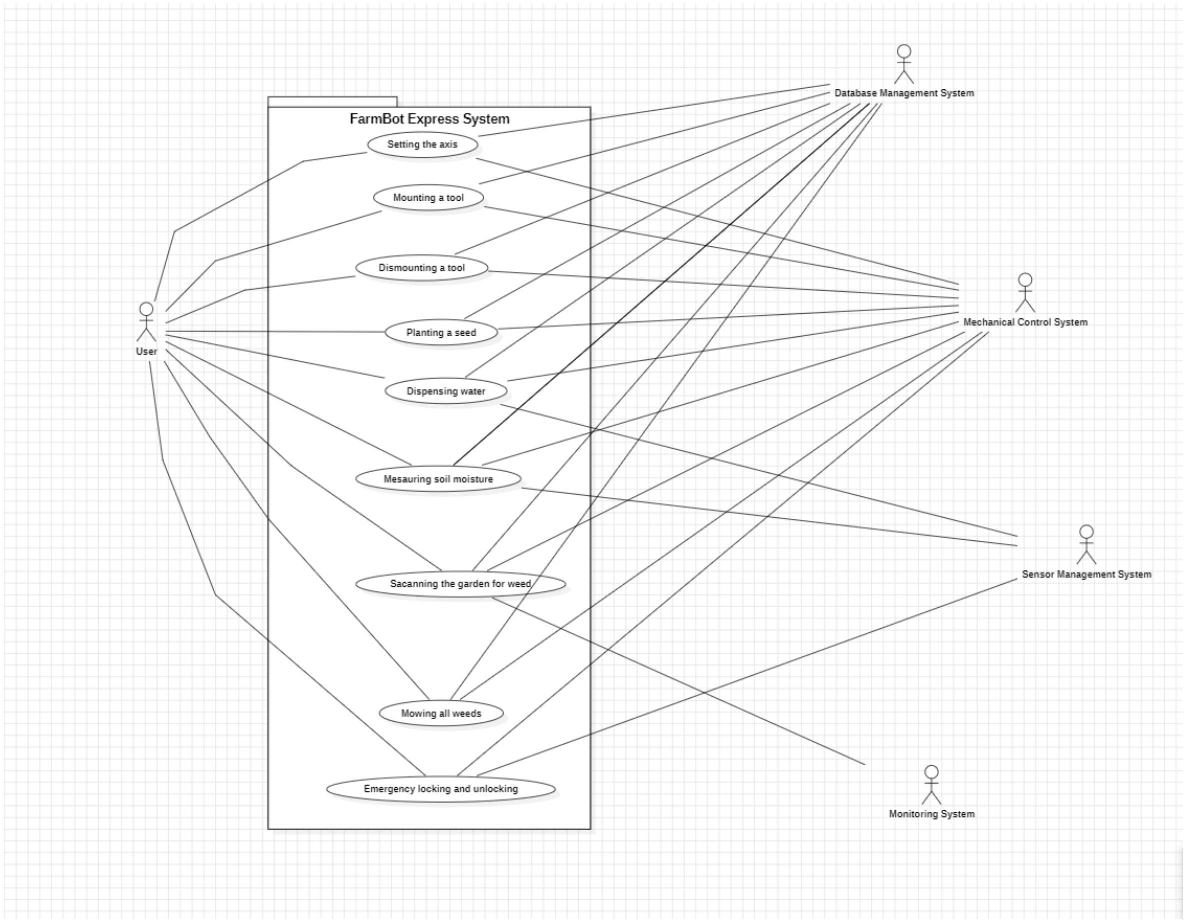


Figure 3.2: Use Case Diagram

Use-Case Name	Mounting tool
Actors	User, Web Application
Description	The function will mount the chosen tool to the FarmBot Express UTM.
Data	Selected tool's verification code, coordinates of the tool slot
Preconditions	Axis setup of the garden, tool slot, and home position must be completed.

Stimulus	User presses the mount tool option from the sequence panel and selects a tool.
Basic Flow	*Step 1:* Web application sends a command for the system to check a tool is not already mounted. *Step 2:* The system sends codes to Arduino Firmware to send direction and step pulses. *Step 3:* The mechanical system motors retract the Z axis to zero and move the X and Y axes above the tool. Then, it lowers the Z axis to mount the tool. *Step 4:* The mechanical system pulls the tool out of the slot. *Step 5:* The switches provide the tool verification pin to check the tool is correctly mounted. *Step 6:* After system verification, the system sends the data to the Web Application. *Step 7:* Web application updates the mounted tool field shown in the Tools Panel.
Alternative Flow#1	*Step 3:* Arduino firmware reads the relevant pins as on and sends a signal to the FarmBot. *Step 4:* FarmBot sends a message to the web app stating that there is a tool mounted already.
Alternative Flow#2	*Step 5:* The tool verification pin is off and does not match. Farmduino raises a signal to the FarmBot. *Step 6:* FarmBot sends a message to the web app stating that the tool is not found.
Exception Flow	*Step 3:* The system detects motors stalling. FarmBot raises an error to the web app interface via a message broker.
Post Conditions	Tool head is positioned to home with the tool.

Table 3.1: Mounting a Tool

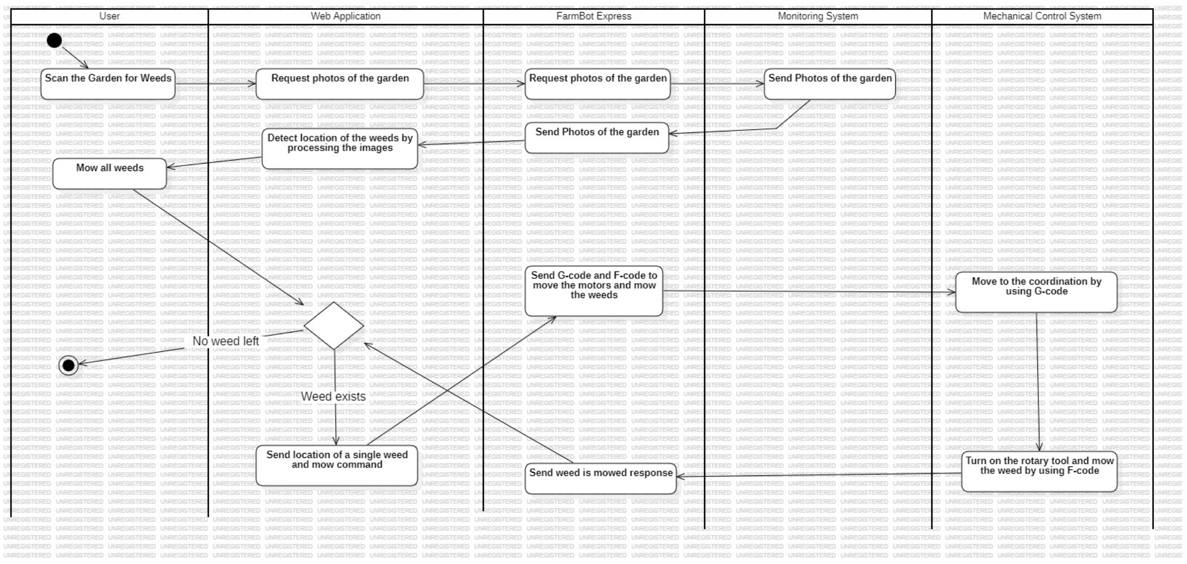

User
Web Application
FarmBot Express
Monitoring System
Mechanical Control System
Scan the Garden for Weeds
Request photos of the garden
Request photos of the garden
Send Photos of the garden
Send Photos of the garden
Detect location of the weeds by processing the images
Mow all weeds
Send G-code and F-code to move the motors and mow the weeds
Move to the coordination by using G-code
No weed left
Weed exists
Send location of a single weed and mow command
Send weed is mowed response
Turn on the rotary tool and mow the weed by using F-code

Figure 3.3: Activity Diagram of Mowing All Weeds

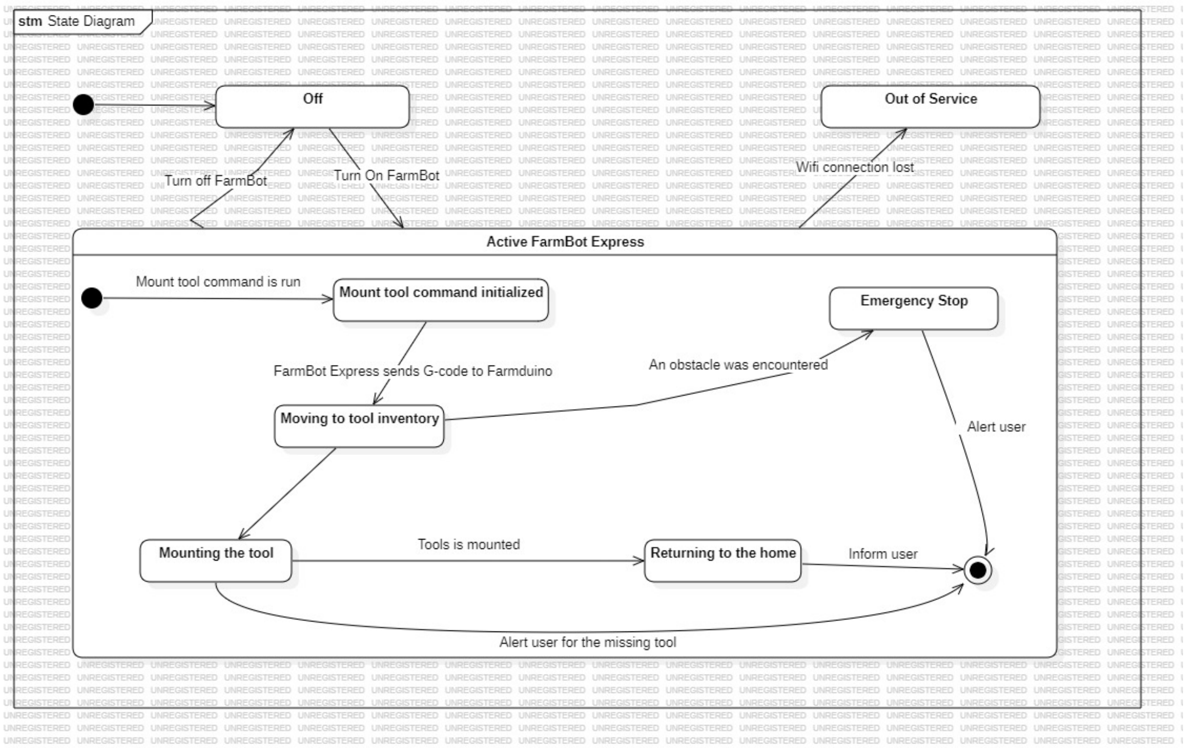


Figure 3.4: State Diagram of Mounting a Tool

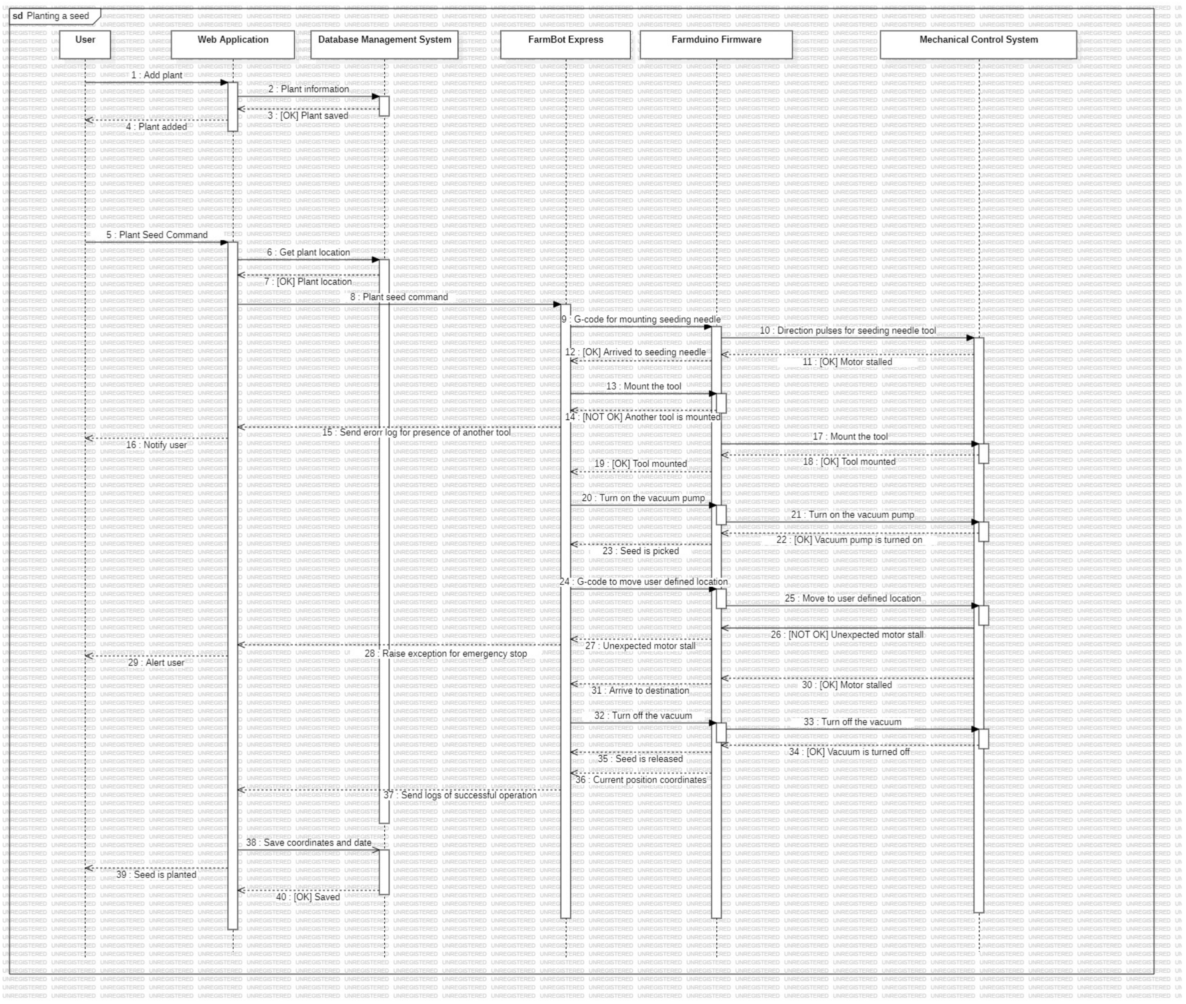

sd Planting a seed
User
Web Application
Database Management System
FarmBot Express
Farmduino Firmware
Mechanical Control System
1 : Add plant
2 : Plant information
3 : [OK] Plant saved
4 : Plant added
5 : Plant Seed Command
6 : Get plant location
7 : [OK] Plant location
8 : Plant seed command
9 : G-code for mounting seeding needle
10 : Direction pulses for seeding needle tool
11 : [OK] Motor stalled
12 : [OK] Arrived to seeding needle
13 : Mount the tool
14 : [NOT OK] Another tool is mounted
15 : Send erorr log for presence of another tool
16 : Notify user
17 : Mount the tool
18 : [OK] Tool mounted
19 : [OK] Tool mounted
20 : Turn on the vacuum pump
21 : Turn on the vacuum pump
22 : [OK] Vacuum pump is turned on
23 : Seed is picked
24 : G-code to move user defined location
25 : Move to user defined location
26 : [NOT OK] Unexpected motor stall
27 : Unexpected motor stall
28 : Raise exception for emergency stop
29 : Alert user
30 : [OK] Motor stalled
31 : Arrive to destination
32 : Turn off the vacuum
33 : Turn off the vacuum
34 : [OK] Vacuum is turned off
35 : Seed is released
36 : Current position coordinates
37 : Send logs of successful operation
38 : Save coordinates and date
39 : Seed is planted
40 : [OK] Saved

Figure 3.5: Sequence Diagram of Sowing a Seed

Use-Case Name	Dismounting tool
Actors	User, Web Application
Description	The function will check for the presence of a tool mounted to the FarmBot Express and dismount the tool in the slot where it belongs.
Data	Tool verification pin, coordinates of the tool slot
Preconditions	Axis setup of the garden, tool slot, and home position must be completed.

Stimulus	User presses the dismount tool option from the sequence panel.
Basic Flow	*Step 1:* Web application sends a command for the system to check for the presence of a tool. *Step 2:* The system reads the tool verification pin data. *Step 3:* System checks the mounted tool to identify which tool is mounted and where it belongs. *Step 4:* The system sends coordinate codes to Arduino Firmware for navigation. *Step 5:* The mechanical system retracts the Z axis to 0(zero) and moves the X and Y axes above the front of the tool slot. Then, it lowers the Z axis to the height of the tool slot, puts the tool back in the slot, and dismounts it. *Step 6:* The Farmduino verifies that the tool has been dismounted and sets the tool data pin accordingly. *Step 7:* After receiving the pin data, the system sends the data to the Web Application. *Step 8:* Web application updates the mounted tool field accordingly.
Alternative Flow#1	*Step 2:* The requested tool is already mounted.
Alternative Flow#2	*Step 2:* There exists another tool mounted. *Step 3:* The Web application raises an exception stating that there is a mounted tool already.
Exception Flow	*Step 1:* The connection is lost during the request.
Post Conditions	-

Table 3.2: Dismounting Tool

Use-Case Name	Measuring Soil Moisture
Actors	User, Web Application
Description	This function measures soil moisture and then uses that data to dose more or less water.
Data	coordinates of the area to measure soil moisture
Preconditions	Soil moisture sensor, correct wired-up UTM and sensors, the availability of the sensor in Sensors Panel in Web Application
Stimulus	User chooses measure soil moisture from the sequence panel and selects variables and activies.
Basic Flow	*Step 1:* User adds the soil moisture sensor. *Step 2:* User builds two simple watering sequences and types the grid's location to be measured. *Step 3:* FarmBot sends commands to the Farmduino to mount the watering nozzle. *Step 4:* After attaching the tool, the firmware sends signals to the motors to move to the user-defined position and waters the grid until an interrupt comes from the FarmBot. The Farmbot makes it wait and water the garden again with default or user-defined time variables. *Step 5:* After watering, Farmbot sends a command to Firmware again to write to the pins of the soil moisture sensor to turn on. *Step 6:* Firmware reads the pins of the sensor data and conveys it to the FarmBot. *Step 7:* Farmbot directly uploads the read moisture data to the web services. *Step 8:* Web application updates the panel with new logs.

Alternative Flow#1	*Step 4:* There is already a mounted tool. FarmBot notifies the user via the web app interface.
Alternative Flow#2	*Step 6:* The data cannot be read, the sensor needs to be calibrated or renewed. *Step 7:* FarmBot sends a message stating that the soil moisture is not measured.
Exception Flow	*Step 1:* The connection is lost during the sequence.
Post Conditions	The sensors shall be checked manually for their accuracy.

Table 3.3: Measuring Soil Moisture

Use-Case Name	Mowing All Weeds
Actors	User, Web Application
Description	This function utilises a rotary tool with the weed-whacking implement to perform powered weeding on all weeds in the garden.
Data	The positions of the planted weeds
Preconditions	The rotary tool is mounted to the UTM.
Stimulus	User presses the mow all weeds command from the sequences panel on the web app.
Basic Flow	*Step 1:* The web app sends the command to the FarmBot. *Step 2:* The FarmBot sends an F-code command to the Farmduino to begin monitoring the rotary tool's load sense pin. *Step 3:* The firmware sends FarmBot-calculated step pulses for the Z axis to move to a staging position 20mm above the weed. *Step 4:* The firmware turns on the rotary tool.

	Step 5: The mechanical system descends the tool to the weed's height using the calculated step pulses by interpolating between the nearest soil height measurements. *Step 6:* The rotary tool mows the weed at 25% speed. *Step 7:* Farmduino makes the rotary tool ascend and turns it off by writing to its pins. *Step 8:* Then the rotary tool proceeded to the next weed until the garden was visited.
Alternative Flow#1	*Step 3:* If no soil height measurements are available, the weed's Z coordinate will be set to the FALLBACK SOIL HEIGHT as defined in the settings panel.
Alternative Flow#2	*Step 2:* The garden is all mowed already. *Step 3:* FarmBot notifies the user via the web interface.
Exception Flow	*Step 1:* The rotary tools stalled due to vines of the weeds. Manual intervention is required.
Post Conditions	Rotary tool should be cleaned and checked.

Table 3.4: Mowing all Weeds

Use-Case Name	Scanning the Garden for Weeds
Actors	User, Web Application
Description	This function scans the whole garden for weeds using a group of points.
Data	Images of grids in the garden
Preconditions	FarmBot's camera must be calibrated
Stimulus	User selects the scan the garden for weeds option in the panel.

Basic Flow	*Step 1:* User creates a grid of points for every location they want FarmBot to take a photo at. The user enters the info of the starting location, number of points in each direction and the spacing between those points. The user toggles the camera view area. *Step 2:* The web application sends commands to the main system through the message broker. *Step 3:* Main system sends G-code to Arduino Firmware over a serial connection to move the camera to the starting location. *Step 4:* The Farmduino calculates the number of steps needed based on the received G-code command. Then, it generates the appropriate number of steps and corresponding direction pulses to the mechanical system. *Step 5:* The mechanical system receives pulses interprets these signals, and makes the motors move in the desired direction and number of steps. *Step 6:* After setting the camera location, the main system runs weed detection software and requests to take a photo at each selected point from the monitoring system. *Step 7:* After taking the photo and processing it, FarmBot uploads it to the web app, along with the coordinates from where the photo was taken, the date and time with the identified weeds using advanced computer vision technologies. *Step 8:* The received photos are viewed in the farm designer on the web app.
Alternative Flow#1	-
Alternative Flow#2	-

Exception Flow	*Step 7:* The processed images are irrelevant. The camera needs calibration.
Post Conditions	Camera needs calibration again.

Table 3.5: Scanning the Garden for Weeds

Use-Case Name	Axis Setup
Actors	User, Web Application
Description	This function finds and sets FarmBot's home position and axis lengths, and restricts FarmBot from moving beyond those boundaries.
Data	Home position coordinates, all three axis lengths
Preconditions	Encoders, stall detection, or limit switches must be wired up correctly and be on.
Stimulus	User may power up the FarmBot, User may request movement error, or User may move the FarmBot by hand.
Basic Flow	*Step 1:* User presses the find home button on the app. *Step 2:* The command is delivered to the FarmBot via the message broker. *Step 3:* FarmBot sends a G-code command to Farmduino to generate direction and step pulses to the mechanical system. *Step 4:* The mechanical system receives the pulses and advanced stall-detecting stepper drivers start moving in the direction of the maximum position, searching for the axis lengths.

	Step 5: Using advanced special hardware, stepper drivers detect when the maximum position has been reached and will signal to FarmBot. *Step 6:* FarmBot starts moving towards the home position, searching for it. *Step 7:* Utilising advanced technologies of the stepper drivers, FarmBot will receive a signal when the home position has been reached. *Step 8:* FarmBot sets the axis length according to the measured distance and number of steps between the maximum and home positions. It also sets the current location as the home position. *Step 9:* The acquired logs are uploaded to the Web application.
Alternative Flow#1	*Step 4:* The mechanical system utilises rotary encoders on each motor to implement requested pulses. *Step 5:* These sensors measure the exact rotation of the motor shafts in real time. They will signal to FarmBot when a motor has stalled. *Step 6:* These sensors keep track of FarmBot's position, hence they will alert the FarmBot when the end of an axis has been reached or the home position has been found. *Step 7:* Real-time position data is received by the FarmBot. *Step 8:* The FarmBot uploads the axis lengths and home position to the web app via a message broker.
Alternative Flow#2	*Step 4:* The mechanical system utilises limit switches which are small buttons that can be positioned at axis end locations.

	Step 5: Whenever FarmBot reaches an end location, it will press the button, which creates a signal indicating the axis end has been reached. *Step 6:* The system updates axis data to the web app available on the Farm Designer.
Exception Flow	*Step 5:* The motor has stalled and raised an error signal. *Step 6:* The motor settings such as steps per mm, or encoder scaling must be reviewed by the user to ensure the settings are correct for the hardware. The motor current value is set to default value and tries to energize the motor to move. *Step 7:* The system sends logging via the message broker that an unknown issue is raised, and a manual check is required.
Post Conditions	-

Table 3.6: Axis Setup

Use-Case Name	Connecting FarmBot to the Web App
Actors	User, Web Application
Description	This function connects FarmBot to the Web App with the relevant user.
Data	WiFi credentials, web app and user credentials
Preconditions	User must have created and verified a web app account already. Farmbot must be connected to the Internet.
Stimulus	User configures the FarmBot via web interface.
Basic Flow	*Step 1:* User should allow access to ports MQTT, MQTT over SSL, HTTP, and HTTPS if they are restricted. *Step 2:* User configures the FarmBot with Wi-fi and web app credentials.

	Step 3: FarmBot disables the Wi-Fi network, and boots up. *Step 4:* The FarmBot finds the network and checks the Wi-Fi credentials such as the network password entered by the user during the configuration. *Step 5:* FarmBot connects to the Internet to send and receive messages. *Step 6:* The FarmBot checks the web application credentials such as the email and password entered by the user during the configuration. *Step 7:* FarmBot connects to the web application and finishes its initialization process. *Step 8:* The FarmBot's model and configuration information is uploaded to the web app via a message broker. *Step 9:* Web app shows the FarmBot.
Alternative Flow#1	*Step 5:* User has entered Wifi credentials incorrect. *Step 6:* Web app states FarmBot could not connect to the network.
Alternative Flow#2	*Step 7:* User has entered web app details incorrect. *Step 8:* Web app states the connection is not successful.
Exception Flow	*Step 3:* The connection is lost during the operation.
Post Conditions	Changes in credentials should be configured to FarmBot as well.

Table 3.7: Connecting FarmBot to the Web App

Use-Case Name	Dispensing Water
Actors	User, Web Application

Description	This function moves to a plant and waters according to the plant's age and its assigned water curve.
Data	age of plants and their assigned sowing information
Preconditions	The watering nozzle tool must be mounted before dispensing water.
Stimulus	User presses the dispense water command on the panel.
Basic Flow	*Step 1:* The user presses the dispense water command on the app. *Step 2:* Web app requests the data of the garden layout, dates of plantations and many other logs from the database management system. *Step 3:* The data is sent with the command to the FarmBot. FarmBot processes the data and takes the relevant crop data from the crop database. *Step 4:* FarmBot selects the solenoid valve and sends relevant data of the amount of time it calculated according to the flow rate of the water and the volume of water the plant needs. *Step 5:* The firmware writes to the pins to turn the solenoid valve on for the requested values. *Step 6:* The firmware synchronously reads the watering sensor pins and sends it to FarmBot. *Step 7:* After the completion of dispensing, FarmBot notifies the web app that the sequence is completed.
Alternative Flow#1	*Step 4:* FarmBot notifies the user that the plants do not need watering currently.
Alternative Flow#2	-
Exception Flow	*Step 4:* The data from the crop database cannot be accessed. The user needs manual calculation.

Post Conditions	The date of watering activity should be saved and sent to the web app.

Table 3.8: Dispensing Water

Use-Case Name	Sowing a Weed
Actors	User, Web Application
Description	This function adds plants to the map, builds planting sequences and sows the seeds.
Data	Weed species
Preconditions	FarmBot must be completely assembled, powered up, and connected to the web app. The vacuum pump, seeder tool, and motor moving operations must work without error.
Stimulus	User adds some plants to the map on the web app. Then, the user runs the default sequences on the panel to sow and do more to the seeds.
Basic Flow	*Step 1:* The Farmbot receives the command, and initially sends commands to Farmduino to mount a seeding needle to the tool head. *Step 2:* After mounting the needle, the Firmware sends direction pulses to the mechanical system to move the Z axis above a gantry-mounted seed. *Step 3:* The Firmware writes to the pins of the vacuum pump to turn it on for picking up a seed. Then a seed is suction-held onto the needle.

	Step 4: When the Farmduino reads that seed is held, The tool head is moved to the user-defined location variable, utilising advanced technologies of rotary encoders or stepper drivers. *Step 5:* Farmduino write to the pins of the vacuum pump to turn it off when the motors stall. Thus, the seed is released in the soil. *Step 6:* The rotary encoders send the current position to the FarmBot. FarmBot marks the current position with the date of plantation and type of the seed. *Step 7:* The FarmBot sends logs data to the web application to be saved in the database, and shown on the web app.
Alternative Flow#1	*Step 2:* There is already a mounted tool. Firmware alerts the FarmBot. *Step 3:* FarmBot sends a message to the web app to notify the user that the dismounting tool is required.
Alternative Flow#2	*Step 3:* The inventory is out of the selected seed. *Step 4:* The issue remains as unknown.
Exception Flow	*Step 1:* The rotary tool is stalled during digging the dirt. Manual intervention is a must.
Post Conditions	The seed inventory and other tools should be checked regularly.

Table 3.9: Sowing a Seed

Use-Case Name	E_stop and unlocking
Actors	User, Web Application

Description	This function shuts down the firmware and motors to prevent any harm in case of emergencies.
Data	E_lock code
Preconditions	-
Stimulus	User clicks the E-stop button from the app or manually on the device.
Basic Flow	*Step 1:* The web app sends a command to the FarmBot to lock the Farmduino microcontroller. *Step 2:* FarmBot's emergency stop button is pressed and it sends a signal to Farmduino to enter the emergency stop state on its software. *Step 3:* The signal also causes cutting power to motor drivers, halting any other peripheral usage and resetting peripheral pins to OFF. *Step 4:* The user or system resolves the issue and presses unlocking. The emergency stop button on the FarmBot is released and Farmbot initiates a reset command on the Farmduino. *Step 5:* Farmduino boots up with default values. *Step 6:* FarmBot sends default logs and state of the system to the web app.
Alternative Flow#1	-
Alternative Flow#2	-
Exception Flow	The connection is lost, the firmware is not locked.
Post Conditions	Before booting up, a manual check is suggested.

Table 3.10: Emergency Stopping and Unlocking

3.3 Logical Database Requirements

3.3.1 Logical Database Requirements

- Each user registered in the system shall have a unique user ID.
- Users shall be able to change their information except for the user ID.
- Username shall be unique and cannot be shared among multiple users.
- Email addresses shall be unique and cannot be shared among multiple users.
- The password information shall be securely stored and only accessible for authentication purposes.
- The `created_at` and `updated_at` fields shall be automatically generated and updated by the system for tracking account creation and information modification times.
- The `language` field shall store the preferred language of the user for localization purposes.
- Each point shall be associated with a device. This relationship indicates which device the point is related to. One device can have many points, but each point belongs to only one device.
- A generic pointer can used to give the device commands about soil.
- Point can be discarded after its use.
- Weeds are points used to manage weeds in a garden, including removal, protection, or watering. They have a plant stage indicating the status (active, removed, or pending) of the weed.
- Plants represent points related to planting, configuring, and managing plants in a garden. `openfarm_slug` is the unique canonical name of the plant in the database.

- Each plants, weed and generic_pointers shall be points. They shall be stored as points in the database.
- Plant Templates shall store to give the user information about the plant type and its location in a saved garden.
- Plant groups and point groups shall be used to group objects in a saved garden and manage them collectively.
- Different garden layouts shall be saved and can be used alternately.
- Curves shall be used to define plant water needs and expected height and spread over time.
- Curves shall be plots of data and shall be shown in web application as a graph.
- Curves shall be adjustable to suit changing needs of plants.
- Sequences shall allow users to combine commands of the FarmBot Express. Maximum step of the sequence associated a device shall be stored in the database along with the corresponding device.
- Variables shalll utilize sequence definition by serving as a data storage.
- Sequence Steps shall be the most basic commands of the FarmBot Express.
- Flow rate of the tools per second shall be stored in the database to precise usage of the tools (unit shall be milliliters).
- Sensor pins shall be stored to send signals to the sensor appropriately
- Create dates of the most of the object shall be stored to able users to filter with creation date.
- Regimens shall make users able to create reusable recipes that can take care of a plant throughout its entire life.
- Events shall make able users to schedule regimens and sequences.

- Events shall make able users to select repeat times of the regimens or sequences of the corresponding event.
- Alerts shall be important messages to alert users.
- Alerts shall be distinguishable from the messages easily.
- FarmBot Express operation system version shall be stored along with the device for proper usage of the device.
- Information of whether FarmBot Express is used indoor or outdoor shall be stored in the database.

3.4 Design Constraints

- **Environmental Regulations:** FarmBot Express is designed for agricultural applications, including the cultivation of edible plants. Therefore; one important rule FarmBot hardware must follow is about not using harmful materials. This helps protect people and the environment from dangerous substances. This way, FarmBot can be safe for farmers to use and prevent harmful chemicals from getting eat or getting into the soil or water. The ecological footprint shall be kept as small as possible.
- **Safety Standards:** The design must comply with safety standards, ensuring the safety of operators, bystanders, and the environment. This includes features like emergency stop mechanisms, collision avoidance, and fail-safe operation.
- **Resource Constraints:** There are constraints which are rooted from the resources that potential users have, such as availability of skilled personnel who can manage hardware, and technological infrastructure which is required for system deployment such as electricity and wifi.
- **Energy Efficiency:** FarmBot shall be designed with energy efficiency in mind, utilizing low-power components and optimizing algorithms to minimize energy

consumption during operation.

- **Affordability and Accessibility:** If FarmBot is intended for small-scale or resource-constrained farmers, the system shall be affordable and accessible, with consideration for cost-effective design solutions.

3.5 System Quality Attributes

3.5.1 Usability Requirements

- **Requirement 1:** FarmBot Express shall precisely use the tools, keeping sensitivity and needs of crops and soil in mind. For example, it shall precisely dispense water to crops, ensuring enough coverage and adequate (neither more nor less) hydration for their health.
- **Requirement 2:** FarmBot shall adjust its actions dynamically to accommodate variations in sensor data. Therefore; it can meet the needs of crops and weeds more accurately.
- **Requirement 3:** FarmBot Express shall follow commands given by the user in sequences, executing tasks according to the specified sequence without any deviation or interruption in normal circumstances (except user interruption or emergency stop).
- **Requirement 4:** FarmBot Express shall generate and transmit logs to the user interface upon completion of each job or command, providing detailed information about the executed tasks.
- **Requirement 5:** FarmBot shall provide feedback on task completion status, indicating when each task is initiated, in progress, or successfully completed.
- **Requirement 6:** The system shall provide intuitive user interface for both novice and experienced users. It should be kept in mind that users may be people who are far from technology.

- **Requirement 7:** The system shall support multiple languages for user interfaces to cater to international users.
- **Requirement 8:** The system shall minimize training length by presenting information in a clear and organized manner.
- **Requirement 9:** Users shall be able to customize settings and preferences to tailor the system to their needs.
- **Requirement 10:** Clear documentation, including user manuals, shall be provided to assist users in troubleshooting and customization. This simplifies user training.
- **Requirement 11:** Peripherals of the FarmBot Express shall be readily available from authorized vendors or through an official supply chain to minimize effort to find them.
- **Requirement 12:** The hardware design shall incorporate fault-tolerant mechanisms to mitigate the impact of component failures. Fault tolerance is important as the hardware works outside and stays in harsh conditions when necessary.
- **Requirement 13:** The hardware interfaces, including buttons and switches, shall be ergonomically designed and labeled clearly for intuitive operation by users. Replicates of the hardware interfaces shall be shown in user interface if necessary.
- **Requirement 14:** The hardware assembly shall be designed to minimize sharp edges or protrusions that could cause injury to users during installation or maintenance. This requirement also prevents animals and plants from being injured.
- **Requirement 15:** FarmBot shall detect and alert users to any errors or deviations from expected task outcomes such as rotary tool stall.
- **Requirement 16:** FarmBot shall provide clear and comprehensible error handling messages or feedback, preferably including suggestions and solutions, to assist users in resolving issues efficiently and resuming operations.

3.5.2 Performance Requirements

- **Requirement 17:** FarmBot Express shall accurately move to locations at specified coordinations within a tolerance of ±1mm (the difference between two coordination points).

- **Requirement 18:** FarmBot Express shall execute scheduled tasks, such as watering crops, at the specified times with a deviation tolerance of no more than ±1 minute from the scheduled time.

- **Requirement 19** FarmBot Express shall repeat specified tasks, such as removing weeds, the designated number of times as instructed by the user, with no tolerance.

- **Requirement 20:** The system shall support a full garden configuration with crops, weeds and points simultaneously without degradation of performance. The grid of the FarmBot Express is 1.2 meters to 3 meters. So it can hold maximum of around 100 plants.

- **Requirement 21:** Response time of the system shall be less than 1 seconds under normal connection conditions. It is important for critical functions such as emergency stop.

- **Requirement 22:** The system shall accurately identify the crops and weeds grown in the garden. Therefore: more appropriate maintenance can be performed and unwanted losses can be prevented.

- **Requirement 23:** The webcam shall provide high-resolution images to allow FarmBot to process images, detect crops and weeds, and differantiate plants. Additioanlly; this allows to monitor their garden remotely.

- **Requirement 24:** The webcam shall have adjustable angles and calibration settings to provide users with clear views of the farming environment.

- **Requirement 25:** Sensors integrated into FarmBot Express shall provide accurate measurements within a reasonable tolerance of the actual value under normal operating conditions Accurate sensor measurements are essential for precise control of agricultural parameters such as soil moisture.

3.5.3 Reliability Requirements

- **Requirement 26:** The system shall efficiently manage memory and system resources of its embedded hardware to prevent system crashes.
- **Requirement 27:** The system shall have a mean time between failures (MTBF) of at least 336 hours (2 weeks). This requirement aims to ensure the system's reliability during extended periods of operation without supervision, such as during holidays, covering approximately 90 percent of typical holiday durations.
- **Requirement 28:** Communication between the user interface and hardware of FarmBot shall be seamless, even in harsh conditions. Any connection loss or data loss during transactions shall be promptly reported to the user to ensure accurate work.This requirement ensures correct operation and accuracy of FarmBot.

3.5.4 Availibility Requirements

- **Requirement 29:**: The user interface shall guarantee 98% uptime excluding scheduled maintenance windows. Because user interface is crucial for FarmBot Express to implement its regular tasks.
- **Requirement 30:** The FarmBot Express system shall require the presence of trained maintenance staff available for on-site assistance and support.

3.5.5 Security Requirements

- **Requirement 31:** User authentication shall be enforced for accessing sensitive functionalities or data. This protects the user garden from external threats.
- **Requirement 32:** Passwords shall be securely stored using encryption techniques to prevent unauthorized access. This protexts the user gard from externak

threats.

- **Requirement 33:** The system shall implement emergency actions to prevent hazards during hardware operations. This requirement is critical to ensure the safety of personnel, crops, animals and the environment.

- **Requirement 34:** The system shall log and audit user activities, including login attempts, command executions, and configuration changes, for security monitoring and forensic analysis.

3.5.6 Maintability Requirements

- **Requirement 35:** The FarmBot web application code and operating system code shall be modularized and well-documented to facilitate easy maintenance and future enhancements. This speeds up the development and update process of the system.

- **Requirement 36:** Clear documentation, including technical guides, shall be provided to developers in troubleshooting and development. This simplifies troubleshooting and development.

- **Requirement 37:** The hardware components shall be modularized to allow for easy replacement and upgrades without requiring extensive disassembly. This makes the moving and shipping processes safer.

- **Requirement 38:** Dependency of the modules to other modules of the embedded system shall be as small as possible, allowing for independent testing and maintenance.

- **Requirement 39:** Standard components shall be preferred in the hardware design to facilitate easy replacement and availability.

- **Requirement 40:** Comprehensive documentation shall be provided for all hardware components, including specifications, schematics, and assembly instructions.

- **Requirement 41:** Critical hardware components such as motors and sensors shall be designed for easy replacement by end-users, with minimal tools and technical expertise required.
- **Requirement 42:** The FarmBot Express system shall be designed for easy disassamble to accommodate changes in farm layout. As farm layout may wear out or be damaged easily, faster re-assamble provides convenience.

3.5.7 Portability Requirements

- **Requirement 43:** The assembly of FarmBot Express shall be designed to be easy and require a minimal number of tools .This requirement aims to reduce the time, effort and number of tools required for assembly.
- **Requirement 44:** The FarmBot interface shall be responsive and adaptive, providing consistent user experiences across various screen sizes and resolutions. This is important to provide a user interface available from different devices.
- **Requirement 45:** The system shall support standard communication protocols, such as MQTT or RESTful APIs, to enable interoperability with third-party devices and services.
- **Requirement 46:** Installation and update procedures shall be automated (if possible) and well-documented to simplify deployment on different platforms and systems.
- **Requirement 47:** The webcam integrated into FarmBot Express must be designed to be replaceable, allowing for easy upgradeability of the system.As technology evolves, there may arise the need to upgrade the webcam component of FarmBot Express to take advantage of improved camera capabilities.

3.6 Supporting Information

Despite the many functionalities it offers, FarmBot's main purpose is not large-scale agriculture.The initiative to introduce the FarmBot technology aims to increase interest in gardening and plant cultivation. Having a Farmbot would enable groups to increase interest in the idea of gardening and raising plants in a number of ways. Through gamification students would learn the best way to place plants and discover the needs of plants and how to observe and check on those needs. Additionally the Farmbot would give the opportunity to teach additional skills such as robotics and computer programming. Users, particularly students, can learn fundamental principles of robotics and automation by programming FarmBot to perform specific tasks such as planting, watering, and harvesting.

4. Suggestions to Improve The Existing System

4.1 System Perspective

Context diagram and explanations of context diagram go here for suggestions to improve the existing system. Plus, other content as appropriate.

Refer to *(Clause 9.6.4, 9.5.4.1)*.

4.2 External Interfaces

External Interfaces Class Diagram and its explanations go here for suggestions to improve the existing system. Plus, other content as appropriate

Refer to *(Clause 9.6.11, 9.5.8)*.

4.3 Functions

Use-Case Name	Alarming mechanism
Actors	User, Web application
Description	Web application notifies the user, against potential animal or theft threads near or inside the garden.
Data	????????
Preconditions	Motion, proximity, vibration sensors; sirens, and strobe lights must be assembled correctly with the FarmBot kit. The FarmBot's software must be updated and configured customarily. The web application interface should be updated for the sensors and alarm panels.
Stimulus	User adds and toggles the motion, proximity and vibration sensors.
Basic Flow	*Step 1:* Web application sends sensor toggling command to FarmBot via a message broker. *Step 2:* FarmBot sends code commands to the Farmduino to turn the relevant sensors on. Firmware writes to the assigned pins. *Step 3:* When sensors detect movements, and the presence of an object or vibration, they directly send signals to the FarmBot. *Step 4:* FarmBot interprets this signal as a potential threat and triggers the alarm system. *Step 5:* The alarm system activates sirens and strobe lights and sends alerts via message broker to notify the user.
Alternative Flow#1	*Step 2:* ???
Alternative Flow#2	*Step 2:* ???
Exception Flow	*Step 1:* ???
Post Conditions	???

Use-Case Name	Automated Backup Data Sequence
Actors	User, Web application
Description	This function provides extra storage for the user to access the data in case of network loss.
Data	??????????
Preconditions	The external SD card must be connected to the correct pins on the FarmBot. Users must install developer-defined functions with libraries or software packages to interface with the storage device on the FarmBot microcontroller. The user must update the web application.
Stimulus	User chooses the external storage device from the panel.
Basic Flow	*Step 1:* User chooses the backup to external device option from the sequences panel. *Step 2:* User enters the period of the sequence. *Step 3:* FarmBot saves the configuration settings, sensor data, or critical system logs regardless of the network connection. *Step 4:* FarmBot interprets this signal as a potential threat and triggers the alarm system. *Step 5:* After saving the relevant data, FarmBot sends a message to the web app to notify the user that the backup attempt was successful. *Step 6:* Web app shows the usage of external storage device.
Alternative Flow#1	*Step 2:* ???
Alternative Flow#2	*Step 2:* ???
Exception Flow	*Step 1:* backup attempt failed
Post Conditions	???

Use-Case Name	Sensor Calibration
Actors	User, Web application
Description	This function checks the reliability and accuracy of the sensors against the duration of usage, and external factors (e.g. weather conditions). Then, it calibrates the sensors regarding conditions.
Data	??????????
Preconditions	The documentation for the calibration process of the sensors must be saved to the crop database. The web application must be up to date.
Stimulus	User selects sensor to test from the panel.
Basic Flow	*Step 1:* Web app sends a command to FarmBot to test the user-selected sensor. *Step 2:* FarmBot sends a G-code to the Arduino firmware. *Step 3:* Firmware writes to the pins of the relevant sensor to turn it on. *Step 4:* The received data is sent to the FarmBot to process. *Step 5:* FarmBot compares the referenced and measured data based on several parameters. *Step 6:* FarmBot sends a message to the web app stating that the sensor measures accurately.
Alternative Flow#1	*Step 2:* sensor doesn't measure accurately.
Alternative Flow#2	*Step 2:* sensor needs calibration.+
Exception Flow	*Step 1:* sensor measurement is failed.
Post Conditions	???

Use-Case Name	Detecting Pest and Diseases
Actors	User, Web application
Description	This function notifies for the early stages of plant diseases and pests before they spread to the garden.
Data	??????????
Preconditions	The web app and FarmBot's software should be up to date.
Stimulus	User selects the pest and disease detecting sequence from the sequences panel.
Basic Flow	*Step 1:* Web application sends the command to the FarmBot to check plants regularly via webcam. *Step 2:* FarmBot requests image data from the monitoring system with the user-defined period. *Step 3:* Using the advanced computer vision algorithms, FarmBot processes the image data. *Step 4:* FarmBot raises a signal to the web app if there is any detection of pest or disease-like changes in the plants. *Step 5:* Web app notifies the user that there has been detection of the plant disease/pest.
Alternative Flow#1	*Step 2:* If there is no detection, the user is not notified.
Alternative Flow#2	*Step 2:* ???
Exception Flow	*Step 1:* ???
Post Conditions	???

Refer to *(Clause 9.6.12, 9.5.5, 9.5.10).*

4.4 Logical Database Requirements

Key data objects (persistent or not) and their major attributes for suggestions to improve the existing system. Draw the **Class Diagram** with associations. A class dictionary can be omitted, provided that the naming is understandable.

Refer to *(Clause 9.6.15)*.

4.5 Design Constraints

- **Cost:** Ensure that any proposed enhancements remain cost-effective to maintain affordability for small-scale farmers and home gardeners.

Refer to *(Clause 9.6.16)*.

4.6 System Quality Attributes

- **Requirement 1:** Automated backup mechanisms shall be implemented to prevent data loss in the event of system failure.
- **Requirement 2:** The system shall be capable of recovering from faults or errors without manual intervention.
- **Requirement 3:** The FarmBot software shall be platform-independent and compatible with major operating systems, including Windows, macOS distributions.
- **Reqirement 4:** Sensors and actuators shall provide feedback to users through visual indicators or audible signals to indicate their status or operation.
- **Requirement 5:** Sensors shall be calibrated regularly to maintain accuracy and reliability over time, with calibration intervals not exceeding [specify duration].
- **Requirement 6:** FarmBot Express hardware shall be equipped with a locking and alarm mechanism to prevent theft of its components. Theft prevention is essential as it resides outside.
- **Requirement 7:** FarmBot Express shall provide integration with popular smart home platforms such as Google Home or Amazon Alexa to enable voice commands and seamless automation. INTERFACE ¡API¿

- **Requirement 8:** Integrate robust data analytics capabilities into the FarmBot Express to collect and analyze data on crop performance, resource usage, and environmental factors. User interface shall provide the user with analyzed and detailed information about the garden. INTERFACE ¡ML¿
- **Requirement 9:** By analyzing data, it shall be able to provide information about how to care for the plants from then on. INTERFACE ¡ML¿
- **Requirement 10:** FarmBot Express shall offer compatibility with weather forecast APIs to adjust planting and watering schedules dynamically based on predicted weather patterns, minimizing water waste and maximizing crop resilience.
- **Requirement 11:**FarmBot Express shall support import and export of gardening data in common file formats (such as CSV or JSON), allowing users to exchange data with external software tools, cloud services, or third-party platforms for analysis, backup, and integration purposes.
- **Requirement 12:** Provide localized support and documentation in multiple languages to cater to users worldwide.
- **Requirement 13:** Provide seasonal gardening recommendations and tips to guide users on planting schedules, crop selection, and maintenance tasks throughout the year.

4.7 Supporting Information

As mentioned in the `Purpose of the System` ,FarmBot Express is not just a gardening tool; it is an educational platform designed to inspire and educate students in botany, robotics and programming. By incorporating data analytics and machine learning models into the FarmBot Software, the learning experience is taken to the next level. Students who receive practical advice and predictions derived from the current data of the garden can accelerate their learning process. It's like having a personalized guide that helps them understand the intricate connections between plants and environmental conditions more quickly. Additionally, they can delve deeper into the characteristics of the plants, expanding their knowledge.