NOVAFLECT

Crop Disease Surveying Drone Software

Software Requirements Specification
For No Name Brands Software

Version 1.0

Crop Disease Surveying Drone Software	Version: 1.0
Software Requirements Specification	Date: 23/10/2023

Table of Contents

Table of Contents **2**
1. Introduction **4**
1.1 Purpose 4
1.2 Scope 4
1.3 Definitions. Acronyms, and Abbreviations 4
1.4 References 5
1.5 Overview 5
2. Overall Description **5**
2.1 Product Perspective 5
2.2 Product Functions 6
2.2.1 Use Case Diagram 1 6
2.2.2 Use Case Diagram 2 7
2.3 User Characteristics 7
2.3.1 "Big Fun Agriculture" Personnel: 7
2.3.2 External Farmers 7
2.3.3 System Administrators 8
2.3.4 Data Entry Personnel 8
2.3.5 Agricultural Experts 8
2.4 Constraints 8
2.5 Assumptions and Dependencies 9
2.5.1 Database Management System: 9
2.5.2 Network Connectivity: 9
2.5.3 Hardware and Security Card integration: 9
3. Specific Requirements **10**
3.1 External Interfaces 10
3.1.1 Integration with Weather Forecasting 10
3.1.2 Integration with Plant Diseases Library 11
3.1.3 Integration with Pesticide Inventory Systems 12
3.1.4 Integration with Drone Systems 14
3.2 Functions 15
3.2.1 Requirements 15
3.2.2 Use Cases 16
3.2.2.1 View Crop Status of a field 16
3.2.2.2 View Sector Information 16
3.2.2.3 View Weather Information 17
3.2.2.4 View Pesticide Inventory 18
3.2.2.5 View Drone Status 19
3.2.2.6 View System Notification 19
3.3 Performance Requirements 20
3.4 Logical Database Requirements 20
3.4.1. Image Data 20
3.4.2. Crop Data 21

3.4.3. Disease Data 21
3.4.4. Inventory Data 21
3.4.5. Map Data 22
3.5 Design Constraints 22
3.5.1. Platform support 22
3.5.2. Processing Speed 22
3.5.3. Data Storage and Performance 22
3.5.4. Internet Access 22
3.5.5. Agricultural Regulations 22
3.5.6. Weather Conditions 22
3.5.7. Project Schedule 22
3.6 Software System Quality Attributes 23
3.6.1 Security 23
3.6.2 Reliability 23
3.6.3 Availability 23
3.7 Object Oriented Models 24
3.7.1 Architecture Specification 24
3.7.2 Domain Model 25
4. Appendix 25
4.1 Sample Inputs & Outputs 25

1. Introduction

1.1 Purpose

The purpose of this document is to outline the comprehensive system requirements for the Crop Disease Surveying Drone Software currently in development for Big Fun Agriculture. The requirements presented in this document were developed with the client's initial request for proposal in mind and have been further refined through subsequent discussions and elicitation processes.

1.2 Scope

The software product is a machine learning mobile application that will integrate the existing modules and systems used by Big Fun Agriculture to maximize profits and improve crop yield and quality in the following ways:

- By assisting workers and farmers with identifying potential diseased/infected crops
- By presenting detailed information of infected crops such as type of disease/infection, degree of damage, and availability of treatments
- By having a system that is compatible with supplied drones, cameras, and information modules such as dirt analyzer and weather readings; and
- By maintaining database information such as the disease library and pesticide inventory to ensure that misreads are minimal and stock does not run out

The farmers and workers will be able to interact with the application through a user-friendly GUI, where they will be able to monitor the status of the crop fields and the associated information with them. The software will be able to cover the main farm with a size of 300 acres.

The main goals of the application is to prevent the development of infection by detection in early stages using drones and cameras, and to provide steps on how to treat them accordingly. The project also sets out to accomplish a higher crop yield and quality over the year, while also reducing the time and logistics (such as manual labor) required for disease detection.

1.3 Definitions. Acronyms, and Abbreviations

Database:	A collection of data that is organized and stored for efficient retrieval and manipulation.
API:	An Application Programming Interface (API) is a set of rules and protocols that allows different software applications to communicate between one another.
DBMS:	A Database Management System (DBMS) is software implemented to manage databases by

	storing, retrieving, and organizing data.
GUI:	A Graphical User Interface (GUI) is a visual way for users to interact with software.
JSON:	JavaScript Object Notation is a lightweight data-interchange format that is both easily parsable and understandable.
Disease Library:	A database containing information about various plant diseases for identification and research.
Pesticide Inventory:	A database that tracks and manages the inventory of stored pesticides for disease treatment.
UC:	A Use Case (UC) is a description of how a system interacts with its user.

1.4 References

IEEE Std 830-1998. IEEE Recommended Practice for Software Requirements Specifications. Available online at http://segal.cs.uvic.ca/seng321/lectures/IEEE_Standard_1998.pdf.

Big Fun Agriculture. Crop Disease Surveying Drone Software.September 24th 2023. Available online at https://4312website.github.io/Website1/static/media/Deliverable1.4fc44b4fa74584311d9c.pdf

1.5 Overview

The following content in this document is a comprehensive description of the solution along with the specific requirements of the system.

2. Overall Description

2.1 Product Perspective

The suggested crop disease detection and treatment system addresses the urgent problem of agricultural crop diseases and fungal infections by providing a strong business case and operational approach. This programme was created to effortlessly integrate with "Big Fun Agriculture's core operations" and is in line with the organization's aim of maximizing agricultural yield, quality, and sustainability. To ensure accurate data, disease diagnosis, and treatment suggestions, the system integrates with numerous external systems, such as weather forecasting services, plant disease databases, and pesticide inventory systems. It must efficiently manage memory for enormous datasets, ensure operational resilience in the field, and adapt to a variety of agricultural areas, among other challenges. The system essentially offers itself as a key instrument for improving crop management, combining hardware, software, and data-driven insights to reduce losses and optimize resource allocation in line with the organization's goals.

2.2 Product Functions

The crop disease detection and treatment system has the following:

Crop Disease Detection: By gathering information from hardware devices, such as drones and cameras, the system is able to identify probable crop illnesses and fungi infections.

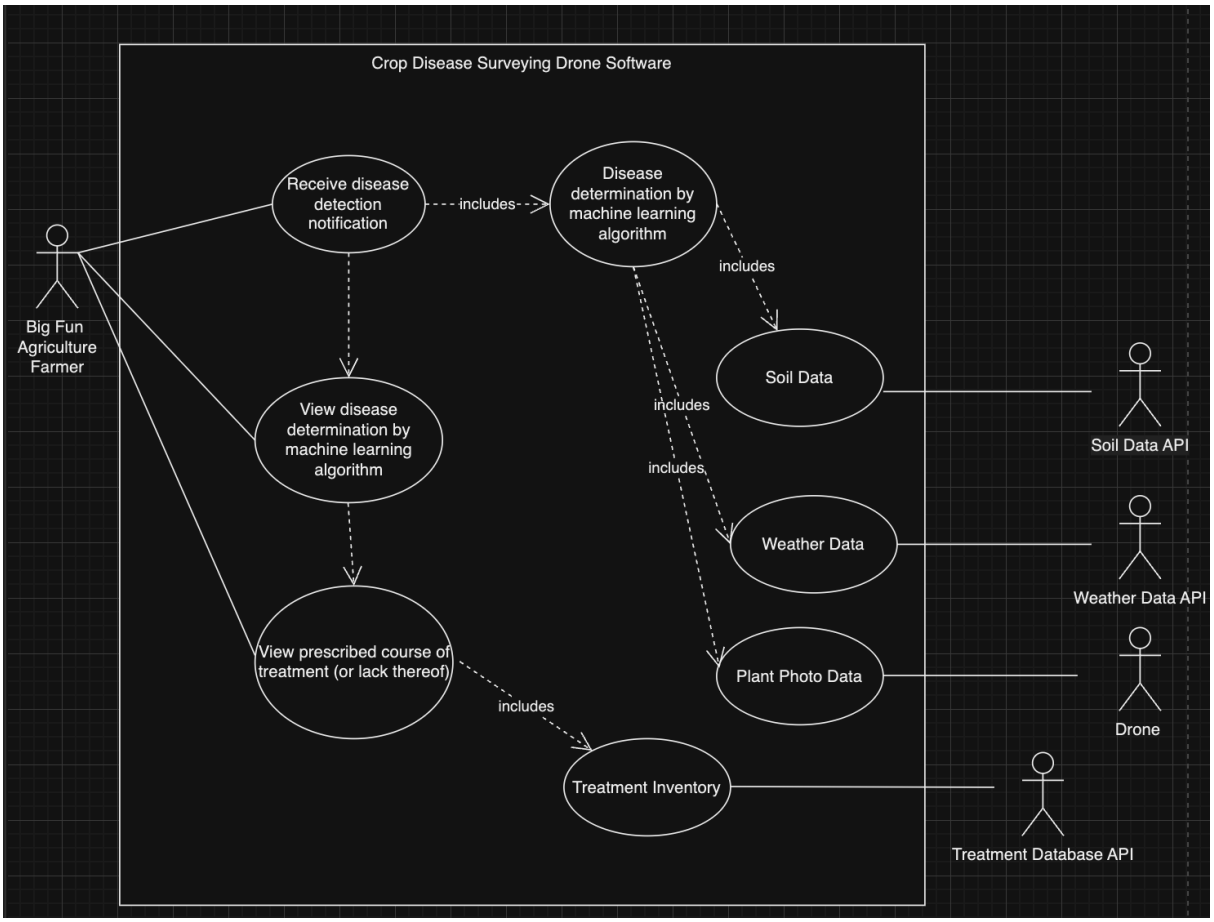
Disease Diagnosis: After the sickness has been discovered, the system uses cutting-edge computer vision (CV) and AI/ML algorithms to determine the kind and severity of the condition and to give users insights.

Treatment Recommendations: After a diagnosis, the system recommends the best courses of action based on the nature of the disease and the stock of pesticides that are readily available.

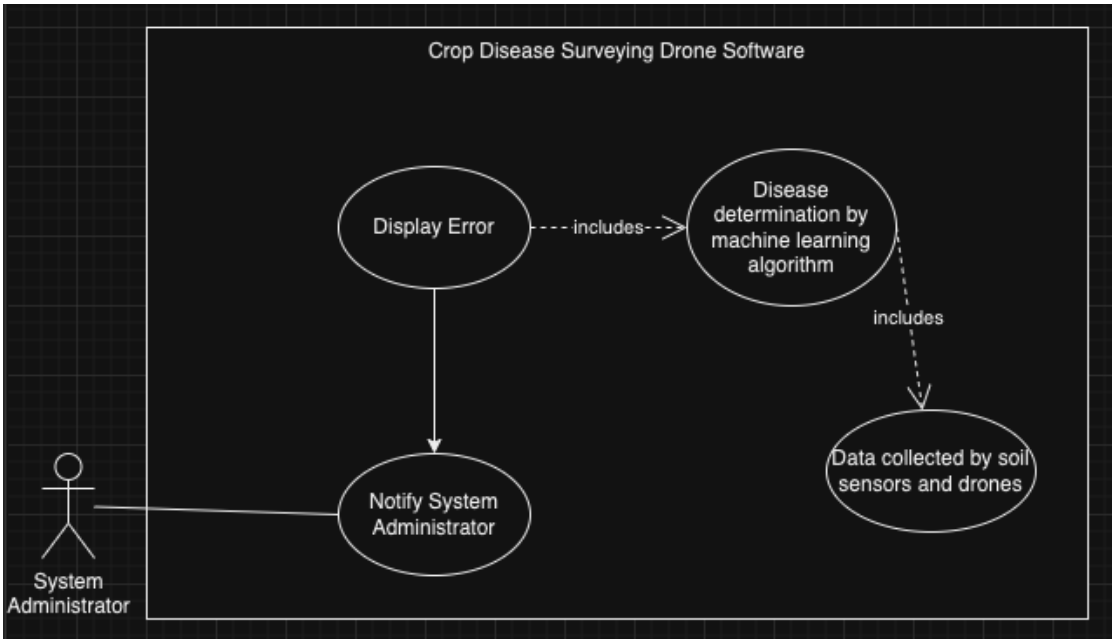
Data Integration: To improve the accuracy of its findings, the system seamlessly combines data from numerous sources, including as weather forecasts, databases of plant diseases, and pesticide inventory systems.

User Interface: Staff and outside farmers can engage with the system, gather data, and get real-time feedback on crop health and disease status thanks to a user-friendly application.

2.2.1 Use Case Diagram 1



2.2.2 Use Case Diagram 2



2.3 User Characteristics

There are different user classes with different needs and technical abilities in the context of the crop disease detection and treatment system:

2.3.1 "Big Fun Agriculture" Personnel:

The "Big Fun Agriculture" administrative staff is in charge of monitoring the system and making sure it runs well. They have in-depth understanding of the organization's farming operations and a wealth of management expertise. Despite not necessarily being IT professionals, they have a strong grasp of the agriculture industry and are able to make tactical choices regarding the treatment of diseases.

2.3.2 External Farmers

External farmers using the system require fast access to information about crop health and treatment suggestions. Although they may not necessarily be IT specialists, they frequently have experience with data input and are familiar with computer programmes. Their technical proficiency may differ, and the system ought to have an intuitive user interface to support a variety of users.

2.3.3 System Administrators

The technical features of the system must be managed and maintained by system administrators. High technical competency is required of them, including expertise in database management, software configuration, and system monitoring. Their background should include administration and troubleshooting of IT systems.

2.3.4 Data Entry Personnel

Employees that enter data, particularly supply staff, are essential to preserving accurate data. They need to be able to enter data quickly and accurately, typically utilising data management programmes like Microsoft Access. They may not have a lot of IT experience, but they are skilled at data entry work.

2.3.5 Agricultural Experts

It's possible that "Big Fun Agriculture" agricultural professionals will participate in the decision-making process for managing diseases. They have extensive knowledge of crop health, illnesses, and pest management. They must analyze the output of the system and use their understanding of agriculture to make judgements that are well-informed.

2.4 Constraints

Non Disruption of farming operations: While switching from the old system to the new one, the system's implementation shouldn't interfere with or adversely affect ongoing farming operations. The amount of downtime or disruption to planting and harvesting cycles should be kept to a minimum.

Resource Constraints: "Big Fun Agriculture" wants the system to be maintained by a team of no more than three IT professionals. The complexity and maintenance needs of the system are constrained by this limitation, which calls for an effective and user-friendly design.

Training Constraints: Due to a lack of resources, the organization's workers and outside farmers should only receive training for a maximum of two days per month. The user interface and design of the system should be simple to use and require little end-user training.

Project Deadline: If the system misses the suggested deadline, the organization reserves the right to cancel the project at no additional cost. Due to the stringent time constraints imposed by this, effective project management and deadline adherence are essential.

Budget Constraints: The project's budget is constrained to no more than 200 million due to the organization's restricted resources. To keep the project commercially feasible, developers must operate within this budgetary restriction.

2.5 Assumptions and Dependencies

2.5.1 Database Management System:

The selected DBMS, in this case [Specify the DBMS], which will be available for data storage, retrieval, and management, is presumed to be used by the system to run.

2.5.2 Network Connectivity:

The system depends on user mobile being accessible and linked to a nearby intranet. This networked environment is required for safe access to the system's data and functioning, ensuring effective user-to-central database connectivity and data transmission.

2.5.3 Hardware and Security Card integration:

The system depends on issuing security cards to employees in order to enable secure user access and monitoring. The system also assumes that card readers have been added as necessary to PCs that require access control. This integration guarantees that security precautions are in place and that user access is limited to authorized persons.

3. Specific Requirements

3.1 External Interfaces

3.1.1 Integration with Weather Forecasting

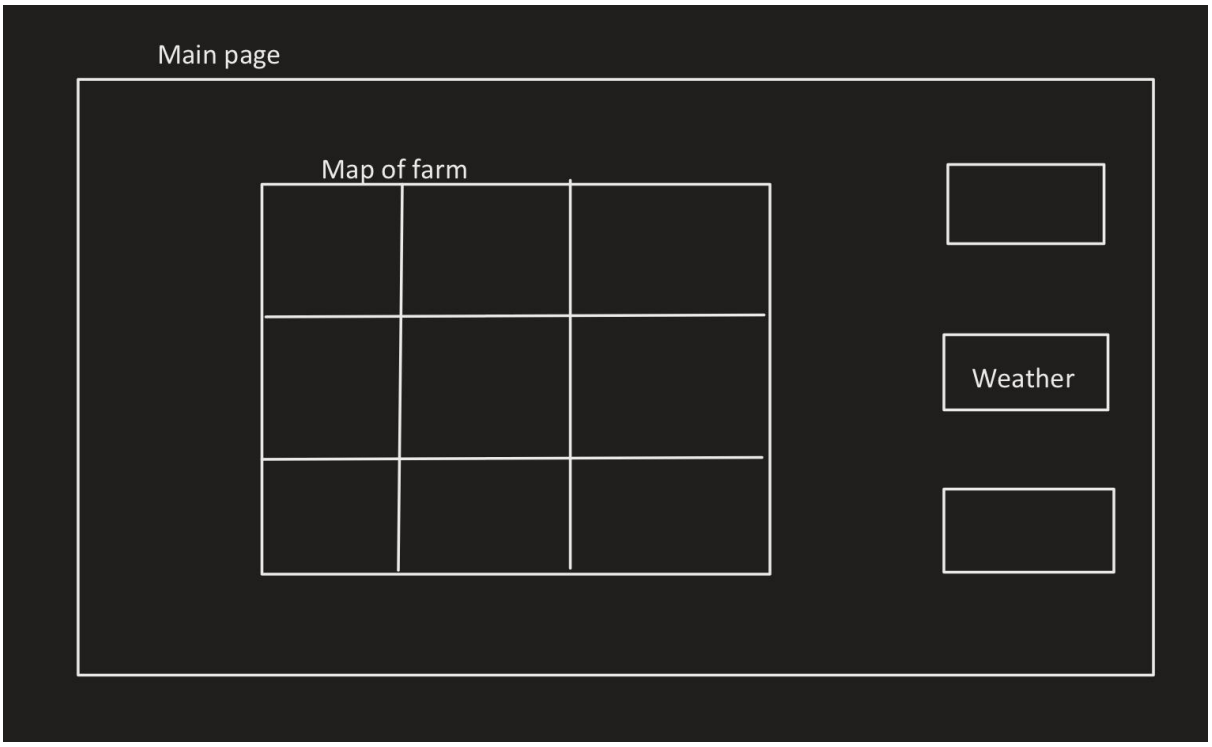


Fig1.0 A prototype of the system UI main page showing where weather would be displayed

a) Name of item: Weather Forecasting Integration

b) Description of purpose: The weather forecasting interface will allow farmers to enhance their farm management decision by having access to real time weather forecasting information.

c) Source of input or destination of output: Input is from external weather forecasting APIs/web services; output to the software's forecast algorithms.

d) Valid range, accuracy, and/or tolerance: Weather data accuracy will be within +/- 5% deviation from actual conditions.

e) Units of measure: Temperature (°C), Precipitation (mm), Humidity (%), UV Index.

f) Timing: Real-time updates every 15 minutes and with forecasts up to 7 days ahead.

g) Relationships to other inputs/outputs: Weather data affects farmer’s decisions related to irrigation, planting, and pest control.

h) Screen formats/organization: Weather data will be displayed in a dedicated section on the right side of the main page, organized by date and type of information.

i) Window formats/organization: It is a resizable weather widget that displays current and forecasted weather conditions.

j) Data formats: JSON format for API integration and easy user-friendly display for the end-user.

k) Command formats: N/A

l) End messages: N/A

3.1.2 Integration with Plant Diseases Library

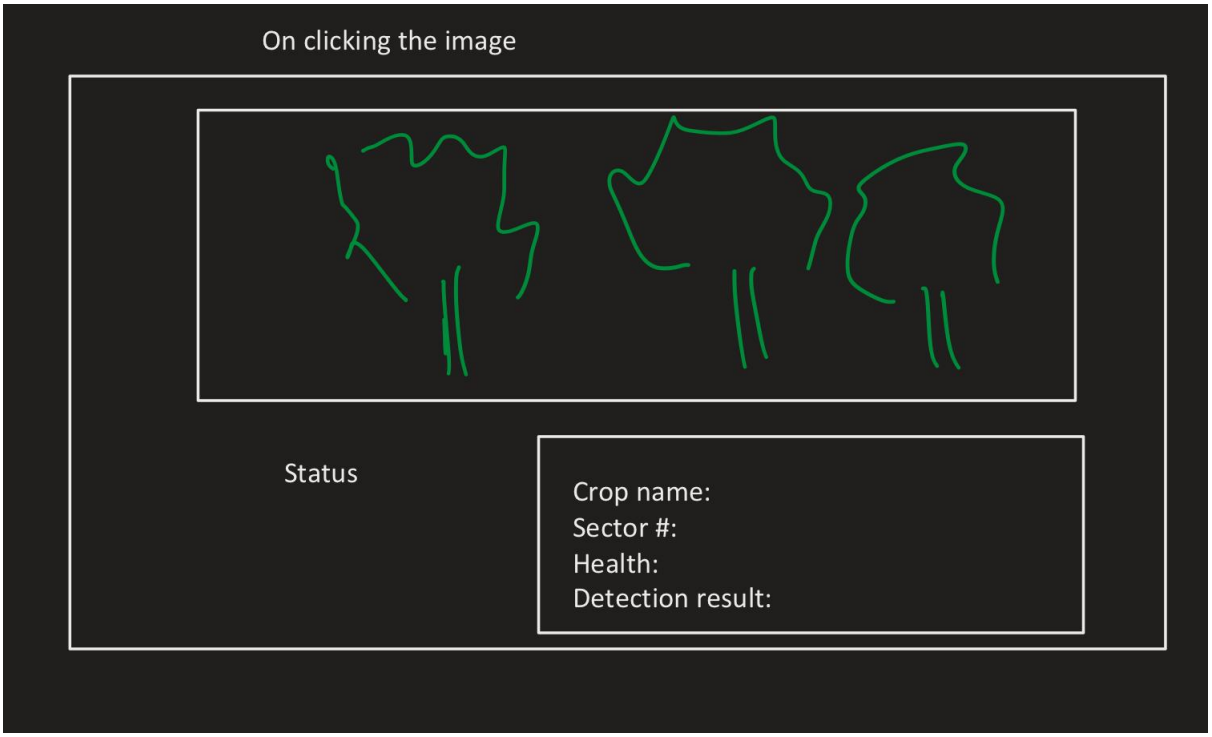


Fig1.2 A prototype of the system UI crop section page showing image and status information

a) Name of item: Plant Diseases Library Integration

b) Description of purpose: Plant Diseases Library interface helps the software to access a wide and comprehensive database of known plant diseases for accurate identification and treatment recommendations to farmers.

c) Source of input or destination of output: Input from the Plant Diseases database via API calls; output to the software's disease identification and detection result algorithms.

d) Valid range, accuracy, and/or tolerance: 95% accuracy in disease identification based on visual inputs given by the drones.

e) Units of measure: 1- 5 spectrum, 1 being healthy and 5 being the plant is beyond recovery

f) Timing: Instant response time for disease identification and detection results.

g) Relationships to other inputs/outputs: The detection result is critical for accurate diagnosis and treatment of the crop.

h) Screen formats/organization: Disease information is displayed in a structured format inside Status widget, categorized by type and symptoms in the detection result section.

i) Window formats/organization: The plant disease library will be accessible through a searchable interface within the software user interface.

j) Data formats: JSON format for API integration and structured data for display.

k) Command formats: N/A

l) End messages: N/A

3.1.3 Integration with Pesticide Inventory Systems

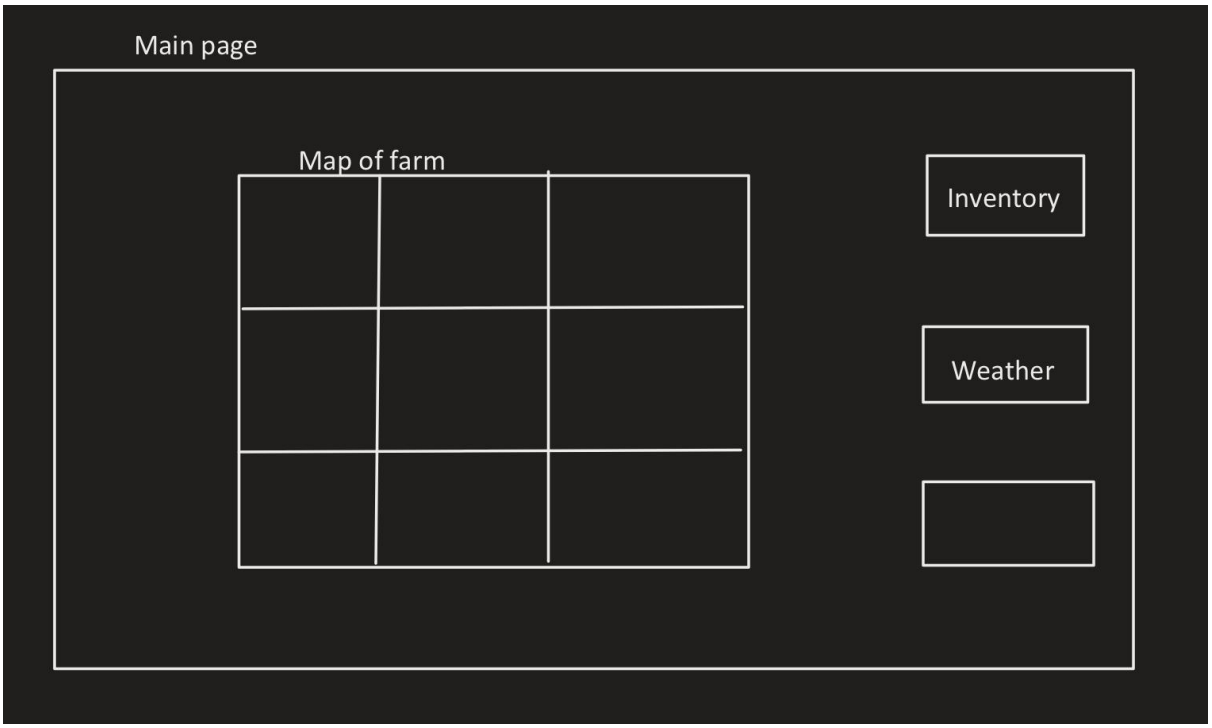


Fig1.0 A prototype of the system UI main page showing where Inventory would be displayed

a) Name of item: Pesticide Inventory Integration

b) Description of purpose: Pesticide Inventory interface connects the software with the farmer’s pesticide inventory system to gain access to information about pesticides that are available.

c) Source of input or destination of output: Input from the farmer’s pesticide inventory system via API calls; output to the software's treatment recommendation section/widget.

d) Valid range, accuracy, and/or tolerance: Accurate information on available pesticides in the inventory and their application rates on the crop.

e) Units of measure: Quantity (liters/gallons), Application Rate (liters/hectare).

f) Timing: Real-time updates on pesticide availability.

g) Relationships to other inputs/outputs: It is crucial for recommending the most effective and effective treatment for the diseases that have been identified.

h) Screen formats/organization: Pesticide inventory displayed in a widget on the right side of the main page, including product name, quantity, and application rates.

i) Window formats/organization: Accessible through the main mage of the software with the treatment recommendation interface.

j) Data formats: JSON format for API integration, organized format for display.

k) Command formats: N/A

l) End messages: N/A

3.1.4 Integration with Drone Systems

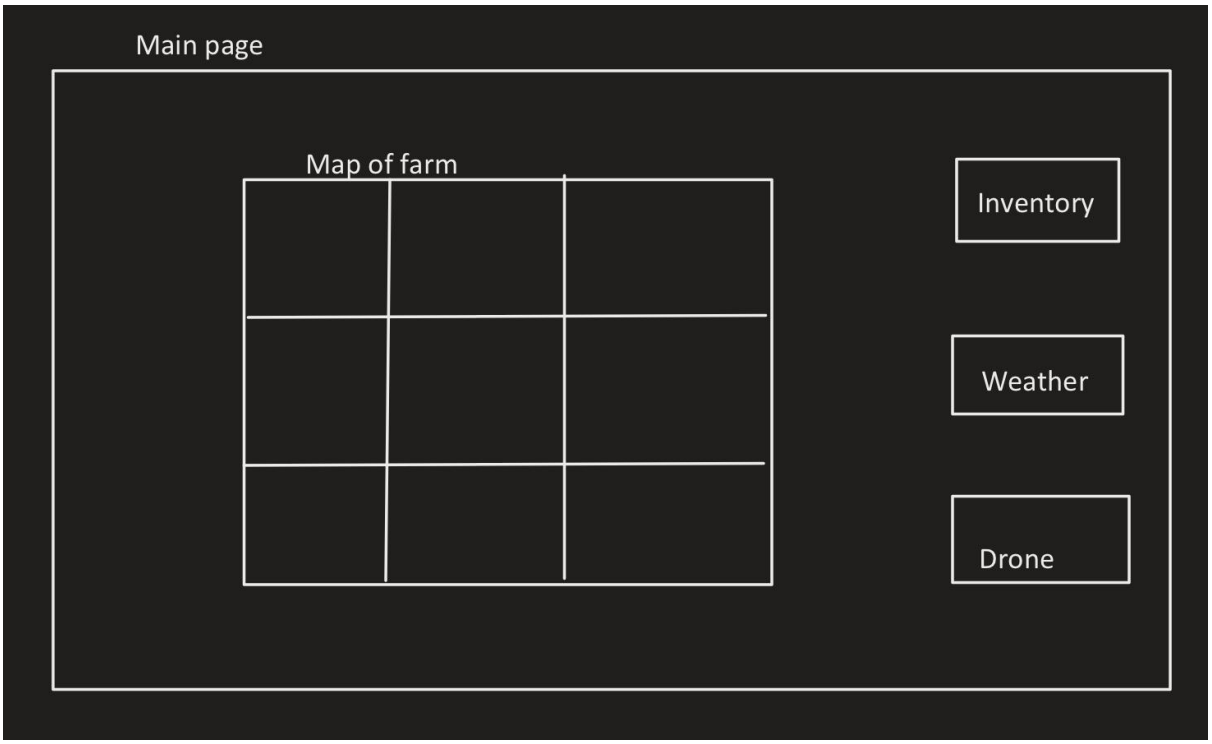


Fig1.0 A prototype of the system UI main page showing where Drone Status would be displayed

a) Name of item: Drone system Integration

b) Description of purpose: Drone system interface connects the software with the farmer's drone system to gain access to drones status.

c) Source of input or destination of output: Input from the farmer's drone system via API calls; output to the software's drone status section/widget.

d) Valid range, accuracy, and/or tolerance: Accurate information on Active drones on the field.

e) Units of measure: number of active drones and number of inactive drones

f) Timing: Real-time updates on drones status.

g) Relationships to other inputs/outputs: drone's data affects farmer's decisions related to irrigation, planting, and pest control..

h) Screen formats/organization: Drones status are displayed in a widget on the right side of the main page, including number of active and inactive drones.

i) Window formats/organization: Accessible through the main mage of the software with the Drone status interface.

j) Data formats: JSON format for API integration, organized format for display.

k) Command formats: N/A

l) End messages: N/A

3.2 Functions

3.2.1 Requirements

R1. The system shall receive images from the drone and analyze it in real time to identify diseases.
As discussed during the requirement elicitation session, analyzing the crop images for diseases in real time is extremely important.

R2. The system shall work on any windowsOS supported device.

R3. The main page of the system shall display a map of each crop field. The map shall have indicators for each part of the map to show the status of the crops and the indicators shall be color-coded as follows:
   a. Green shall indicate "Good" crop status.
   b. Yellow shall indicate "Unknown" crop status.
   c. Red shall indicate "Bad" crop status.
   As requested during the requirement elicitation session, having a status indicator on the main page would be helpful.

R4. The map shall be interactive, allowing users to click on image markers. Clicking on an image marker shall enlarge the image.The enlarged image shall display the following information:
   a. Name of the crop.
   b. Sector number.
   c. Health status.
   d. Detection results, including:
      i. Whether the crop is diseased or not.
      ii. If diseased, display the disease name (e.g., "DISEASE DETECTED - FUNGI").
      iii. Disease percentage (if <50%, indicating the crop is salvageable).
      iv. Remedy information.

   As requested during the requirement elicitation session, having an interactive GUI is necessary.

R5. The main page shall include three buttons for accessing information on:
   a. Inventory.
   b. Weather.
   c. Drone Status.

R6. The system shall notify the user if any disease is detected and suggest the user information on available remedies and inventory status. For example The system shall display alerts, such as "*Fungi has been detected! You can use x-pesticide for the treatment of this crop. You currently have 4 kg of this pesticide in inventory. For more details, click on the image from the map*."

R7. The application shall be designed for ease of use and beginner-friendliness.

R8. The system shall include an option for users to access a tutorial video for the application.

3.2.2 Use Cases

3.2.2.1 View Crop Status of a field

Name: View Crop Status of a field
Use Case Number: UC01
Authors: Ruth Bezabeh
Event: The customer navigates to the main page, no inputs are passed
System: The system updates the page with real-time analysis of data received from drones of all sections of the field.
Actors: Customer (initiator)
Drones
Overview: The user is able to view a Green, yellow or red status indicator on all sections of the field.
References: R1, R3, R7
Related Use Cases:
Typical Process Description:

Actor Action	System Responsibility
1. The customer opens the main page	
	2. The system retrieves the status information for each section of the field
	3. The user interface is updated to show the real-time status of all sectors

3.2.2.2 View Sector Information

Name: View Sector Information
Use Case Number: UC02
Authors: Ruth Bezabeh
Event: The customer navigates to the main page and clicks on one of the sections. The identifying sector number is passed.
System: The system analyzes the data received from the drones and checks against the disease library and pesticide inventory to determine status and recommend treatment
Actors: Customer (initiator)
Drones
Disease Library
Pesticide Inventory
Weather API

Overview: The system detects diseases and recommends treatment options. The user is able to view the Name of crop, health status, detection results and remedy information
References: R1, R3, R4, R6, R7
Related Use Cases: UC01
Typical Process Description:

Actor Action	System Responsibility
1. The customer clicks on one of the sections from the main page	
	2. The system retrieves the real-time data from drone image api
	3. The system analyzes the image, weather and soil status. See 3b.
	4 The system determines crop health status. See 4a Detects disease and 4b Does not detect disease
	4a The System Detects Disease
	5. The system determines degree of damage
	6. The system determines recommended treatment and checks inventory
	7. The system displays red for health status, Disease detected - <name of disease> for detection status,degree of damage and treatment remedy and inventory Level of pesticide.
	4b. The system doesn't detect disease
	5. The system displays green for health status and no disease detected for detection status
	3b. The System receives unavailable drone status can't analyze Image.
	4. The system displays yellow for health status and cannot analyze image for disease detection

3.2.2.3 View Weather Information

Name: View weather information
Use Case Number: UC03
Authors: Ruth Bezabeh
Event: The customer navigates to the main page and clicks on Weather. No input is passed.

System: The system retrieves weather information from the weather API
Actors: Customer (initiator)
Weather API
Overview: The system displays current and forecasted weather information.
References: R5, R7
Related Use Cases: UC01
Typical Process Description:

Actor Action	System Responsibility
1. The customer clicks on weather on the main page	
	2. The system retrieves current and forecasted weather from weather api
	3. The system updates the UI with the weather information

3.2.2.4 View Pesticide Inventory

Name: View Pesticide Inventory
Use Case Number: UC04
Authors: Ruth Bezabeh
Event: The customer navigates to the main page and clicks on Inventory. No input is passed.
System: The system retrieves weather information from Pesticide Inventory
Actors: Customer (initiator)
Pesticide Inventory
Overview: The system displays Inventory.
References: R5, R7
Related Use Cases: UC01
Typical Process Description:

Actor Action	System Responsibility
1. The customer clicks on Inventory on the main page	
	2. The system retrieves data from pesticide inventory
	3. The system updates the UI with the current pesticide inventory level

3.2.2.5 View Drone Status

Name: View Drone Status
Use Case Number: UC05
Authors: Ruth Bezabeh
Event: The customer navigates to the main page and clicks on Drone Status. No input is passed
System: The system retrieves weather information from the Drone Status API
Actors: Customer (initiator)
Drone Status API
Overview: The system displays current and forecasted weather information.
References: R5, R7
Related Use Cases: UC01
Typical Process Description:

Actor Action	System Responsibility
1. The customer clicks on Drone Status on the main page	
	2. The system retrieves the Drones status from the Drone status API
	3. The system updates the UI with the Drone status for all drones.

3.2.2.6 View System Notification

Name: View System Notification
Use Case Number: UC06
Authors: Ruth Bezabeh
Event: The customer navigates to the main page and clicks on Notifications. Notification id is passed.
System: The system sends a notification in the event that a disease is detected
Actors: Customer (initiator)
Overview: The system displays current and forecasted weather information.
References: R5, R7
Related Use Cases: UC01, UC02
Typical Process Description:

Actor Action	System Responsibility
1. The customer clicks on Notifications	
	2. The system Displays message, with the passed notification id, that

	includes the type of disease detected, treatment recommended and inventory level of said treatment

3.3 Performance Requirements

1. The application should support multiple users simultaneously.
2. The application should analyze incoming drone images in real-time, with a maximum delay of 2 seconds between image capture and analysis.
3. The interactive map should respond to user interactions (such as clicking on images) without significant lag. The user should experience minimal delays when interacting with the map.
4. When a user clicks on an image, the enlargement of the image and display of relevant information should occur within 2 seconds.
5. The response times for critical user interactions, such as accessing weather or drone status information, should be less than 3 seconds to ensure a smooth user experience.
6. Disease detection alerts should be displayed to the user immediately after detection, with a maximum delay of 5 seconds. These alerts should not disrupt the overall user experience.
7. Inventory status should be updated in real-time, so users have accurate information on pesticide or remedy availability. Any delay in updating the inventory should not exceed 5 seconds.
8. The application should be able to gracefully handle errors and exceptions, providing clear error messages to users, and ensuring that errors do not cause system crashes.
9. The application should have a connection with a database for efficient data storage, retrieval, and archiving to prevent performance bottlenecks due to excessive data accumulation.

3.4 Logical Database Requirements

3.4.1. Image Data

a) *Types of information used by various functions:* The images from the drone are used by Map functions to portray the crop image as well as used to do crop analysis.
b) *Frequency of use:* The image data is used in real time for analysis
c) *Accessing capabilities:* The data should be accessible by other functionalities
d) *Data entities and their relationships:* The images received from the drones should be stored, along with their associated data, including crop and disease information.
e) *Integrity constraints*: Images from the drone should not be lost.
f) *Data retention requirements*: Keep images for a limited time due to storage constraints

3.4.2. Crop Data

a) *Types of information used by various functions:* The crop data is used to determine the disease and its remedy.
b) *Frequency of use:* The crop data is used whenever a disease is detected
c) *Accessing capabilities:* The data should be accessible by other functionalities
d) *Data entities and their relationships:* It should store information about each crop, including its name, sector number, and health status.
e) *Integrity constraints*: Maintaining data consistency between crop data and disease data.
f) *Data retention requirements*: Storing historical disease detection results and crop health data for future analysis and reporting as well as defining data retention policies for historical crop and disease data to manage database size.

3.4.3. Disease Data

a) *Types of information used by various functions:* The Disease data such as disease name, remedies, and percentage thresholds is used by the Map function to present the details to the user. It is also used by the crop data to determine the disease.
b) *Frequency of use:* The disease data needs to be occasionally updated as new diseases and remedies are discovered or updated.
c) *Accessing capabilities:* The data should be accessible by other functionalities
d) *Data entities and their relationships:* It should maintain data related to diseases, including their names, remedies, and disease percentage thresholds.
e) *Integrity constraints*: Ensuring data consistency and accuracy through validation rules.
f) *Data retention requirements*: Defining data retention policies, especially for historical crop data, to ensure the system doesn't accumulate unnecessary data. Should archive older data if needed for historical analysis or regulatory compliance.

3.4.4. Inventory Data

a) *Types of information used by various functions:* The inventory data such as the Name of the remedy ,Quantity available in inventory is used by the notification function.
b) *Frequency of use:* The inventory data should be regularly updated as users consume or restock inventory.
c) *Accessing capabilities:* The data should be accessible by other functionalities
d) *Data entities and their relationships:* It should keep track of available inventory items and their quantities.
e) *Integrity constraints*: Implement data consistency checks to prevent invalid entries.
f) *Data retention requirements*: Keep the database up to date.

3.4.5. Map Data

a) *Types of information used by various functions:* The Map data is used to get an overview of the geospatial data representing crop fields and their statuses.
b) *Frequency of use:* The Map data is frequently updated to reflect crop statuses.
c) *Accessing capabilities:* The map data should be easily accessible.
d) *Data entities and their relationships:* The map data should store information about the map, including the location of crop fields and their associated statuses (Green, Yellow, Red).
e) *Integrity constraints*: Each crop entry in the "Crop Data" entity needs to be linked to a corresponding field in the "Map Data" entity.
f) *Data retention requirements*: Update the Map data as the image data changes.

3.5 Design Constraints

3.5.1. Platform support

- The application must run on any Windows Operating System supported device regardless of the OS version.

3.5.2. Processing Speed

- The application must analyze images in real-time. This functionality requires a system with sufficient processing power to handle image analysis quickly and efficiently. Slower hardware may lead to delays in disease identification.

3.5.3. Data Storage and Performance

- Storing and retrieving a large number of images and associated data for real-time analysis may pose constraints on database performance and storage capacity.

3.5.4. Internet Access

- Real-time analysis and access to external resources, like weather data, depend on a stable Internet connection. This could be a constraint for users in areas with poor connectivity.

3.5.5. Agricultural Regulations

- Compliance with regional or national agricultural regulations, particularly regarding the use of pesticides and the handling of agricultural data, may introduce constraints on the application's functionality and data handling.

3.5.6. Weather Conditions

- The application's performance may be influenced by environmental factors since drones are used for data collection. Adverse weather conditions could impact data collection and real-time analysis.

3.5.7. Project Schedule

- The development of the application may be subject to time constraints, such as deadlines for delivery or deployment.

3.6 Software System Quality Attributes

3.6.1 Security

- It is important that the software system integrate data encryption protocols in order to protect and provide security to the farm's sensitive data
- User authentication is needed in order to ensure that unauthorized users don't have access to the farm's sensitive data
- The implemented software needs to ensure that all the external policies and regulations imposed by the government about the security issues in the agricultural industry are followed without fail. The data about the farm's pesticide inventory needs to be protected as stated by the Agricultural Board of Association.
- The software needs to obtain an Agricultural Data Protection Act certificate and any other necessary data privacy protection certificates in order to comply with industry standards

3.6.2 Reliability

- The software system should be able to handle large traffic and high volumes of data without glitching and crashing. The performance of the software system should be intact in any scenario.
- Rigorous testing and bug detection should be done on the software in order to make it stable and reliable.
- If the software faces any system errors or failures there should be failover procedures to handle the situation.
- To keep the software updated and address reliability issues there should be scheduled maintenance on the software system.

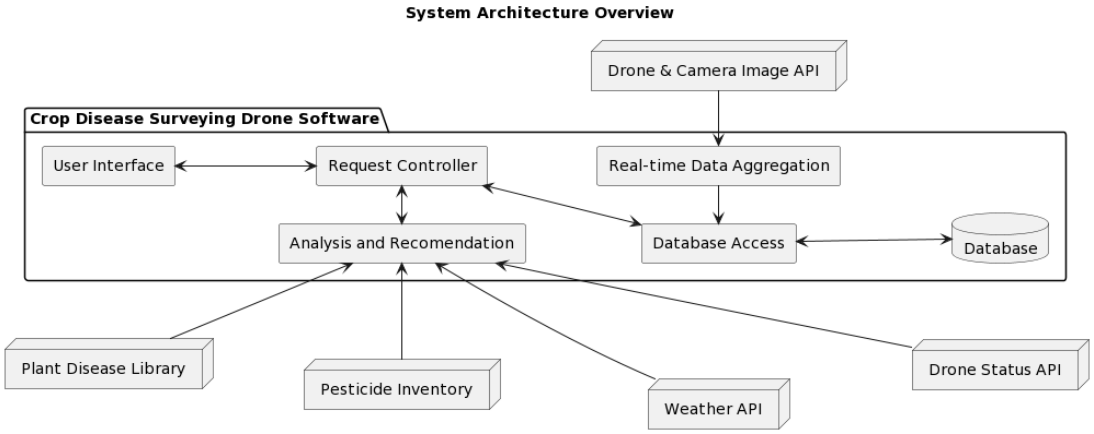
3.6.3 Availability

- In case of any downtime caused by the system failure, the system should have a procedure in place for checkpointing, recovery, and restart.
- The software system should integrate redundancy and data backup mechanisms in order to ensure that the system operations remain continuous even if there is any hardware or program failure.
- The system should have high availability in order to satisfy farmers' needs and also to meet high volume or crop farming operations

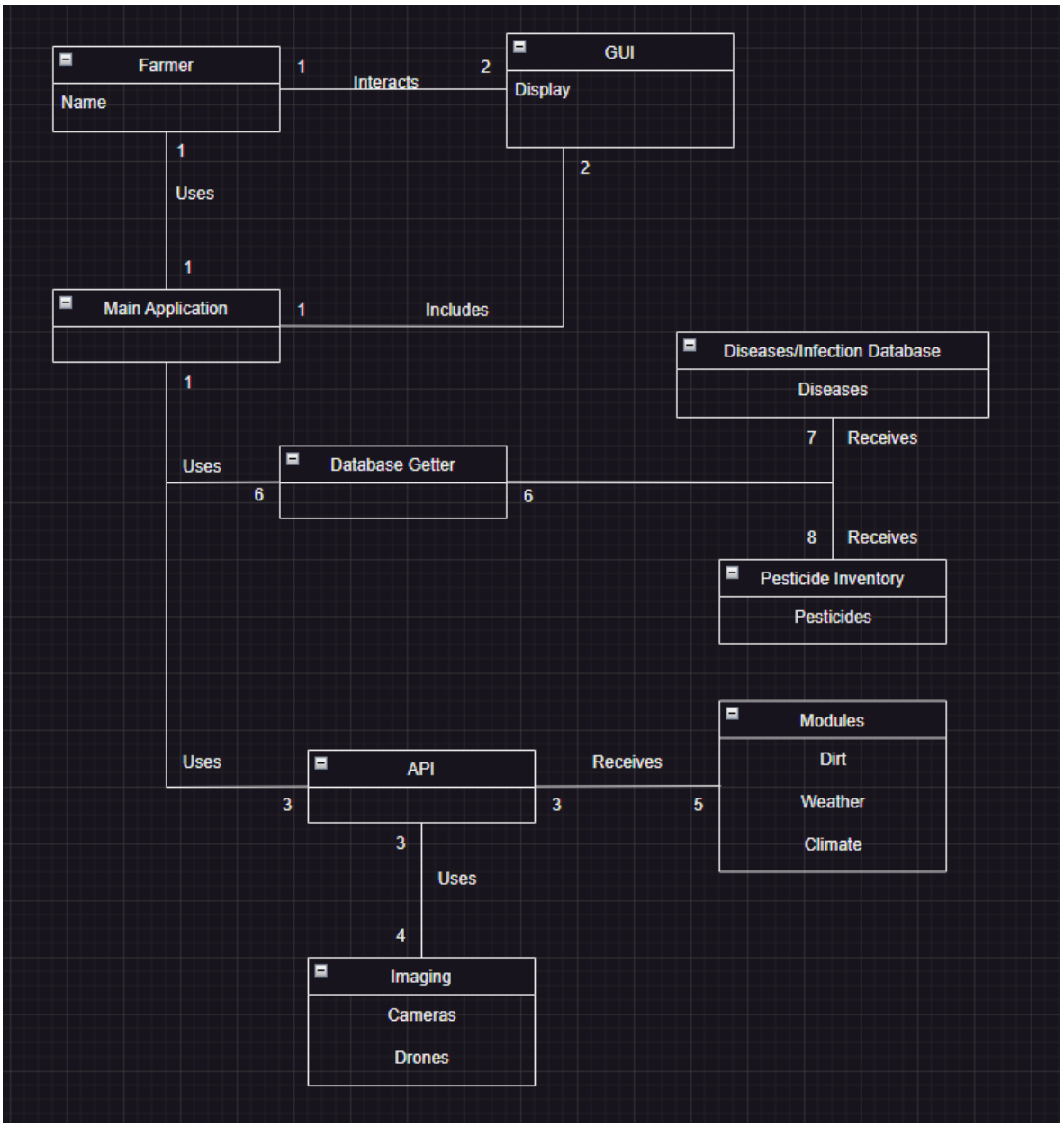
- The maintenance performed on the system to update the software should be scheduled during hours of low traffic in order to minimize disruption of service.

3.7 Object Oriented Models

3.7.1 Architecture Specification



3.7.2 Domain Model



4. Appendix

4.1 Sample Inputs & Outputs

The inputs for the application include information modules (ex: dirt), database information (ex: diseases) and the imaging software from drones and cameras. The output from the application will be information regarding the disease, which/how many plants are diseased and potential treatments/solutions, from which the user (farmer) can decide on how to proceed. For example, if crop A is infected with disease B, and can only be solved with a pesticide C, the application will first determine if crop A is indeed infected or is in poor health due to another reason (ex: low water in dirt, high pH in dirt, weather, etc). It will access the plant diseases library, match the corresponding symptoms with the disease, and then access

the pesticide inventory system to see what the possible solutions are. So to the user (farmer), the application will output that crop A is infected with disease B and can be cured by using a pesticide C (in regards to the goal of detecting and preventing development of disease).