ASHESI UNIVERSITY

SKAÏGREEN: AN AERIAL DRONE AND DEEP LEARNING PROJECT FOR THE COUNTING AND NITROGEN DEFICIENCY DETECTION OF COCONUT TREES.

APPLIED PROJECT

B.Sc. Computer Science

Emmanuel Nkunim Adu-Sarkodee & Michael Kwame Eshun

2023

ASHESI UNIVERSITY

SKAÏGREEN: AN AERIAL DRONE AND DEEP LEARNING PROJECT FOR THE COUNTING AND NITROGEN DEFICIENCY DETECTION OF COCONUT TREES.

APPLIED PROJECT

Applied Project submitted to the Department of Computer Science, Ashesi University, in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Science.

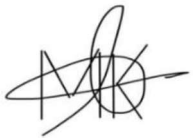
Emmanuel Nkunim Adu-Sarkodee & Michael Kwame Eshun

2023

DECLARATION

We hereby declare that this applied project is the result of our own original work and that no part of it has been presented for another degree at this university or elsewhere.

Candidates' Signatures:



Candidates' Names: Michael Kwame Eshun, Emmanuel Nkunim Adu-Sarkodee

Date: May 12, 2023

I hereby declare that the preparation and presentation of this applied project were supervised in accordance with the guidelines on supervision of applied projects laid down by Ashesi University.

Supervisor's Signature:

Supervisor's Name: Ayawoa Sitsope Dagbovie, PhD

Date: May 12, 2023

ACKNOWLEDGEMENT

First, we would like to give honor to whom honor is due and acknowledge God Almighty, who was with us and helped us from the ideation stage of this capstone project to its final testing stage. All glory goes to Him. We are also grateful for the support, guidance, and constructive feedback from our supervisor, Dr. Ayawoa Dagbovie. Her expertise and guidance were instrumental to the success of this project. I, Emmanuel Nkunim Adu-Sarkodee, would like to express my gratitude to my teammate, Michael Eshun, for his deep commitment, diligence, and hard work on this project. Aspects of this project (the engineering of the drone and the training of the deep learning model for detection and counting), are attributed to Michael, and I acknowledge him for that. I, Michael Kwame Eshun, would like to thank and appreciate Emmanuel for his immense pursuance and diligence in this project. Aspects including the frontend and the detection of nitrogen deficient coconut trees were executed by Emmanuel. Finally, we would like to thank our friends and peers who provided us with technical support and encouragement throughout this project. Their contributions were significant factors in the successful completion of this capstone project.

ABSTRACT

This capstone project involved the implementation of a drone system to obtain an aerial view of a coconut farm, with the goal of using deep learning models to detect and count coconut plants, as well as identify nitrogen-deficient plants. The YOLOv5 algorithm was used for object detection, with a unique ID assigned to each plant for counting. A VGGNet convolutional neural network was trained on a dataset of labeled coconut plant images to classify plants as "healthy" or "unhealthy". The results of the project included an automated drone system that could fly and capture footage with a single click, allowing for efficient counting of coconut plants and detection of nitrogen-deficient plants.

Key Words: Drone, YOLOv5, Unique ID, Nitrogen deficiency, Coconut plants, Convolutional Neural Network.

Table of Contents

Chapter 1: Introduction

1.1 Background

The Ghanaian economy's reliance on the agricultural sector cannot be overlooked. In 2018, Ghana was the second largest producer of plantain worldwide, only behind the Democratic Republic of Congo [1]. Other notable crops (GDP-wise) produced by Ghanaian farmers include pineapple, cassava, and beans. Even though attempts are being made to industrialize the Ghanaian economy, there would still be the need for the primary producers to provide inputs for the secondary sector's use. Therefore, the agricultural sector in Ghana is still very relevant and will remain as such, even when large- scale industrialization occurs in Ghana.

Given the importance of agriculture to Ghana's economy, what steps can the country take to optimize the benefits of its agricultural industry before it becomes a fully industrialized nation? To achieve this, it would be necessary to identify and analyze each factor that supports the agricultural sector. This pertains to the farmer, the crops, as well as the physical surroundings of the other two components. In the 21st century, technology plays a significant role in most aspects of life. Therefore, ways in which technology could improve farming in Ghana would be a good starting point to enhance the agricultural sector.

Drone technology has evolved from being a concept in the early 2000s [2] to being a fully-fledged technology used in many sectors, including healthcare, construction, surveying, and mapping. The possibility of drone technology being used to improve farming could also be looked at. For example, if a farmer needed to monitor their crops from a remote location, they could use a drone to observe the progress of their crops. Also, if the drone detected wilting in a particular crop, a sensor could be triggered to activate installed sprinklers to water the plants [3]. Find the code for this project here.

1.2 The Problem

For this applied project, Skaïgreen contacted a farmer to identify a pertinent problem on their farm that could be solved. The farmer highlighted the challenge of obtaining an aerial view of his coconut farm to monitor crop progress and make informed decisions. By addressing this problem, the farmer could obtain a comprehensive overview of his crops' progress and make timely decisions accordingly. The farm currently has a size of 14.8 acres (approximately 5.99 hectares) or 11 football fields.

Due to the large size of the farm, the use of drone technology was suggested as a potential solution due to its ability to provide real-time updates and allow the farmer to monitor crops remotely and detect any issues promptly. This technology could help optimize farming practices and ultimately increase crop yield.

In addition to the coconut farmer, interviews were conducted with other farmers who cultivated different crops. Based on the findings collected (see Appendix A for interview questions), it was observed that farmers typically employed caretakers to manage their crops. However, this approach had some drawbacks. For example, inspecting a large farm manually could be time-consuming and challenging, leading to the possibility of overlooking areas with nutrient-deficient plants. Also, estimating crop yields by manually counting crops was also a tedious and difficult task. Furthermore, farm caretakers could apply insecticides or fertilizers to areas that did not require them, thus wasting resources. Insecticides may be applied across the entire farm because pinpointing the areas that need treatment could be challenging. Therefore, applying insecticides generally may be perceived as a simpler approach.

To automate these processes, it would be necessary to use drone technology to capture high-resolution images of the farm and analyze them using machine learning algorithms [4].

This can provide accurate information on crop growth and health, estimate yield projections, and identify areas requiring fertilizers or insecticides. Such automation can significantly reduce the laborious nature of crop management and save time and resources. In summary here are the problems associated with conventional crop viewing to track a farm's progress:

- May take a longer than expected amount of time to complete.
- Certain areas of the farm could be overlooked.
- The amount of manual labor the farm caretaker does may be too arduous.
- The tasking nature of counting the crops manually.
- All the issues mentioned above are intensified when the farm is relatively larger in size.
- Wasteful insecticide/ fertilizer application to areas of the farm that do not require them.

1.3 Motivation

The role of farmers in Ghana's agricultural sector is critical to the economy's success (see Section 1.1.). To encourage them to produce high-quality and high-yield crops, they need support. Therefore, the primary motivation would be to determine the best ways to support farmers and incentivize them to produce their best crops.

For instance, the research conducted by Silwal et al. [5] aimed to assist apple farmers with harvesting using a low-cost robotic apple harvester. They tested the technology in a commercial apple orchard and reported on workspace adjustments and performance criteria to guide future improvements. The system the apple harvesters adopted led to an improved average localization time of 1.5 seconds per fruit, with an 84% success rate, and an average picking time of 6 seconds per fruit. This suggests that the automation of the manual efforts involved in successful apple harvesting could yield higher performance in terms of the number of apples

harvested [5]. Similarly, the use of an automated drone system to view the entire coconut farm could supplement the farmer's manual labor efforts and yield better results.

In addition to facilitating crop yield increase for farmers, another motivation behind the development of a drone automated system with machine learning algorithms is the potential to drive technological innovation and progress in the agricultural industry. This is because of how farm resources like land are reducing quantity and quality-wise. To clarify, Goedde et. Al published an article which reported that, while the demand for food is increasing, the supply side is encountering limitations in terms of land and farming resources [6]. Therefore, the addition of technology in diverse forms to improve food quality and increase crop yield would contribute to the eradication of food insecurity currently being faced in certain parts of the world. In this regard, the utilization of drone technology in this project is poised to make a substantial contribution to the technological advancement of the agricultural sector.

1.4 Related work & Existing Solutions

Field Spectroscopy to Determine Nutritive Value (NV) Parameters of Individual Ryegrass Plants [7]

A field spectrometer was used to collect the canopy spectra of distinct ryegrass plants. In this regard, the field spectrometer was used to create forecasting models for eight NV parameters for breeding programs. The targeted NV parameters included acid detergent fiber, ash, crude protein, dry matter, in-vivo dry matter digestibility, amongst others. Regression with partial least squares was used to create the models.

Similarly, field spectroscopy can break down each element of the farm's various components. Regression with partial least squares could then be used to create a model for each farm parameter, including crop leaf color (to detect whether watering is needed/ nitrogen

deficient plants), soil texture, animals that may feed on the crops, the farmer themself, and other nutritive value parameters. It is then possible for a drone to capture footage of the farm, classify its components into specific parameters, and use regression by the least square to learn each classification, as seen in [7].

Implementation of Drone Technology for Farm Monitoring & Pesticide Spraying [8]

In this paper, the authors conducted a comparative analysis of conventional methods of crop monitoring in relation to pest or disease detection. This was then compared with drone-automated crop monitoring. They discovered that farmers manually observed crops to look for any potential threats, including diseases, pests, and slow growth. This required visual examination and manual ground sample collection from several sites. However, drone technology introduced color and infrared photography which was important for monitoring crop growth. According to their research, the authors concluded that the primary uses of drones in the agricultural industry were for crop monitoring and field mapping (Figure 1.4). Similarly, drone-automated crop monitoring could be adopted on the coconut farm. Conventional methods that farmers normally use include manual observation and ground sample collection to identify potential threats to crops such as diseases and pests. However, with the adoption of drone-usage on the farm, advanced image data analytical techniques to identify crops with diseases through color and infrared photography could be garnered. From Figure 1.4 (b) below, each crop was identified and marked in green [8]. Based on this, the counting of crops could be facilitated through first spotting all plants, and then performing a count on them. Figure 1.4 (a), also shows a field map of the farm, with the section requiring attention highlighted at the topmost segment of the image highlighted in red. Both points derived from Figure 1.4 would be valuable contributions to the main aim of this project, which is to detect nitrogen deficient plants, and provide a count of all the coconut plants on the farm.

(a) (b)

Figure 1.4: Unmanned Aerial Vehicle application in agricultural field (a) Field mapping, (b)Crop monitoring

YOLO Fish Detection with Euclidean Tracking in Fish Farms [9]

In this article, a method was introduced to improve fish detection and fish trajectories in challenging water conditions. Firstly, an Image Enhancement algorithm was applied to improve unclear images. Then, an object Detection algorithm was utilized to detect fish in the enhanced images. This process allowed features such as the number of fish and trajectories to be extracted from the coordinates of the detected objects. To enhance the image, the authors introduced a technique for improving unclear water images/videos using the MSR (multi-scale retinex) algorithm. This technique involved applying a Single-Scale Retinex (SSR) filter to the image and then a MSR filter to enhance it further. The resulting images were clearer, making them useful for detecting and tracking fish [9]. As relates to the YOLO object detection, IDs were assigned to detected fish across video frames. The fish were tracked for 8 frames and the Euclidean distance was used to determine if a detected fish is the same as in previous frames. A new ID was assigned if the distance was greater than 50 pixels, and a border drawn around the fish for a visual representation. In relation to the coconut farm, the image captured by the drone could be enhanced by using the MSR algorithm to apply a filter on the image. The YOLO algorithm could be applied to count crops by assigning unique labels/IDs to each plant detected

in the images. The algorithm can then iterate through each ID and store them in a unique variable, ultimately counting the crops upon each ID iteration.

Drone-Based Data Solutions for Cereal Crops [10]

The author proposed a drone-based data solution as a way of suppressing food insecurity. It was concluded that, drones have a wide spectrum of applications in agriculture. Their applications range from crop monitoring, weed and pest management, fertilizer and weed-stress assessment, water stress assessment, among others. Regarding crop monitoring, the authors experimented the use of drones for the growth and health monitoring of cereal crops using various criteria ranging from height of crops to different vegetation indices. Normalized Difference Vegetation Index (NDVI) and Red-Edge NDVI indices were used to study the maize crops [10]. In relation to water-stress assessment, the authors used drones to assess the level of water-stress on farmlands. The use of multispectral drone images provided data for water-stress assessment on the maize field. For weeds and disease detection, a semi-automatic object-based image procedure together with random forest algorithm were used to classify soil, weed and maize. Likewise, Normalized Difference Vegetation Indexes could be used to map out healthy and infected areas on the coconut farm, by studying features of the coconut plants.

A Review on the use of Drones for Precision Agriculture [11]

In this paper, the author discussed the application of drone technology for precision agriculture. Semi-autonomous drones were employed in precision farming as well as disaster relief or traffic monitoring. Semi-autonomous drones are those that can follow predetermined flight courses based on waypoints and altitude information. The waypoints were determined by a positioning measurement system such as Global Navigation Satellite System (GNSS). To fly at constant altitudes, an altimeter was embedded on the drone. An example of an altimeter

is a barometer or, better still, an ultrasonic sensor. The drone was then equipped with a variety of cameras, including a multispectral camera and a red-green-blue (RGB) camera. By utilizing Near Infrared (NIR) imaging, both methods were used to keep tabs on a crop's vitals and detect any problems early on. Therefore, the coconut farmer could save time by using drones equipped with GNSS functionality (to aid in mapping out the farm), and NIR imaging to spot any issues with crops.

1.5 Proposed solution - Skaïgreen

The proposed solution is the implementation of a system that utilizes drone technology to obtain an aerial perspective of the entire coconut farm. The primary objectives of this system are to detect and count the number of coconut plants present, as well as identify those which are deficient in nitrogen. It is worth noting that nitrogen-deficient plants are identifiable by their yellow leaf color [8]. For reference, please refer to Figure 1.5, which illustrates the distinction between a healthy coconut plant and one that is nitrogen deficient.

Figure 1.5: Nitrogen deficient coconut plant (a) Healthy coconut plant(b)

This proposed solution involves the utilization of advanced deep learning models and convolutional neural networks to process the captured aerial footage and detect specific

features of interest. Specifically, the YOLO version 5 object detection algorithm will be implemented to identify coconut plants and assign unique IDs for accurate counting. Additionally, this algorithm will be used to differentiate between healthy coconut plants and those that are nitrogen deficient.

To facilitate the farmer's access to the captured footage and enable them to conduct a count of the coconut plants and detect nitrogen-deficient ones, a website application will be developed. Further details about the design and functionality of this website application will be discussed in Chapter 3.

Chapter 2: Requirements Analysis

2.1 Introduction

In this section, the requirements analysis will be developed, which will focus on determining the procedures necessary to establish the feature specifications of the system to be built. To achieve this goal, the main activity would be to obtaining and analyze information from stakeholders using various methods. Interviews were the primary means of extracting information due to the depth of information they provide, which is necessary for the proper development of both functional and non-functional requirements of the Skaïgreen system. In conducting this requirement analysis, each stakeholder was taken into consideration. After identifying the stakeholders and interviewing them, answers obtained from asking the interview questions were analyzed and framed into functional requirements. The following summarizes the procedure for requirements gathering and analysis:

- Identification and description of stakeholders (famers and farm owners)
- Generation of separate interview questions meant for both stakeholder parties.
- Conduction of the interviews on stakeholders
- Using interview responses to generate functional requirements.

2.1.1 Stakeholders and Description

The main stakeholders identified are grouped into two primary categories: the farmer and the farm owner. The farmer in this context represents the caretaker of the farm who performs most of the manual labor (for example, weeding, the spraying of insecticide on plants amongst others). The farm owner is the individual who puts in the resources to set up the farm, employs the farm caretaker, and decides on the usage of profits the farm makes. This distinction

between both parties is vital because, they each have their unique roles they perform to contribute to the farm's progress in various ways. These unique roles could then inform the building of the system, hence the functional and non-functional requirements.

For example, the farm caretaker would give information on the nutrient deficiencies the crops grown may have, and certain techniques he or she may use to detect such diseases from the onset. This could inform the detection techniques the algorithm would use to detect nutrient deficient crops. Au contraire, information obtained from the farm owner would give details on the logistic, monetary, and other technical details associated with the farm. Such information could be inferred from to inform for instance, appropriate flight times of the drone, and the user interface the drone would run on.

2.2 Requirement Analysis Aim

To re-iterate, the goal of gathering and analyzing requirements for the Skaïgreen system was to identify and understand the necessary specifications tailored to the needs of both farmers and farm owners. These requirements would be used to make informed front end choices that will facilitate a smooth user experience for both farmers and farm owners.

Data was gathered by conducting interviews with both farmers and farm owners to obtain information about their specific needs. Specifically, the coconut farm owner and caretaker were interviewed to gather insights into the crops grown, indicators of crop attacks or deficiencies, maturity periods, and other relevant details (Refer to Appendix A for interview questions). Additionally, farm owners provided information on the farm's size and specific activities that took place on the farm.

2.2.1 Requirements Gathering – Introduction

The data on requirements were gathered through interviews conducted with farmers from various farms. Selective sampling was used to obtain information from specific users, whereby farm owners were selected based on their specific criteria of being owners of well-mapped farms. This criterion of farms was important to allow the YOLOv5 algorithm to easily identify and map out plants (see Section 4.1.2). This contradicted a random selection of multiple farm owners. Detailed information was obtained through interviews with these farmers to gain in-depth information. Functional requirements were based on responses obtained from the interviews. Specifically, five farmers were interviewed using selective sampling to better understand the challenges they face on the farm. Based on the interviews, a client with a coconut farm was identified, whose responses formed the functional requirement of the Skaïgreen system.

2.2.2 Requirements Gathering – Interview Insights

From conducting interviews on both parties (the farmer and the farm owner) two main insights were obtained. A question that was asked the farmers was "*In relation to the farm, what kinds of problems do you face*"? A major theme from the responses was the issue of pest/ disease attacks on the farm (Kindly refer to Appendix A below). They mentioned that they usually saw the effects of the pest activities on their crops, long after they could do anything to stop them. As a preventive measure, some farmers mentioned that some pests were seasonal (that is, they begin their activities during for instance, the rainy season), so they spray the insecticide/ pesticide on the crops before that season begins. This informed the addition of the functional requirement of detecting signs of pest attacks or nutrient deficiencies on the plants, long before their destructive properties spread across multiple crops.

Another key insight gained from the farmers was the attack of animals unto their crops. Whether it was a herd of cows treading on their crops and destroying them, or a billow of goats

eating their crops, the farmers only saw the damaging effects on the crops. This information therefore shows the need for an alarm system of a sort, that would notify the farmer of animal presence and allow for necessary action to be taken.

A question that the farm owners answered was "*How would you describe what the main contributors to losses being incurred from farming activities are*" (Appendix A)? A major theme from the responses obtained was the lack of technical know-how, either on their part or on the farmer's part. This implies that the system implementation needs to be customized to meet the specific needs of the farmer and the farm owner, while also being technologically compatible for both parties.

In addition to the general information obtained from farmers and farm owners who were not the primary coconut farmer, the client (coconut farmer) emphasized the significance of segregating the processes on the web application. For example, the function for uploading pictures to detect Nitrogen-deficient crops should be a distinct process from the function for counting coconut trees. The rationale behind this is to ensure that the failure of one process, such as the upload for counting, does not impact the other process.

2.3 Functional Requirements - Introduction

In the functional requirements section, the primary goal is to develop a system that improves the activities of the coconut farmer, integrating smoothly with their current operations. Skaïgreen seeks to achieve this objective by deriving functional requirements from interview insights and on-ground research on the farm. The primary function of Skaïgreen is for the drone to fly autonomously, triggered by a button click on the front-end, to capture footage of the farm. This footage would enable the machine learning algorithms to accurately count the coconut trees on the farm and detect any Nitrogen-deficient plants.

2.3.1 The functional requirements for the system:

- **[FR01]** User registration: - On the sign-up page, users (farm owners) should be able to enter their valid credentials to gain access to the Skaïgreen system.
- **[FR02]** User Login: - Already registered users should be able to gain access to the Skaïgreen platform.
- **[FR03]** Drone Manual/ System help: - There must be a help icon on the Skaïgreen web app, that links to the drone specifications being used in the system.
- **[FR04]** Autonomous flying of the drone: At the click of a button on the frontend, the drone should fly at specific times to take video footage or images of the farm, for the image processing algorithms to give feedback to the user, based on specific sightings.
- **[FR05]** Count the number of coconut crops: - Using YOLOv5, the coconut plants on the farm would be detected, and a unique ID would be assigned to each plant to facilitate the counting process.
- **[FR06]** Detect Nitrogen deficient plants – Using YOLOv5 a bounding box should be used differentiate between healthy plants and those lacking in Nitrogen.
- **[FR07]** Provide a bird's eye view of the farm – Using the drone, pictures of the farm should be taken to provide the user with a clear overview of the farm.
- **[FR08]** Confirming that drone footage captured was saved: A JavaScript popup box is shows if the drone footage is successfully saved on the user's computer, and an error message displayed if it is not.

2.3.2 Non-functional requirements:

- **[NFR01]** Reliability: - There should be consistent availability of the web app for users' use. Also, in case a malfunction occurs on one side, it should not impair the entire functioning of the system.

- **[NFR02]** Security- The user's login and registration password should be encrypted in the database, such that it not easily accessible by outside parties.
- **[NFR03]** Maintainability –Updates to the web app or drone should be applied as and
- when the need arises.
- **[NFR04]** Correctness – There must be full functionality as stated in the functional
- requirements.
- **NFR05]** Robustness –If the user were to input invalid data, there should be a prompt
- to notify the user to enter the right details.
- **[NFR06]** Compatibility – Use of the web app should be applicable to any operating system or web browser and should cater for the coconut farm where diseases are detectable by the drones.
- **[NFR07]** Responsiveness – There should be an efficient provision of a seamless user experience, whereby on the web app, users get expected results upon performing clicking and dragging actions.
- **[NFR08]** Usability – The Skaïgreen web application should be usable for multiple users. Aside from, that, the user interface should provide a seamless user experience for users.

2.4 Use Cases and Diagrams

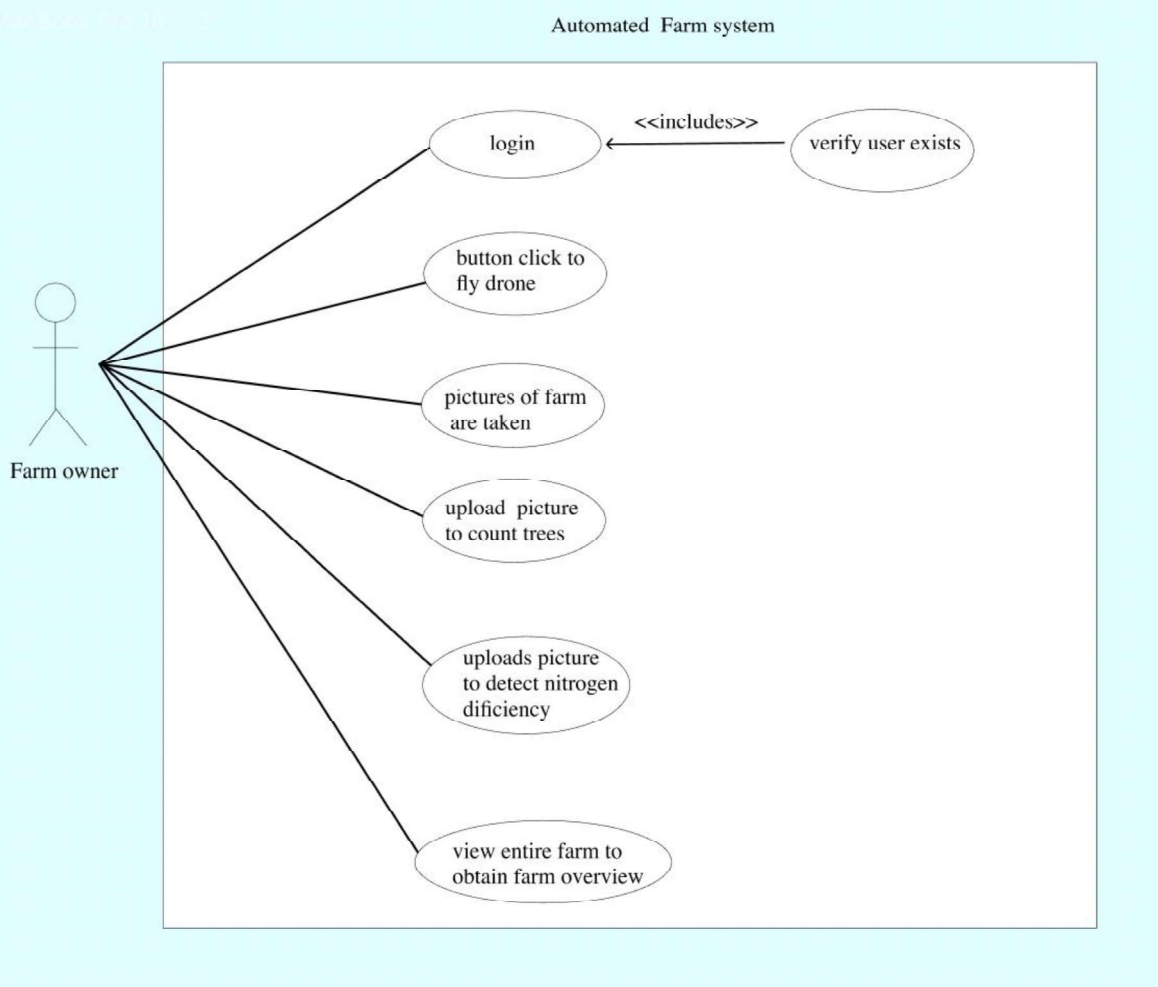


Figure 23.9 A use case diagram describing the process a user goes through from logging in to viewing the farm.

In the use case diagram above, the process a farm owner goes through in using the Skaïgreen web application can be seen. Each section of the use case diagram is used to reflect a specific requirement, as described previously in Section 2.3.1. The login use case can be attributed to the nonfunctional requirement of usability, which would ensure that multiple users can log into the system to use it.

Secondly, the “button click to fly drone” use case can be attributed to the functional requirement of autonomously flying the drone. This would be to solve the problem of having

a bird's eye view of the farm, through which the YOLO machine learning capabilities can be seen.

Also, the "pictures of farm are taken" use case can be attributed to the functional requirement of counting the number of crops and detecting nitrogen deficient trees. Although this use case is not explicitly stated as a functional requirement, it would be a necessary step for the user to take whilst the drone is in flight. This would feed into the next step for the YOLO machine learning algorithms to detect the coconut trees from the images taken.

Again, the "upload picture to count trees" use case can be attributed to the functional requirement of counting the number of coconut crops.

Fifthly, the "uploads picture to detect nitrogen deficiency" use case can be attributed to the functional requirement of detecting nitrogen deficient plants.

Lastly, the "view entire farm to obtain farm overview" use case can be attributed to the functional requirement of providing a bird's eye view of the farm. This would be to give the user an accurate view of the farm. This was implemented by allowing the drone to fly at a height of 15 feet above the coconut trees. After conducting experiments on the farm, this height was found to be optimal, providing a view that was not too low to the ground and localized to only a few coconut trees, nor too high that the coconut trees were out of sight.

Chapter 3: Architecture and Design

3.1 Introduction

This chapter highlights the system architecture implemented by Skaïgreen. The web application utilizes a three-tier architecture.

3.1.1 Three Tier Architecture

The presentation tier, application tier, and data tier make up the three-tier system architecture. The user's interaction with the web application falls under the presentation tier. The Skaïgreen system enables cross-browser interaction with the web application. The user can easily retrieve footage from the drone through the Skaïgreen web application. With the click of a button on the web application, the drone would be configured to fly autonomously.

The Application or logic layer manipulates data sourced to the presentation tier. For the drone to fly autonomously at the click of a button via the web application, Python code controls the automation of the drone flight. The drone flies automatically given the coordinates of the farm. Resources used in this tier include Python and PHP.

The data tier collects and stores data. The primary resource used by Skaïgreen for storing data is the MySQL Database. Data stored in the data tier is retrieved and displayed in the presentation tier for viewing by the user (data here refers to captured images). The data is also utilized by the logic tier for training and testing the machine learning algorithms used for detecting and counting the coconut trees and Nitrogen-deficient plants.

3.1.2 Tools, Technologies, Experimental Setup and Computational Model

The tools that will be integrated to build the Skaïgreen system are categorized into two main parts: frontend tools and backend tools. Each category will be fully represented below.

3.1.3 Frontend Tools

The following technologies will be used for the frontend development of the Skaïgreen system: HTML, CSS, And Figma. These tools will present data in a coherent fashion for easy user interaction.

- HTML (Hypertext Markup Language) is a valuable tool that enables the presentation of the structure of data on a web page to users. With HTML, individual pieces of information can be presented in a coherent manner for a specific purpose. For instance, in the implementation of the dashboard element of the Skaïgreen system, HTML code was utilized. By using the <li> tag in HTML, a list was created, forming the menu options available in the Skaïgreen dashboard. The use of HTML aligns with the fulfillment of NFR08, which underscores the importance of usability by providing a seamless user interface.
- CSS (Cascading Style Sheets) is a frontend tool that enables users to style the web page according to design requirements. HTML and CSS are closely linked tools, and in the previous example, the HTML <li> tag that formed the dashboard menu items needed to be styled using CSS. Specifically, the color of the menu items can be styled in a way that the menu box of a specific item changes color upon hovering. The use of CSS aligns with the fulfillment of the non-functional requirement of usability, which emphasizes that the user interface should provide a smooth and hassle-free experience for users (NFR08).

- Figma is a robust cloud-based design system that enables users to create design templates, construct wireframes, and export code [9]. In the context of the project, Figma was utilized to generate a wireframe template of the expected frontend web design after implementation with CSS and HTML. Additionally, it provided a means to test certain frontend user requirements with the farm owner without the need for code implementation. The use of Figma aligns with the fulfillment of the non-functional requirement of responsiveness, whereby the web application is expected to provide the intended outcome when users perform clicking and dragging actions (NFR07).

3.1.4 Backend Tools

Below are following technologies that will be employed in the backend development of the Skaïgreen system: these tools ensured the efficiency and smooth running of the web application.

- PHP – This tool controls the dynamics and the interactivity of web pages [10]. In the Skaïgreen system, PHP was used to implement three primary functionalities.

  Firstly, it was used to connect to the Python code used to trigger the drone to fly. Specifically, a PHP function called shell_exec() was used to execute the Python script used to initiate the flight of the drone. This tool was used in accordance with FR05, which emphasized on the autonomous flight of the drone, as defined in Section 1.5 of the introductory chapter.

  Secondly, PHP was used to connect the Skaïgreen system to a database. To connect to the database using PHP, database credentials were defined for the MySQL server, including the host, username, password, and database name. Then, a new Mysqli instance was created by passing in these credentials to the Mysqli constructor.

This PHP function showed fulfilment of the non-functional requirement of reliability which states, “there should be consistent availability of the web app for users’ use” (NFR01). Without storing user credentials, how would the system be reliably available for users?

Lastly, PHP was used to create a session for each user, which was used to store and access session variables across different pages. This was done to ensure that each user terminated their login session and once logged out, no other user could access their login session. This contributed to overall security of the system (NFR02).

- JavaScript – This tool was used alongside HTML and CSS. This involved the dynamic update of content on the webpages.

Firstly, JavaScript was used to implement regular expressions, which ensured that login credentials followed a specific order. For instance, a regular expression was created to ensure that the password field consisted of at least eight characters, contained at least one letter, and at least one number. The use of regular expressions was to implement the non-functional requirement of robustness- “if the user were to input invalid data, there should be a prompt to notify the user to enter the right details” (NFR05).

Also, JavaScript was used to confirm the user’s choice to fly the drone. To do this, an event listener was created on the “fly drone” menu button on the dashboard, that led to a pop-up box being triggered. This pop-up box then asked the user to either fly the drone or cancel the flight. This tool was used in accordance with FR05, which emphasized on the autonomous flight of the drone, as defined in Section 1.5 of the

introductory chapter. Figure 3.1.4 shows the popup modal that appears after clicking the “Fly Drone” button, on the dashboard of the Skaïgreen web application.

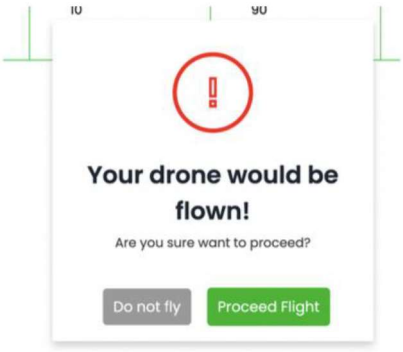


Figure 3.1.4 – JavaScript functionality to confirm the user’s choice of flying the drone.

- MySQL- This is a cloud-based database, meaning that it is accessible to multiple users on the internet [11]. In the Skaïgreen system, MySQL was used to store video footage and imagery captured on the drone. Since the images could not be stored in conventional image format (jpg, jpeg, png, etc.) on MySQL, URL links to the images were stored in the database tables, to provide user access to them. MySQL was also used to store the registration credentials of users that logged into the system. This tool demonstrated the non-functional requirement of reliability, which emphasizes the need for the web application to be consistently available for users to use (NFR01).
- Python- In the Skaïgreen system, this high-level programming language was used in a two-fold format. Firstly, it was used to implement the YOLOv5 machine learning algorithm. Secondly, it was used to implement the functionality needed for the drone to fly autonomously. Both of these sub-architectures of the Python tool would be explained in the Chapter 4 of this paper, which elaborates on the implementation details of Skaïgreen. Python played an integral role in fulfilling three key functional requirements – implementation of the YOLOv5 algorithm to detect and count the

number of coconut trees (FR06), implementation of the YOLOv5 algorithm to detect nitrogen deficient coconut trees (FR07) and the implementation of the drone to fly autonomously (FR05).

3.2 System Overview

The Skaïgreen system had two main components: hardware and software. The hardware component involved the drone technology used to hover over the farm to capture images. The footage obtained by the drone was accessed on the software component, which was a web application interface where farmers could view the results of the preprocessed images with detections to monitor the status of crops on the farm.

The web application was developed using frontend tools such as HTML, CSS, and JavaScript to structure the web pages and align them with specific functionalities. The drone's captured footage was analyzed using machine learning algorithms implemented in Python.

In summary, the system was implemented in three stages. The first stage involved collecting data from the drone and storing it in a database system. The second stage involved creating a website that had the means to present the status of crops and the results of image processing analysis on the crops from the database. Finally, the last stage involved reflecting the trained data (images) from the database onto the web application.

3.2.1 Activity Flow Diagram

The diagram below depicts a visual representation of the activity flow of the Skaïgreen system. It visually shows the entire process of using the system, beginning from the website landing page, and ending on the reflected data from the database unto the frontend (dashboard page).

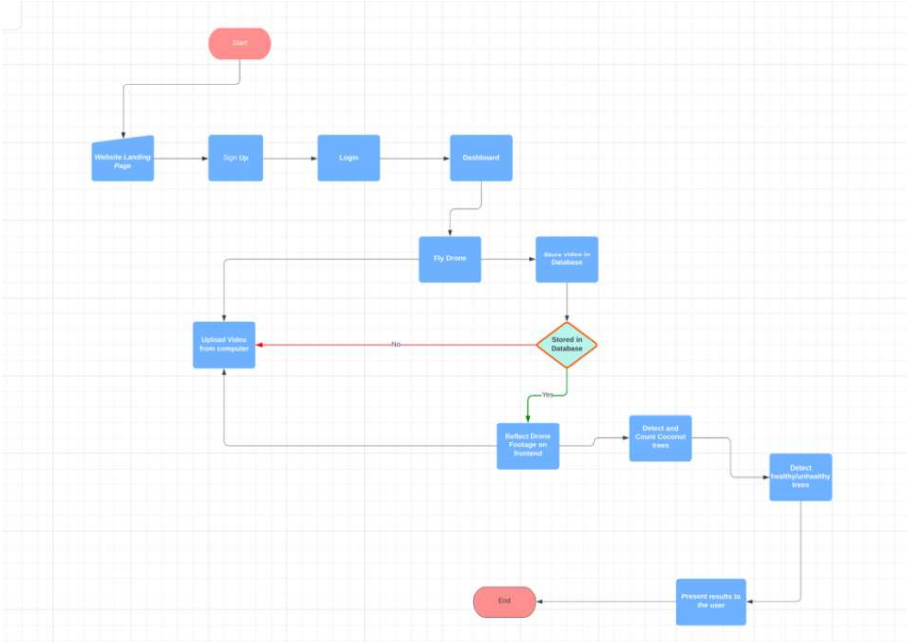


Figure 3.2.1 Activity flow diagram indicating the process the farm owner goes through in using Skaïgreen.

3.2.2 Entity-Relationship Diagram

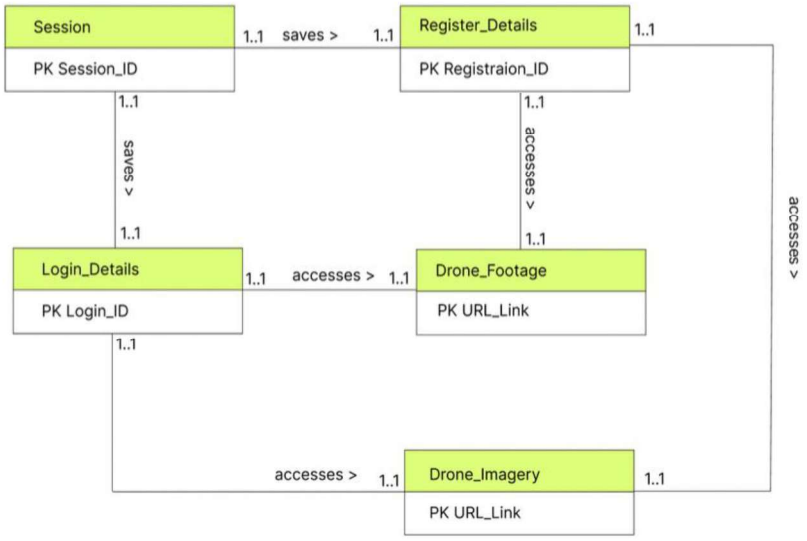


Figure 3.2.2 Entity Relationship diagram showing the database entities of Skaïgreen.

3.2.3 Entity-Relationship Diagram Rules

- Each unique session ID should save exactly one registered user's registration details.
- A registered user's registration details should correspond to exactly one session ID.
- Each unique session ID should save exactly one registered user's login credentials.
- A registered user's login details should correspond to exactly one session ID.
- The login details for a user can only access the drone footage for that single user.
- The drone footage specific to user is only accessible via his/her login credentials.
- The login details for a user can only access the drone imagery for that single user.
- The drone imagery specific to a user is only accessible via his/her login credentials.
- The registration details for a user can only access the drone imagery for that single user.
- The drone imagery specific to a user is only accessible via his/her registration credentials.
- The registration details for a user can only access the drone footage for that single user.
- The drone footage specific to a user is only accessible via his/her registration credentials.

Chapter 4: Implementation

4.1 Introduction

As stated in the previous chapter (Section 3.2), the Skaïgreen system comprises of a hardware and a software component. The hardware component pertains to the drone technology, and the software primarily centers on the implementation of the YOLO v5 machine learning algorithm.

4.1.1 Hardware – Google Tello drone

The Google Tello Drone was used to capture images and videos of coconut crops on the farm. This captured footage then fed into the next stage of processing – application of YOLOv5. The Google Tello drone was the drone chosen for Skaïgreen because of its programmable capabilities. Using Python script, the drone can be programmed to move in multiple directions. This feature would be useful in the fulfillment of FR04 (Atonomous flying of the drone). More detail on the programming capabilities of the drone would be provided in Section 4.2.1.

The Google Tello drone had some weaknesses which did not make it an all-round best fit for this project. Firstly, the drone had a forward-facing camera which was not suitable. To capture crops from the drone flight height of 15 feet, the camera needed to be downward facing. To address this issue, a 3D printed mirror holder was designed and installed in front of the camera. The mirror reflected the crops towards the camera, allowing for wider coverage of the farm. This process can be seen in Figure 4.1. Figure 4.2 shows the physical installation of the 3D printed mirror unto the drone.

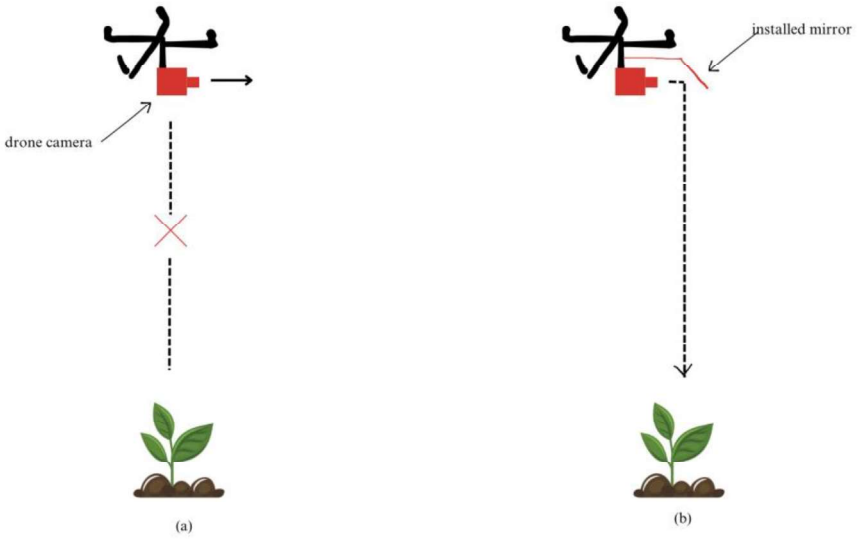


Figure 4.1 Process for reflection of a crop into the drone camera with the aid of a mirror. (a) shows the drone view before mirror installation. (b) shows the drone view after the mirror's installation.

Figure 4.2 3D printed mirror holder (a) 3D printed mirror holder mounted unto drone (b)

In addition, the Google Tello drone has a flight time of approximately 11 minutes and requires 1 hour and 30 minutes to charge [12]. This ratio of flight time to charge time is highly inefficient, considering the size of the coconut farm (14.8 hectares). The documentation for the Google Tello drone [18] indicated that it was designed for short interval flights indoors and not

for outdoor use, which explains its short flight time. To overcome this challenge, the DJI Mavic Mini 2 Drone was used to collect the image data on the farm due to its average flight time of 30 minutes and three replaceable batteries that could be charged independently [19]. This allowed for continuous operation by swapping the batteries as needed. However, the DJI Mavic Mini 2 Drone was not programmable, and as such, the Google Tello drone was only used for testing the programmable features of drones and implementing autonomous flight (as described in Section 4.2.1).

It would be pertinent to note that, it was due resource constraints that the programmable drone needed for outdoor farm management was not acquired. A more appropriate drone would have been the DJI Agras T40 (see Section 6.3 for more information).

4.1.2 Software – YOLOv5 machine learning algorithm – Nitrogen Deficiency Detection

Primarily, the YOLOv5 algorithm follows a process of data collection, preparation of the data, training, interpreting results from training and deploying unto the web application.

4.1.2.1 Data Collection

As mentioned in Section 1.2, the coconut farm's total size was 14.8 hectares. However, for testing the Skaïgreen system, a smaller section of the farm was selected. The dataset used for testing comprised of 300 images captured using the DJI Mavic Mini 2 Drone specifically in the selected section of the farm. To re-iterate, the Google Tello Drone was not used for image capturing on the farm due to its limitations (see Section 4.1.1). The images were captured at a height of 2 meters using the DJI Mavic Mini 2 Drone, which provided a close-up view of the crops, making it suitable for detection. See Appendix B below.

4.1.2.2 Preparation of data

Labeling

The data was annotated using the Roboflow annotation tool, as explained in Section 4.2.2. To prepare the dataset for training, the images were labeled by applying bounding boxes to outline the coconut trees in the images. The bounding boxes were then assigned text labels - "healthy" and "unhealthy" - to distinguish between the two types of trees. Labeling the dataset was crucial for enabling the YOLOv5 model to identify healthy and unhealthy plants accurately. See Appendix C below.

Data Augmentation

To augment the dataset, synthetic image variations were generated using techniques such as vertical and horizontal flips, as well as clockwise and counterclockwise rotations. The original dataset comprised 300 images, but an additional 100 synthetic images were created through augmentation, resulting in a total of 400 images. The augmentations were crucial to avoid overfitting, which occurs when the trained dataset only recognizes images that resemble those in the dataset. Without augmentation, the model would fail to accurately predict an image if it is unlike the images in the trained dataset (such as detecting nitrogen deficiency). By adding noise to the images, augmentation creates a more resilient system capable of detecting nitrogen deficiency in images captured under different lighting conditions, camera angles, and other factors. See appendix D below.

Dataset Sectioning – Train, Test, Validate

The dataset was divided into three categories, each of which was essential for training the model. The first category, the "train" set, consisted of images that enabled the model to identify healthy and unhealthy images. This set was utilized for training the model. The second category, the "test" set, comprised images that evaluated the accuracy of the trained model derived from the train set. The test set verified whether the model could accurately recognize

images that displayed nitrogen deficiency in coconut trees. Finally, the third category, the "validation" set, confirmed the accuracy of the results generated by the test set. See Figure 4.2.2 (a)

4.1.2.3 Training

Yolov5 was used to train the data. The YOLOv5 was trained to identify a set of characteristics in images of coconut plants. The YOLOv5 was trained on a dataset of images of healthy and unhealthy coconut plants, with labels indicating whether each plant is showing signs of nitrogen deficiency. YOLOv5 identified patterns and features in the images that were associated with nitrogen deficiency. These include changes in shape, size, and leaf color, as well as other visual cues that are indicative of poor plant health. New images of coconut plants were classified as healthy or unhealthy, based on the existence of these features.

```
# train yolov5s on custom data for 100 epochs
# time its performance
%%time
%cd /content/yolov5/
!python train.py --img 416 --batch 16 --epochs 279
--data {dataset.location}/data.yaml --cfg ./models/custom_yolov5s.yaml
 --weights '' --name yolov5s_results  --cache
```

Figure 4.1.2.3 – Code used to train the Yolov5 Model with the initialization of the batch size and number of epochs.

4.1.2.4 Deploying the Model

The model was trained and subsequently deployed onto a Flask-based web application. The dashboard displays images depicting the number of healthy and unhealthy coconut plants identified, as well as the total number of coconut plants detected.

4.1.2.5 Analysis of Results after Training (Nitrogen Deficiency Detection)

To analyze the accuracy of the model used to train the dataset, a graph known as the Model Precision Accuracy graph, or mAP for short was utilized. The mAP is used to measure

how well the model identifies the object of interest within an image [13]. After the model was trained over 279 epochs, the mAP value was 50.7%. This mAP value indicated that on average, the model correctly identified nitrogen deficient and healthy coconut trees in 50.7% of the images in the dataset. Figure 4.1.2.5 illustrates the Model Precision Accuracy graph, covering 279 epochs. The vertical axis represents the mAP values, while the horizontal axis represents the number of epochs. The graph shows that the mAP value reached its peak at the 114th epoch. In an ideal scenario, the mAP value would continue to increase until the 279th epoch.

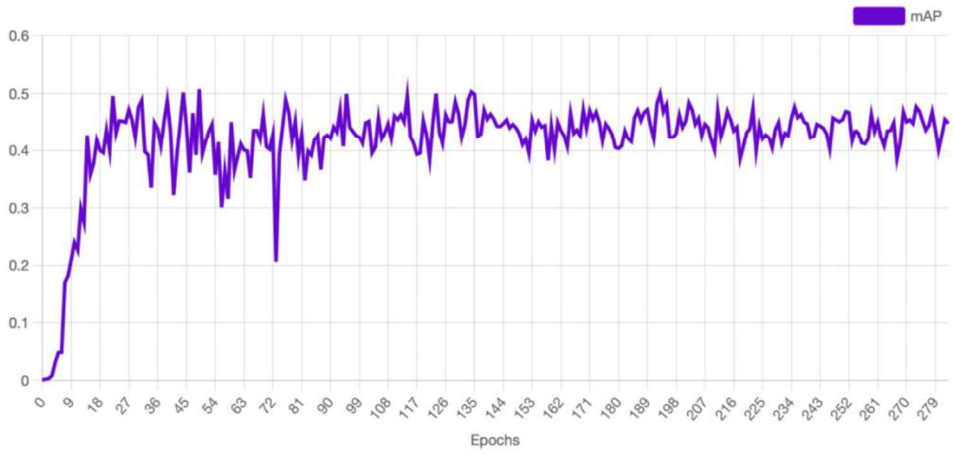


Figure 4.1.2.5 – Model Accuracy Precision graph for Nitrogen Deficiency Detection

4.1.3 Software – YOLOv5 machine learning algorithm – Coconut Plant Counting

The process of counting coconut plants follows a similar approach to nitrogen deficiency detection. It involves data collection, data preparation, dataset training, and model deployment. However, the key difference lies in the algorithm used.

Algorithm and Code Snippet

Figure 4.1.3 showcases a code snippet that detects and counts coconut plants in both images and videos. A bounding box is drawn around each detected coconut plant, and the corresponding class name (i.e., "coconut") and count are assigned to it. The draw_counts()

function labels the coconut plants, and the putText() function adds the class name and count to the coconut image frame. The input data comprised four images and two videos, which were processed and detected using a pre-trained model. The count_object() function was used to count the number of coconut plants. The OpenCV library was used to outline the detected images with bounding boxes. The draw_counts() function was then called to label the coconut plants with their respective class names and counts. The annotated images and videos, along with the coconut counts, were saved in a CSV file.

```python
def draw_counts(frame, counts, font_scale=1, thickness=2):
    for idx, (class_name, count) in enumerate(counts.items()):
        cv2.putText(frame, f'{class_name}: {count}', (10, 30 + idx * 30), cv2.FONT_HERSHEY_SIMPLEX, font_scale, (0, 0, 255), thickness, cv2.LINE_AA)

# Load image or video
input_data = 'Coconut-Trees-4/test/videos/cc1.MP4'

# Process image or video
if Path(input_data).suffix in ['.jpg', '.jpeg', '.png']:
    # Read image
    img = cv2.imread(input_data)
    img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

    # Run inference
    results = model(img_rgb)

    # Count objects
    counts = count_objects(results.xyxy[0], class_names)

  # Draw bounding boxes on the image
    for *xyxy, conf, cls in results.xyxy[0]:
        x1, y1, x2, y2 = int(xyxy[0]), int(xyxy[1]), int(xyxy[2]), int(xyxy[3])
        cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)

    # Draw counts on the image
    draw_counts(img, counts)

    # Save and display image
    cv2.imwrite('output_image.jpg', img)
    cv2_imshow(img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

    # Save counts to a CSV file
    counts_df = pd.DataFrame([counts], columns=counts.keys())
    counts_df.to_csv('object_counts.csv', index=False)
```

Figure 4.1.3 – Code Snippet of the algorithm for counting coconut trees detected.

4.1.3.1 Analysis of Results after Training (Coconut Tree Detection and Counting)

As discussed in Section 4.1.2.5, the Model Precision Accuracy graph displayed mAP values against each epoch of training. The model was trained to detect and count coconut trees, and its average precision accuracy was 78.2%. This value suggests that, on average, the model correctly identified coconut trees in 78.2% of the dataset images. Figure 4.1.3.1 presents a graph plot of the mAP values across training epochs. The model used 160 epochs, and at the 160th epoch, the mAP value was 65%. The graph indicates that the mAP value reached its peak

at the 87th epoch. In an ideal scenario, the mAP value would continue to rise until the 160th epoch.

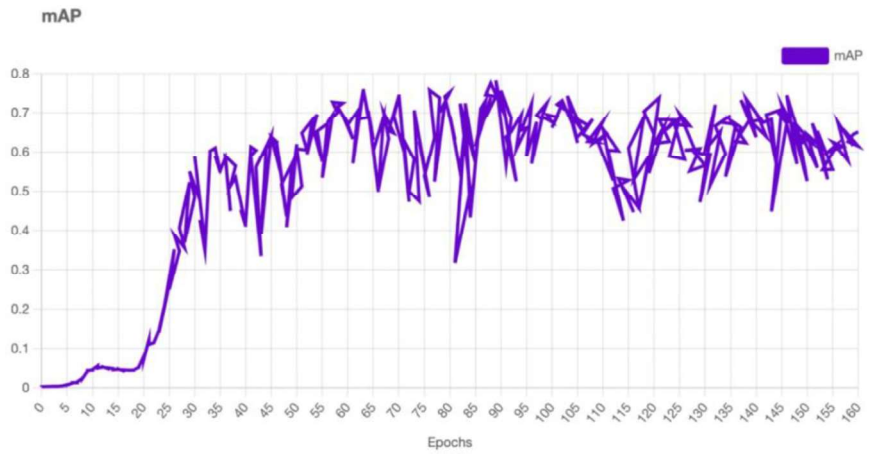


Figure 4.1.3.1– Model Accuracy Precision graph for Coconut Counting

4.1.4 Risk Management

As part of the implementation of Skaïgreen, certain risk factors must be taken into consideration. To complete an adequate risk management analysis, each risk would be listed and steps to mitigation would be explained accordingly.

Weather: The drones are designed for outdoor surveillance however, in the case of thunderstorms, severe winds, amongst others, their operation may be heavily hampered. What then could the farmer do in case the drone is being flown as usual and there is a sudden thunderstorm? The drone would have to be taken down and stored in a safe place. However, this action of stopping the drone in flight could disrupt the data collection process. As an additional future work, there could be the installation of sensors to detect bad weather, to trigger the drone to automatically terminate the flight process.

Battery Death: In general, drones have an average flight time of 10 to 15 minutes [12], hence cannot fly indeterminately. Therefore, to mitigate this problem, it would be suggested to the user in the "help section" of the web app, to ensure that the drones' batteries are fully

charged before use, or before the scheduled flight time. If the drones' batteries run low during flying, a notification would be sent to the user's web app, to stop flight, charge the drone and continue flying after it has been charged.

4.2 Key Technologies and Frameworks

This section will detail the technologies utilized by Skaïgreen in developing its web application. Once users successfully authenticate their credentials, the onboarding process begins. The dashboard displays crop status, including the number and health condition of crops, and provides a grid layout for a comprehensive view of the farm. This grid allows the coconut farmer to track the yield of crops. Furthermore, the web application features an autonomous drone flight option, powered by a Python script that leverages the Tello library to provide coordinates for automated flight.

4.2.1 Ground Control Station and Tello Network

The Tello drone can only fly via Python script when the code and drone network are in sync. Once the same network is utilized, the drone can take off at the execution of the Python script. The drone then captures images while in motion. The ground station monitors surveillance activities of the drone on the farm as well as remotely controlling the drone. The ground station is situated on the coconut farm.

```
from djitellopy import tello
from time import sleep # add delays between each command

# create tello object to connect to tello
tl = tello.Tello()
tl.connect()

#check for battery of the drone
print(tl.get_battery())

tl.takeoff()
tl.send_rc_control(30,0,0,0) #drone moves right
sleep(2) #delay before landing

tl.send_rc_control(0,20,0,0) # drone moves forward
sleep(2) #delay before landing

tl.send_rc_control(0,0,20,0) # drone moves upward
sleep(2) #delay before landing

tl.send_rc_control(0,0,0,60) #drone rotates
sleep(2) #delay before landing

tl.send_rc_control(0,0,0,0) #to prevent drone from landing in forward motion | lands like a helicopter
tl.land()
```

Figure 2.4.1 Code Sample: Automated left, right, backwards, and forward movements of the drone

In Figure 2.4.1, tl.send_rc_control() defines the directions the drone ought to fly. This function passes four arguments as directions. In an orderly fashion, the first argument maneuvers the drone to the left and right, the second argument points forward and backwards, the third argument points upward and downward and lastly the fourth argument moves in a clockwise and counterclockwise velocity.

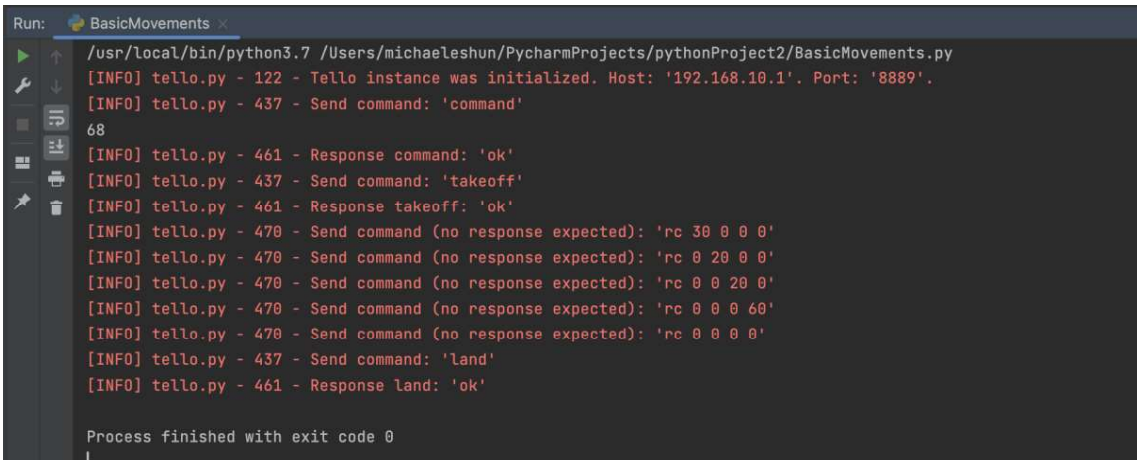


Figure 2.4.2 Code Sample: Illustration of ground station and Tello Network

4.2.2 Roboflow

The images captured by the drone were processed in Roboflow, a computer vision platform that was utilized for annotating or labeling images, as well as training and testing computer vision models [13]. The images were annotated to aid in the detection of coconut crops, and the resulting dataset underwent training, testing, and validation. To improve training results, the dataset (images uploaded into Roboflow) went through four major steps: source images, train/test split, preprocessing, and augmentation.

The dataset that was uploaded into the Roboflow system consisted of 100 images captured by the DJI Mavic Mini 2 Drone (as discussed in Section 4.1.2.1) and an additional 200 images obtained from various open-source coconut plant images on the internet, resulting in a total of 300 images. After data augmentation (see Section **4.1.2.2)**, The total number of images in the dataset came to 400. The dataset was then split into three categories for the train/test split: training, testing, and validation. As seen in Figure 4.2.2(a), 88% of the images were assigned to be trained, 8% of the images were used in the validation set, and 4% of the images were used in the testing set. During the preprocessing stage, image properties such as

size, contrast, and grayscale were adjusted to ensure the model learned from different variations of the dataset. Finally, the images underwent augmentation, where synthetic training data was created to improve the model and increase the diversity of learning examples. The resulting model was then deployed.

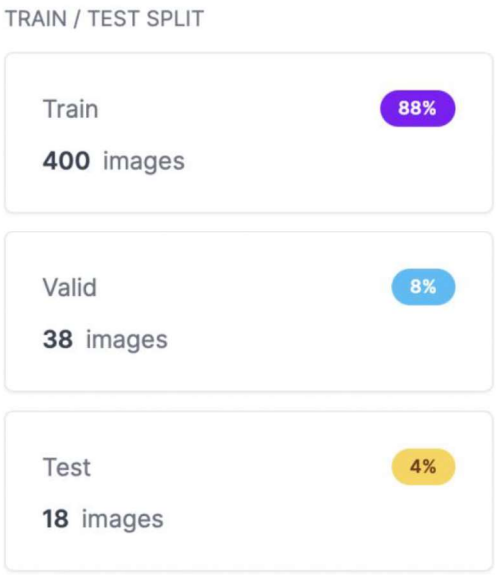


Figure 4.2.2(a) Splitting the dataset into training, validation, and test sets.

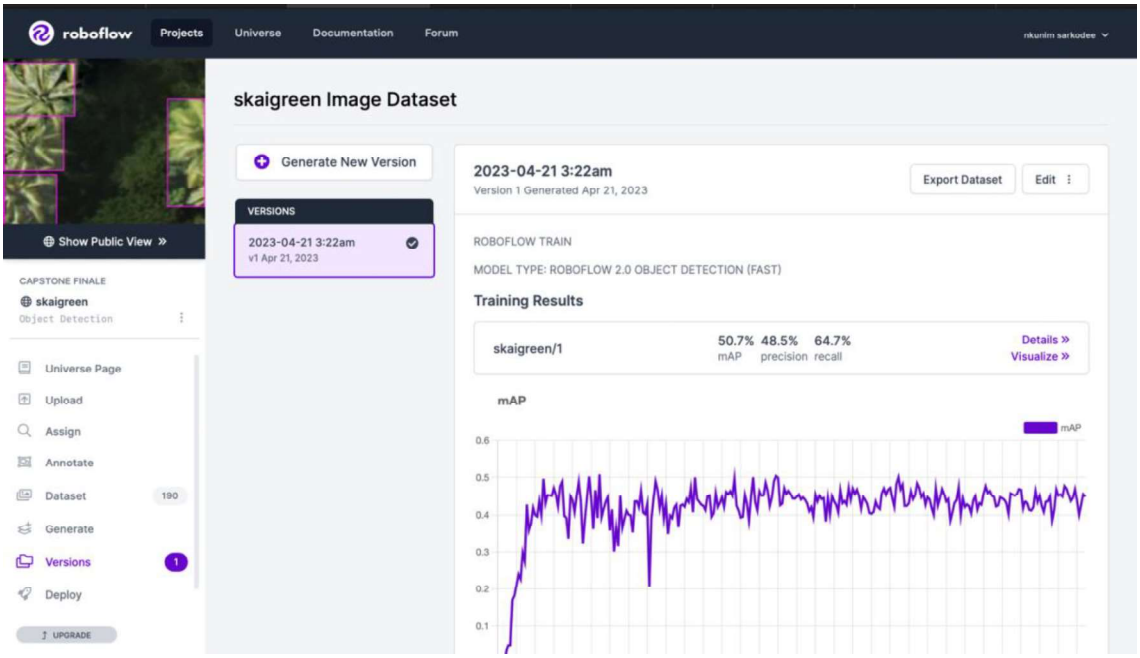


Figure 4.2.2(b) Roboflow Interface

4.2.3 PyTorch

PyTorch was used as the machine learning framework for deep learning purposes. The torch library enabled the training of the Yolov5 model. The Roboflow model generated was deployed, and Yolov5 was employed to aid in the detection of the coconut trees. After setting up Yolov5, training started with 279 epochs and a batch size of 16, which represented the total number of complete passes in the training dataset and the specific quantity of data samples that were processed and underwent model updates, respectively.

```
# train yolov5s on custom data for 100 epochs
# time its performance
%%time
%cd /content/yolov5/
!python train.py --img 416 --batch 16 --epochs 279
--data {dataset.location}/data.yaml --cfg ./models/custom_yolov5s.yaml
 --weights '' --name yolov5s_results  --cache
```

Figure 4.2.3 Code sample: Training Yolov5

4.2.4 Implementation Results

Landing Page

Figure 4.2.4.1 shows the landing page the user would view upon accessing the Skaïgreen website thought the internet. On this page, information about the services offered by Skaïgreen can be seen. The button to take the user to the login/ Signup Page can also be seen.

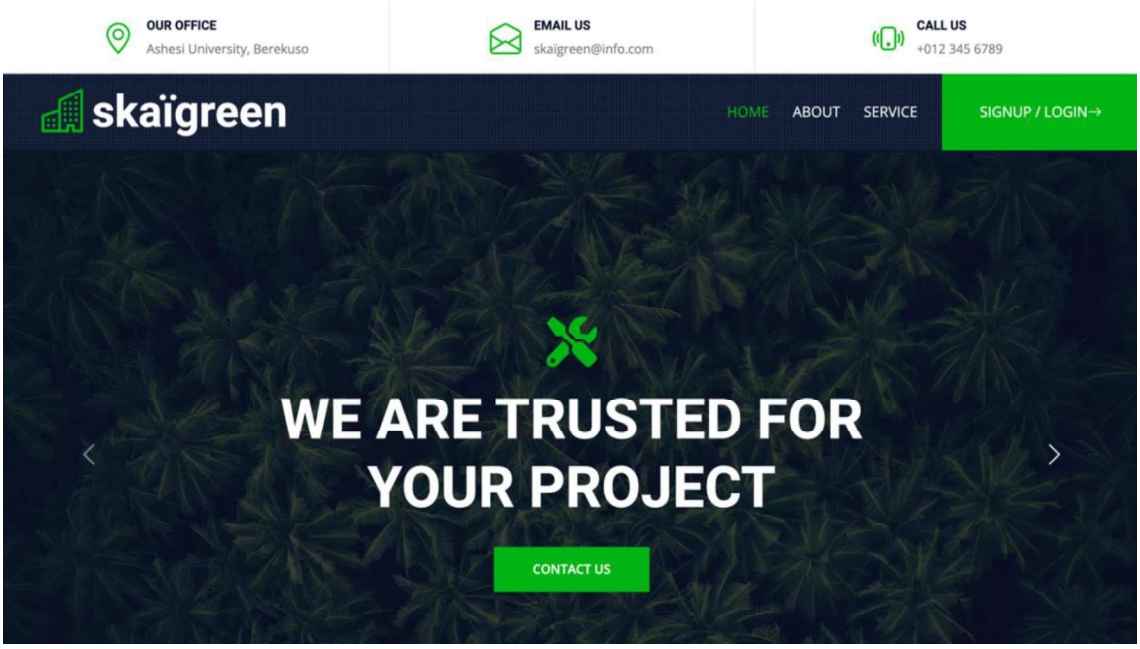


Figure 4.2.4.1 Landing Page for the Skaïgreen website Application

Login & Signup Page

Figure 4.2.4.2 shows the Signup Section of the Skaïgreen web application. Here the farmer would be required to enter valid credentials, to gain access to, and utilize the web application. Upon a successful Login or Sign Up, the dashboard would then be made accessible to the user.

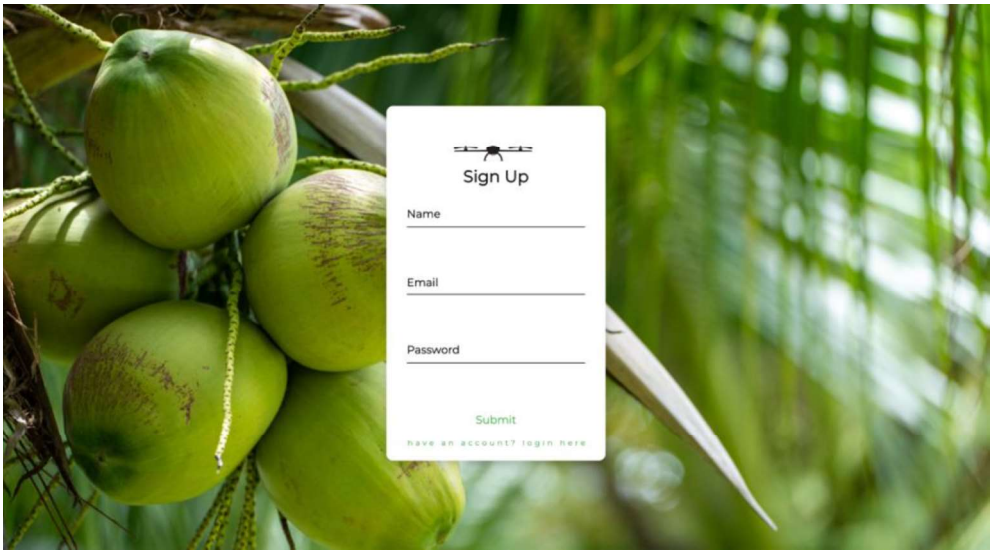


Figure 4.2.4.2 Login/Signup for Skaïgreen website Application

Dashboard Page

Figure 4.2.4.3 shows the Dashboard of the Skaïgreen web application. Here the farmer would be able to view certain key elements of the farm such as the numb er of coconut trees, the number of defective coconut trees out of the total count of trees, as well as the number of healthy coconut trees. On the left-hand side of the dashboard page, varying aspects of the Skaïgreen web app can be accessible.

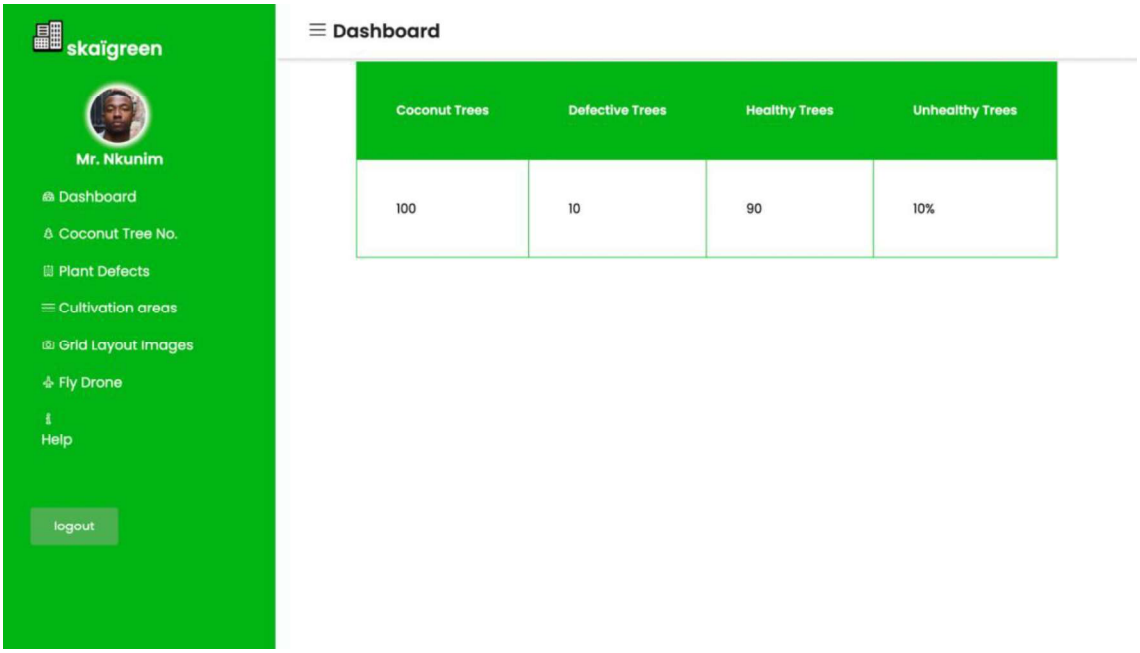


Figure 4.2.4.3 Dashboard Page of the Skaïgreen web application

Upload Functionality

Figure 4.2.4.4 shows the upload functionality of the Skaïgreen web application. In the requirements gathering section, the coconut farmer specified the upload of footage for nitrogen

detection, should differ from the upload of footage for coconut counting. Hence, a separate upload button was made for each to fulfill that requirement.

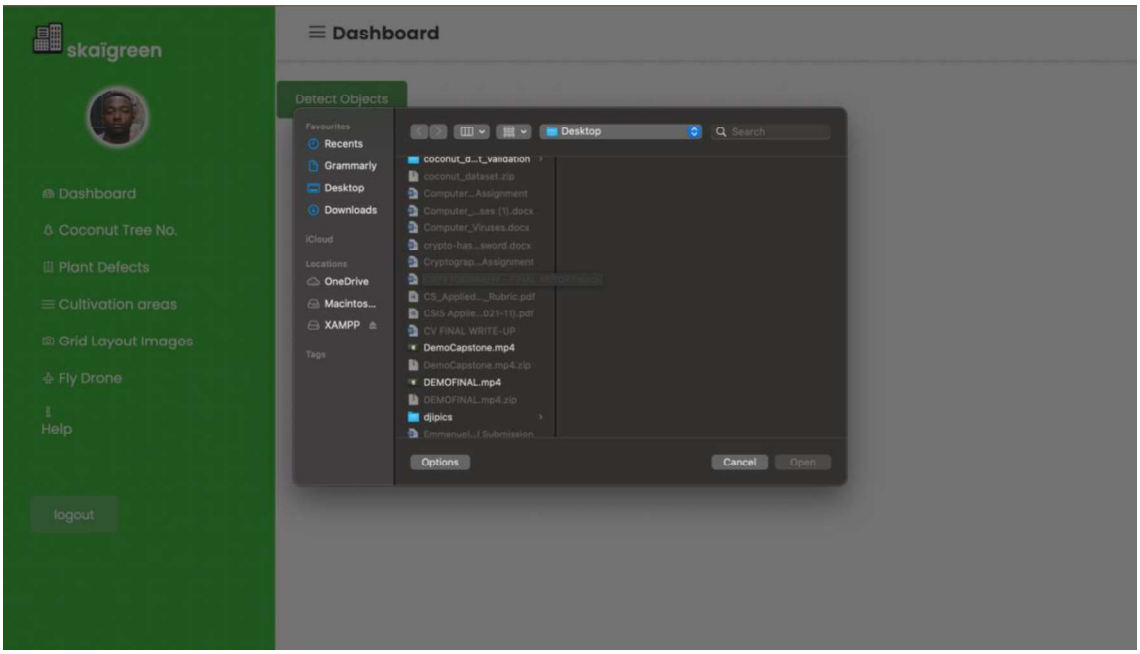


Figure 4.2.4.4 Selection of image file to be used for detection.

Detection of Nitrogen Deficient plants

Figure 4.2.4.5 shows the detection of nitrogen deficient plants on the Skaïgreen web application. After uploading the footage for detection, the resulting detected image is displayed with pertinent details shown in a green box, under the detected image.

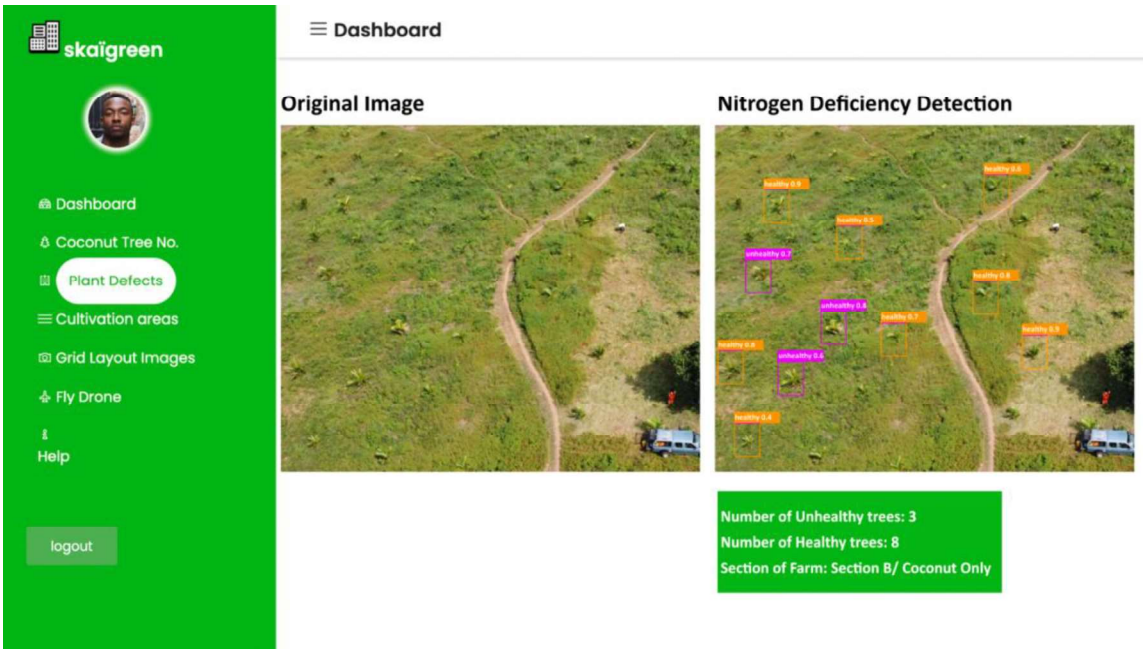


Figure 4.2.4.5 Detection of Nitrogen Deficient coconut plants

Counting of Coconut Plants

Figure 4.2.4.6 shows the counting of coconut plants on the Skaïgreen web application. After uploading the footage for counting, the resulting image is displayed with pertinent details on the number of coconut plants counted shown in a green box, under the detected image.

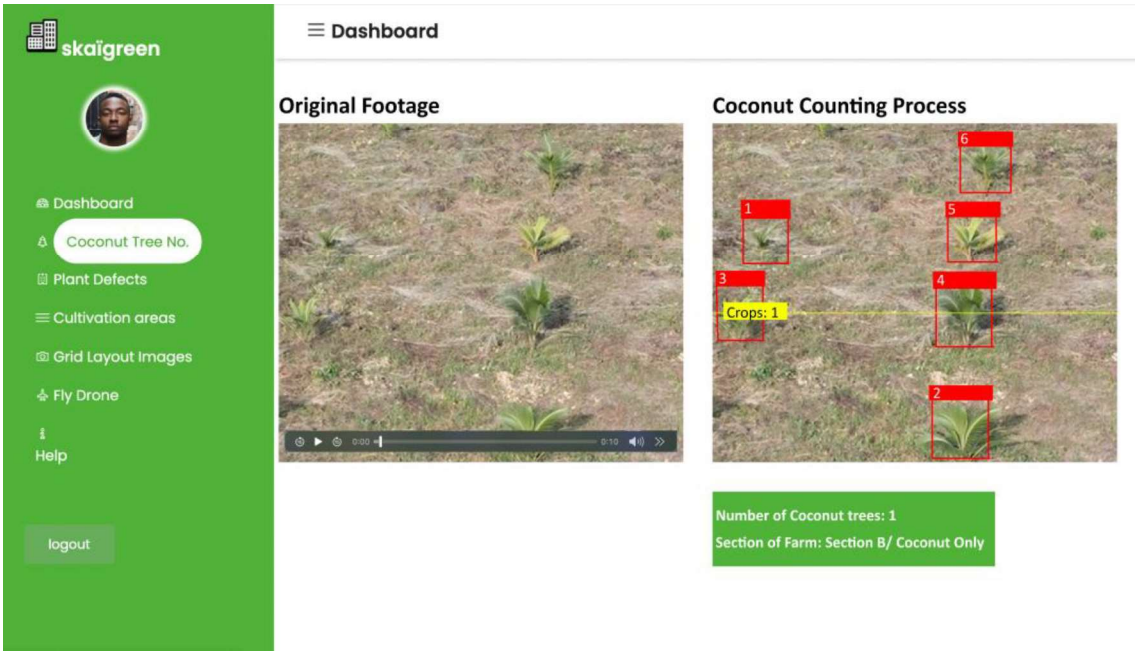


Figure 4.2.4.6 – Counting of coconut crop

Autonomous Flight of Drone.

Figure 4.2.4.7 shows the process a user would go thorough before flying the Google Tello drone via the Skaïgreen web application. This pop-up modal appears when the user clicks the "Fly Drone" option on the side menu of the dashboard. It would be an important safety step to ensure that the drone is in the right position to take flight.

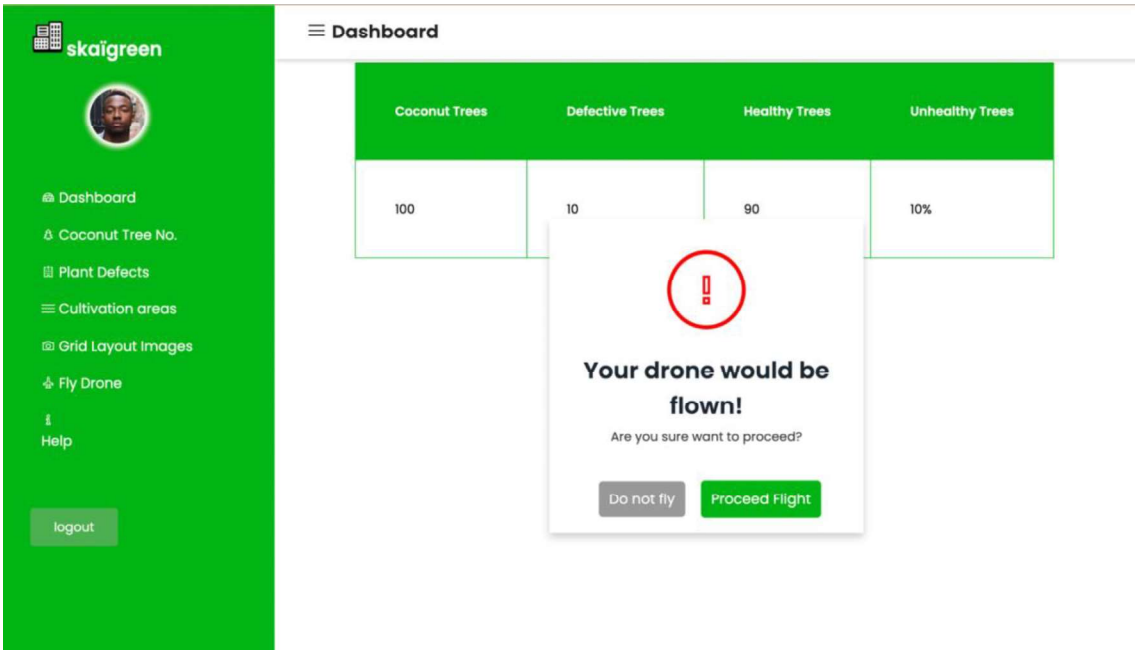


Figure 4.2.4.7 – Automated flight of drone with button click aid

Chapter 5: Testing and Results

In this section, the validation of the functional requirements of Skaïgreen would be presented. This would be done by testing the system in three distinct ways, namely: component testing, system-level testing, and user testing.

5.1 Component Testing

In this section, each component of the Skaïgreen system would be tested. These components include, nitrogen deficiency detection, coconut plant counting, and provision of a bird's eye view of the farm.

5.1.1 Nitrogen Deficiency Detection Test

The aim of this test was to verify if upon flying the Google Tello drone to collect footage, the YOLO machine learning algorithm would be able to detect the Nitrogen Deficient coconut plants. To conduct this test, 2 images were captured. One image showed a healthy coconut tree branch with green leaves and the other showed an unhealthy coconut plant with yellow leaves, indicative of nitrogen deficiency. The following image shows the results after parsing the YOLO Nitrogen deficiency detection algorithm on both images.

(a) (b)

Figure 5.1.1 – Results after Nitrogen Deficiency Detection Testing.

Analysis

From conducting this test, it is evident that the nitrogen deficiency detection functionality of the Skaïgreen system is functional. The model's detection threshold was set at 0.4 or 40%, indicating that images with a probability lower than 40% of being nitrogen deficient coconut plants were not recognized. Therefore, the YOLO algorithm's accuracy in distinguishing between nitrogen deficient coconut plants and healthy ones is demonstrated by its 80% and 70% confidence levels, respectively.

5.1.2 Coconut Tree Counting Test

The purpose of this test was to validate the YOLO algorithm's ability to detect and count coconut trees from images or footage. To carry out the test, drone video footage of a section of the coconut farm was captured using a DJI Mavic Mini 2 Drone. Subsequently, the video was uploaded to the YOLO machine learning algorithm for detection and counting of the coconut plants. Figure 5.1.2 displays the outcome of this process.

Analysis

To count the coconut plants, as the video footage of the drone is being played, detected coconut plants that pass the yellow threshold bar are added to the total crop count, and displayed accordingly. However, the test showed a miscount of the crops, as one of the crops with ID 2, ID 3 and ID 4, were not included in the crop count (The IDs are the numbers in the red section of the bounding boxes surrounding the coconut plants, as seen in Figure 5.1.2). That is, two out of the three coconut plants that passed the threshold bar were counted, thus leaving one plant out, Again, upon careful inspection, one of the coconut plants just above the yellow threshold bar, was not detected. In this regard, since majority of the coconut plants were

counted and detected and not all, it would be important to inform the user that the system provides a rough estimation of the number of coconut plants and is not 100% accurate. Moreover, a re-evaluation of the YOLO algorithm, including adjusting the number of training epochs and considering a different YOLO version such as YOLOv7, could potentially result in more precise outcomes.

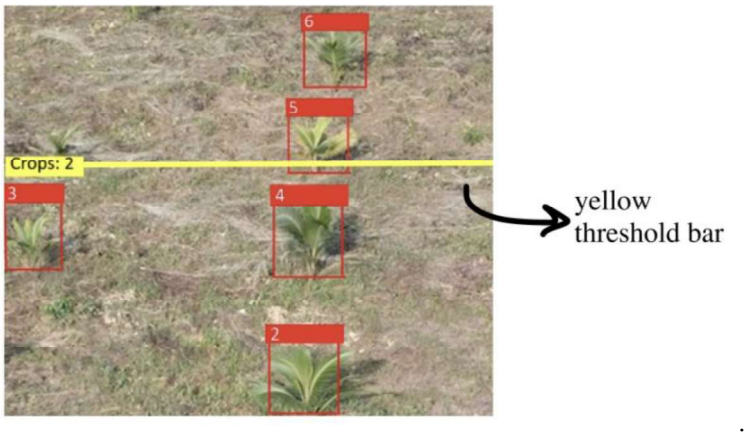


Figure 5.1.2 – Results after coconut detection and counting.

5.2 System Level Testing

In this section, the entire Skaïgreen system was tested to determine if all the specified functional requirements were met. The following flow chart diagram demonstrates the use Skaïgreen system from start to finish. To test, the results after each step in the flow chart diagram would be recorded and analyzed:

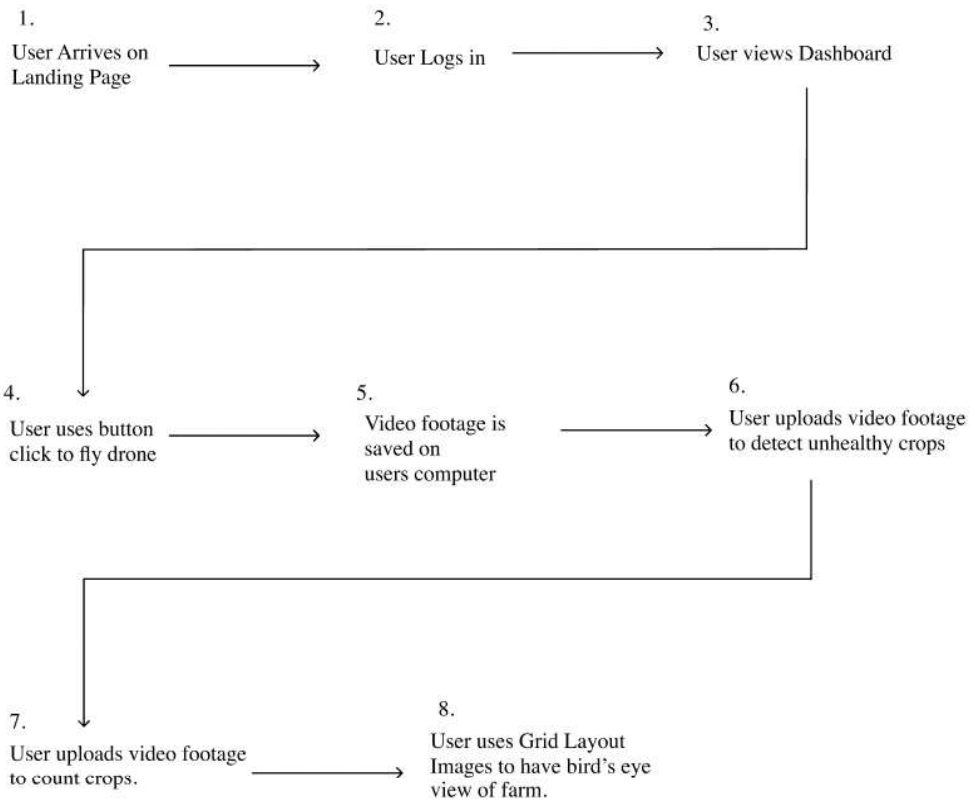


Figure 5.2 – Flow Chart Showing Skaïgreen System Process in 8 steps.

5.2.1 Landing Page Testing

The landing page was hosted on the world wide web, and so remains accessible to multiple users. To test this feature, two users in different locations were asked to visit the webpage to see it was viewable. Both users confirmed their viewing of the Skaïgreen website landing page. Kindly refer to Appendix E below.

5.2.2 User Login/ Registration Testing

To test the user Registration process, a new user with new credentials was entered into the Skaïgreen system. The test here was to see if the user's registration credentials were saved into the MySQL database. From going through this process, it was evident the user's

registration credentials were saved into the database table. Kindly refer to Appendix F and G below.

5.2.3 User Button Click to Fly Drone Test

The aim was to test if clicking the button would trigger the drone to take off. Upon clicking the button on the Skaïgreen web application, the drone flew forward for 5ft and landed safely.

5.2.4 Video Footage Saved on User's computer Test.

To test that video that was recorded by the drone, during its "button click flight" was saved on the user's computer. A JavaScript popup box was created if the drone footage was successfully saved on the user's computer, and an error message was displayed if it was not. From the drone flying, the video was successfully saved unto the user's PC, into the downloads folder. Kindly refer to Appendix H below.

To obtain information about the tests for detecting unhealthy crops and counting coconut trees, please refer to the component tests conducted in Sections 5.1.1 and 5.1.2, respectively, as documented in the results and analysis of the study.

5.3 User Level Test

To conduct the user test, the main user was sent a URL link to the Skaïgreen web application to enable them to use the system in real-time. To gather feedback from the user after using Skaïgreen, a Google Form was created and sent to the user (see Appendix I below). The form included questions aimed at determining the level of satisfaction with each functional requirement that was documented. Figure 5.2.1 illustrates the format in which the questions were presented to the user.

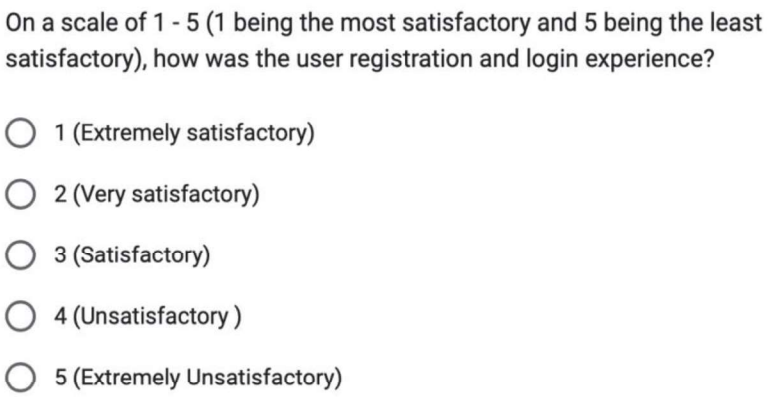
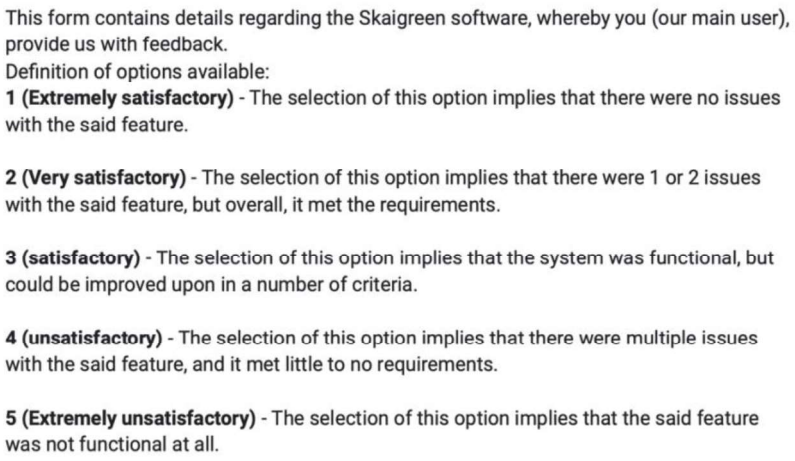


Figure 5.2.1 – Google Form question example

Based on the sample question provided in Figure 5.2.1, the purpose was to assess the user's level of satisfaction with the login and registration feature of Skaïgreen. The comprehensive rating scale provided allowed the user to rate their experience. If any issues the user faced were not covered by the questions asked, the user could report encountered issues. To do this, there was a section on the Google Form that allowed the user to provide more detailed feedback.

To clarify the meaning of each option selected, the definition of "extremely satisfactory, very satisfactory" amongst others was provided as a heading at the top of the Google Form questionnaire. This was done to ensure that the user understood the meaning of each rating option. Figure 5.2.2 shows this.


This form contains details regarding the Skaigreen software, whereby you (our main user), provide us with feedback.
Definition of options available:
1 (Extremely satisfactory) - The selection of this option implies that there were no issues with the said feature.

2 (Very satisfactory) - The selection of this option implies that there were 1 or 2 issues with the said feature, but overall, it met the requirements.

3 (satisfactory) - The selection of this option implies that the system was functional, but could be improved upon in a number of criteria.

4 (unsatisfactory) - The selection of this option implies that there were multiple issues with the said feature, and it met little to no requirements.

5 (Extremely unsatisfactory) - The selection of this option implies that the said feature was not functional at all.

Figure 5.2.2 – Descriptions of ratings for features of the Skaïgreen system

Chapter 6: Conclusion and Recommendations

6.1 Conclusion

This project aims to enhance the monitoring of farm produce by improving the detection of crop health and yield. The Skaïgreen system has been developed based on the requirements definition process to help coconut farmers conveniently access and monitor their farms using a web application and an aerial drone. The system allows farmers to view their entire coconut farm, detect and count the number of coconut plants, and identify nitrogen-deficient plants.

The Skaïgreen system is user-friendly and allows farmers to operate the drone autonomously by simply clicking a button. The drone automatically navigates through the desired regions using the provided coordinates and provides a bird's eye view of the entire farm. The dashboard provides real-time monitoring and displays a detailed count of the crops on the farm.

In conclusion, the Skaïgreen system represents an innovative solution that utilizes drone technology, and deep learning models to improve coconut crop monitoring and support the farmer in enhancing crop yield and ensuring optimal crop health.

6.2 Limitation

Some limitations came up when flying the drone. These include:

- Responsiveness to mobile devices: The Skaïgreen system was not designed as a mobile application due to the larger resolution of the farm footages, which exceeded the dimensions of a mobile frame, as well as the specific requirements provided by the client. As a result, the system's responsiveness to mobile devices was not prioritized.

- Short battery life span: The battery of the Google Tello Drone is limited. As such the flight time of the drone was also limited. On the other hand, charging the drone took an hour and thirty minutes to fully charge.
- Flight time: Due to the short life span of the drone's battery. The drone had a limited flight time of 13 minutes. This hindered the full aerial monitoring of the entire farm.

6.3 Recommendations and Future Works

This section offers recommendations and potential new developments for the Skaïgreen system. The following suggestions have been proposed:

- Excel feature: The data displayed on the dashboard should be retrievable in an excel sheet format. This feature would allow for a more detailed analysis of the data, including crop count, healthy and unhealthy crops, and analytic diagrams depicting yield estimates at the end of a harvesting period. This information would enable farmers to make time-sensitive decisions for a fruitful yield [14].

- Drone upgrade: An upgrade to the DJI Agras T40 drone is recommended, as it is suitable for long flight hours with an internal battery life of 3.3 hours. The drone is programmable and capable of plant stand counting, which aids in yield management. Additionally, the drone can estimate plant health and make decisions that affect the yield [15].

- Location setup: When a coconut plant is detected, the Skaïgreen system should pin the location of the plant for the farmer's convenience. The farmer should receive a notification via the mobile application or SMS indicating the location of the plant on the farm. This feature would enable farmers to locate specific plants more easily, thereby saving time and increasing efficiency [16].

- Bad weather detection sensor: The drone should be equipped with a bad weather detection sensor that can alert the user to terminate the flight automatically. Alternatively, the drone could automatically return to the user's location to avoid damage due to bad weather conditions. The installation of sensors on the drone would aid in achieving this objective [17].

References

[1] “Ghana: export of agricultural products 2008-2020 | Statista.” https://www.statista.com/statistics/1111140/export-of-agricultural-products-from-ghana/ (accessed Sep. 30, 2022).

[2] D. R. Blidberg, “The Development of Autonomous Underwater Vehicles (AUV); A Brief Summary”.

[3] H. M. Jalajamony, M. Nair, P. F. Mead, and R. E. Fernandez, “Drone Aided Thermal Mapping for Selective Irrigation of Localized Dry Spots,” IEEE Access, vol. 11, pp. 7320–7335, 2023, doi: 10.1109/ACCESS.2023.3237546.

[4]B. Neupane, T. Horanont, and N. D. Hung, “Deep learning-based banana plant detection and counting using high-resolution red-green-blue (RGB) images collected from unmanned aerial vehicle (UAV),” PLOS ONE, vol. 14, no. 10, p. e0223906, Oct. 2019, doi: 10.1371/journal.pone.0223906.

[5] Design, integration, and field evaluation of a robotic apple harvester - Silwal - 2017 - Journal of Field Robotics - Wiley Online Library. Retrieved December 19, 2022 from https://onlinelibrary.wiley.com/doi/full/10.1002/rob.21715?casa_token=GEZDMJ2QKp0AAAAA%3Az-JFlyOrbYJYQWCaPkU3DMod2Sly_fUcJ3V-JKy0IvcP_nLFZo6UZ_DuLv1y-4wzy-EE3jbgeISq2YFu

[6] “Agriculture’s technology future: How connectivity can yield new growth | McKinsey.” https://www.mckinsey.com/industries/agriculture/our-insights/agricultures-connected-future-how-technology-can-yield-new-growth (accessed Apr. 20, 2023).

[7] (PDF) Field Spectroscopy to Determine Nutritive Value Parameters of Individual Ryegrass Plants. Retrieved November 18, 2022 from https://www.researchgate.net/publication/344743072_Field_Spectroscopy_to_Determine_Nu

[8] A. Hafeez et al., “Implementation of drone technology for farm monitoring & pesticide spraying: A review,” Information Processing in Agriculture, Feb. 2022, doi: 10.1016/j.inpa.2022.02.002.

[9] Y. Wageeh et al., “YOLO fish detection with Euclidean tracking in fish farms,” J Ambient Intell Human Comput, vol. 12, no. 1, pp. 5–12, Jan. 2021, doi: 10.1007/s12652-020-02847-6.

[10] U. S. Panday, A. K. Pratihast, J. Aryal, and R. B. Kayastha, “A Review on Drone-Based Data Solutions for Cereal Crops,” Drones, vol. 4, no. 3, Art. no. 3, Sep. 2020, doi: 10.3390/drones4030041.

[11] P. Daponte et al., “A review on the use of drones for precision agriculture,” IOP Conf. Ser.: Earth Environ. Sci., vol. 275, no. 1, p. 012022, May 2019, doi: 10.1088/1755-1315/275/1/012022.

[12] S. J. Kim and G. J. Lim, “A Real-Time Rerouting Method for Drone Flights Under Uncertain Flight Time,” J Intell Robot Syst, vol. 100, no. 3, pp. 1355–1368, Dec. 2020, doi: 10.1007/s10846-020-01214-z.

[13] “Roboflow: Give your software the power to see objects in images and video.” https://roboflow.com/ (accessed Apr. 20, 2023).

[14] H. Lardé et al., “Comparison of Quantification Methods to Estimate Farm-Level Usage of Antimicrobials Other than in Medicated Feed in Dairy Farms from Québec, Canada,” Microorganisms, vol. 9, no. 5, Art. no. 5, May 2021, doi: 10.3390/microorganisms9051106.

[15] “AGRAS T40 - One for All - DJI.” https://www.dji.com/t40 (accessed Apr. 20, 2023)

[16] E. E. Fujita, H. Uga, S. Kagiwada, and H. Iyatomi, “A Practical Plant Diagnosis System for Field Leaf Images and Feature Visualization,” IJET, vol. 7, no. 4.11, p. 49, Oct. 2018, doi: 10.14419/ijet.v7i4.11.20687.

[17] M. Sheeny, E. De Pellegrin, S. Mukherjee, A. Ahrabian, S. Wang, and A. Wallace, “RADIATE: A Radar Dataset for Automotive Perception in Bad Weather,” in 2021 IEEE International Conference on Robotics and Automation (ICRA), May 2021, pp. 1–7. doi: 10.1109/ICRA48506.2021.9562089.

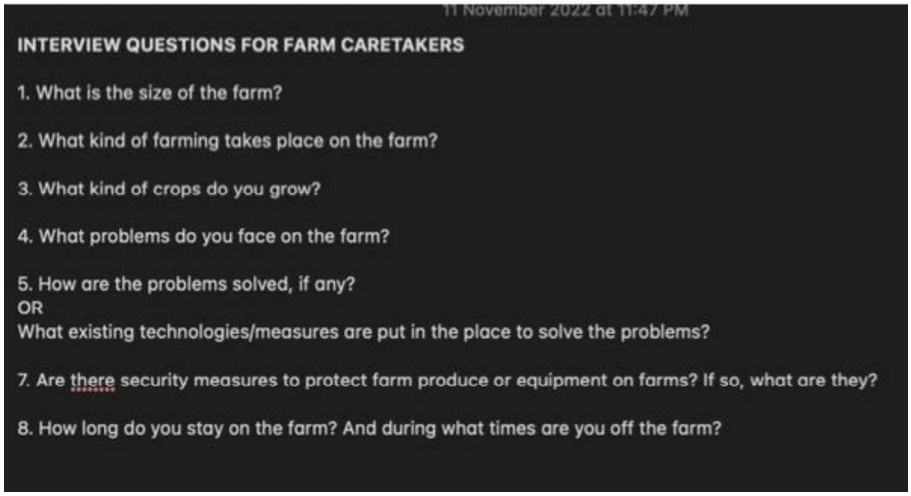
[18] Tello. Retrieved April 25, 2023 from https://www.ryzerobotics.com/tello

[19] Mavic Mini - DJI. *DJI Official*. Retrieved April 25, 2023 from https://www.dji.com/mavic-mini

‘

Appendix

Appendix A: Interview questions



11 November 2022 at 11:47 PM

INTERVIEW QUESTIONS FOR FARM CARETAKERS

1. What is the size of the farm?

2. What kind of farming takes place on the farm?

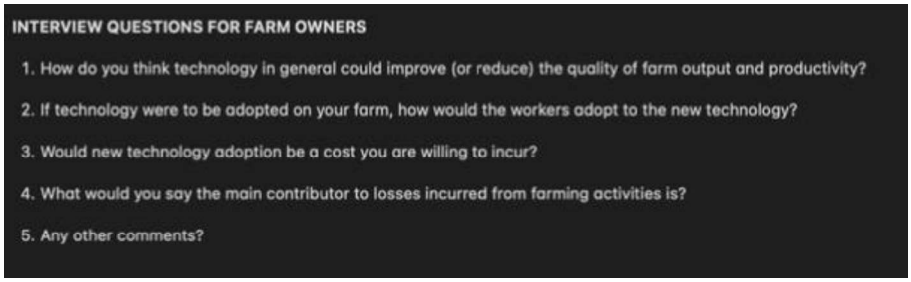
3. What kind of crops do you grow?

4. What problems do you face on the farm?

5. How are the problems solved, if any?
OR
What existing technologies/measures are put in the place to solve the problems?

7. Are there security measures to protect farm produce or equipment on farms? If so, what are they?

8. How long do you stay on the farm? And during what times are you off the farm?



INTERVIEW QUESTIONS FOR FARM OWNERS

1. How do you think technology in general could improve (or reduce) the quality of farm output and productivity?

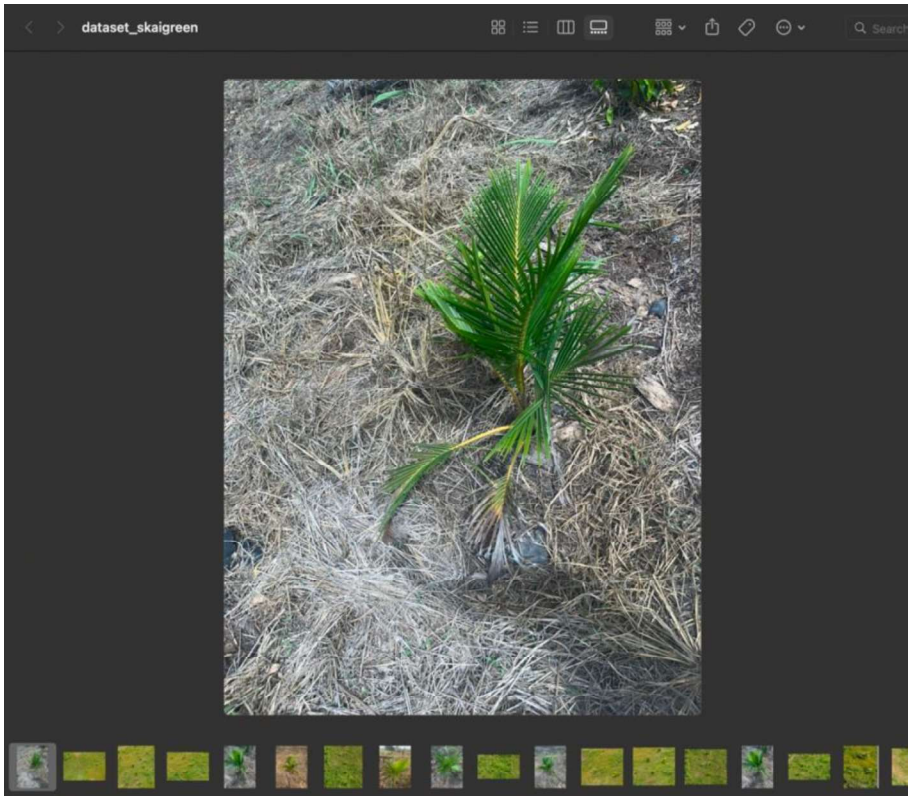
2. If technology were to be adopted on your farm, how would the workers adopt to the new technology?

3. Would new technology adoption be a cost you are willing to incur?

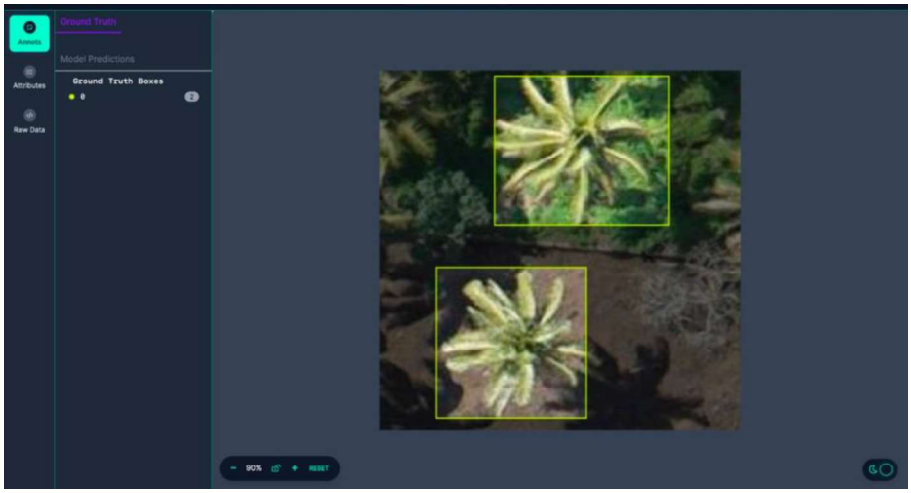
4. What would you say the main contributor to losses incurred from farming activities is?

5. Any other comments?

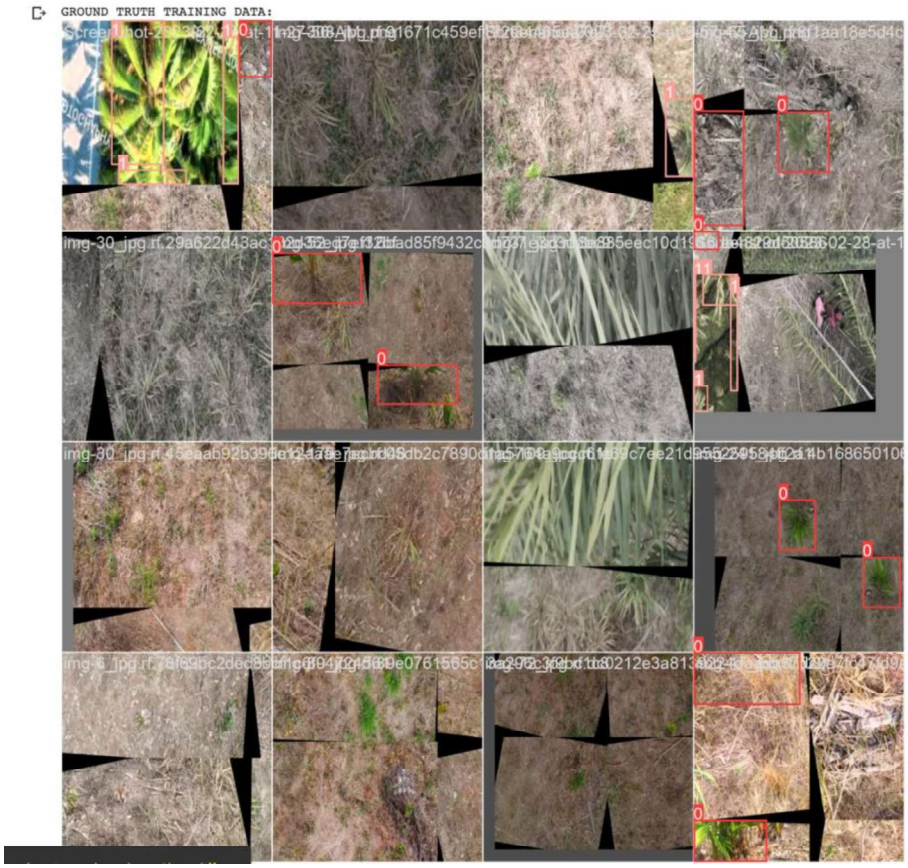
Appendix B: Coconut dataset



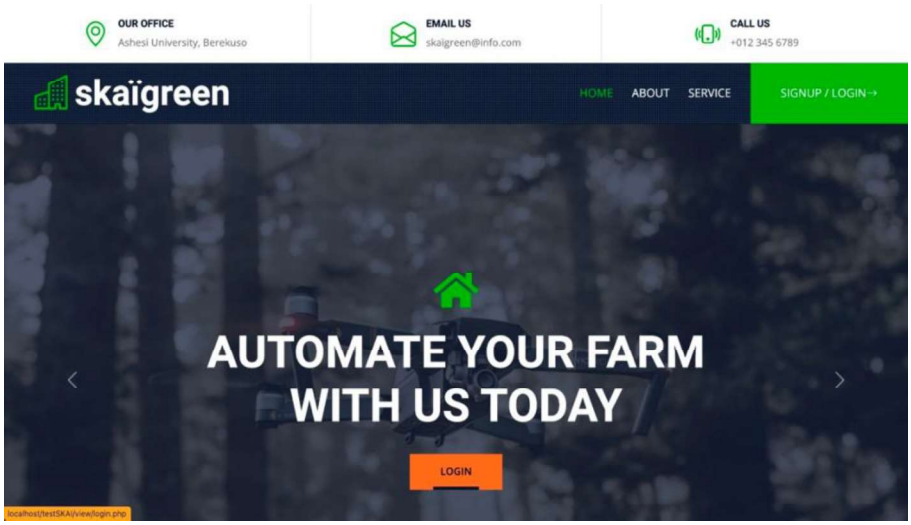
Appendix C: Dataset labelling



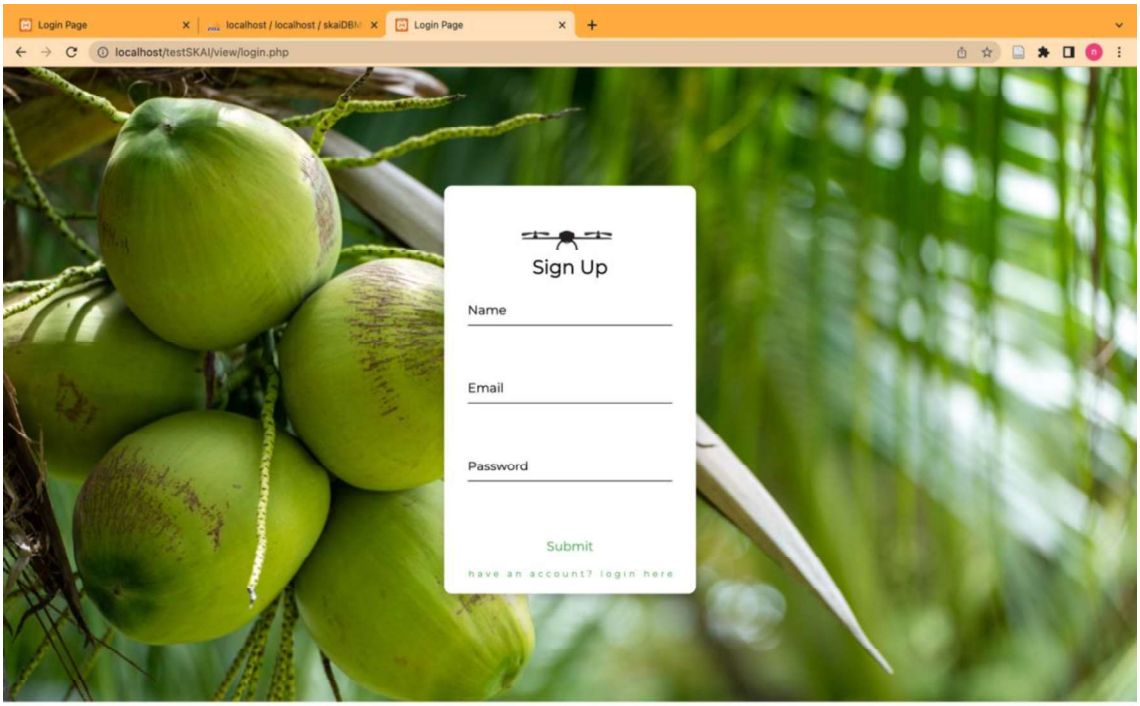
Appendix D: Data augmentation



Appendix E: Landing page



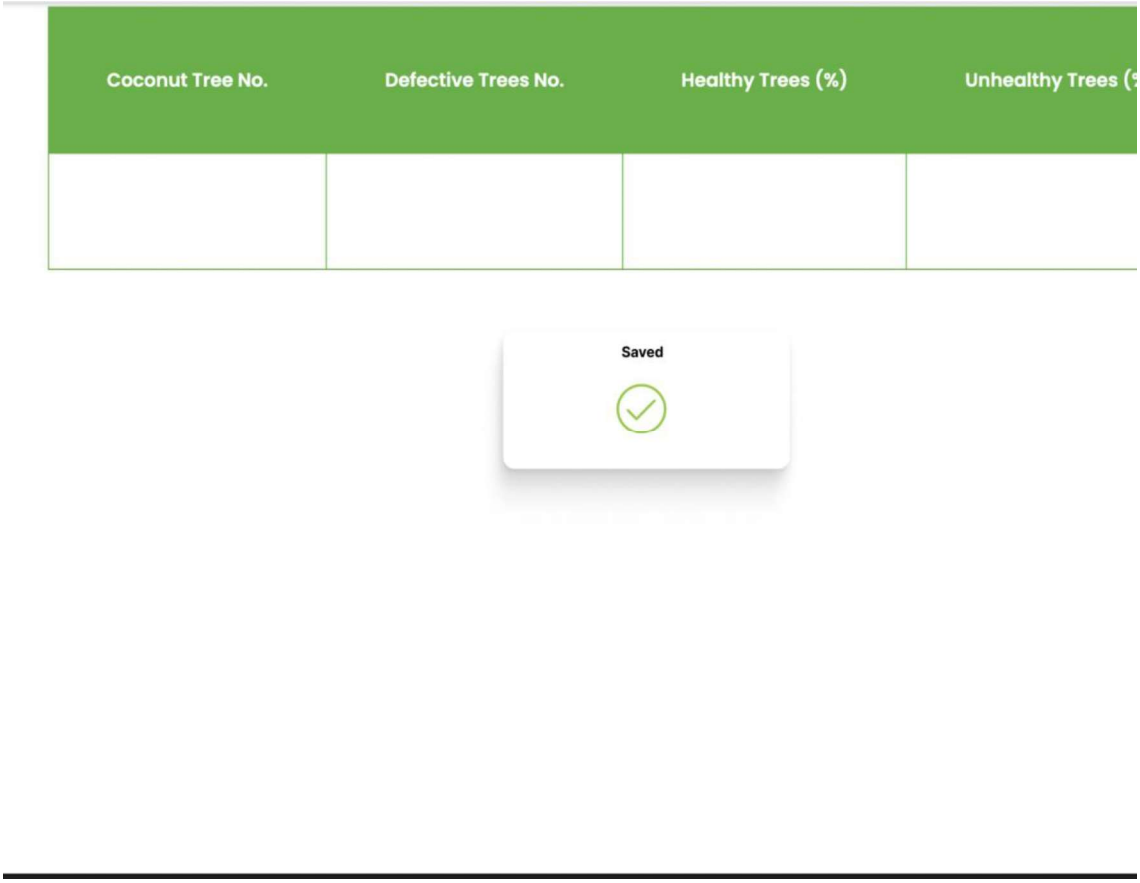
Appendix F: Registration credentials saved to database.



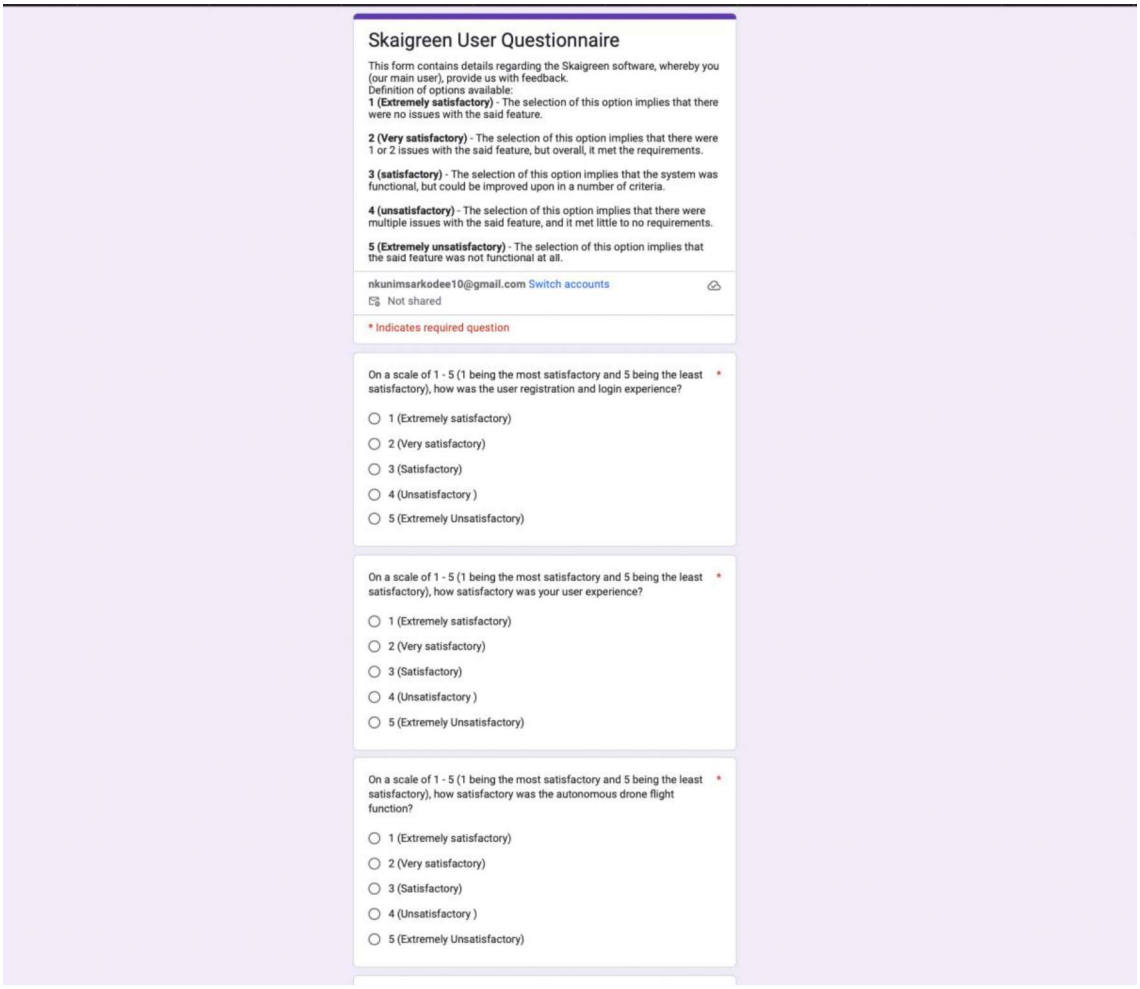
Appendix G: Registration credentials saved to database

name	email	password
Nkunim Adu-Sarkodee	emmanuel.sarkodee@ashesi.edu.gh	$2y$10$nAxfXKK85oUC.TfB129XnuhtJKV6sB.Lik7Yg6oApdK...
Ralph	ralph@vanillaearth.com	$2y$10$.idr1rIfhSOrrSNDqW162euBX.D4NzjRxaOoc6n/ZkG...
Philip Amateifio	philip@gmail.com	$2y$10$wQSjkQLkmRc7vGoUO1mZkudXQrD11UILp0LXbkPtvAp...
Paa Kwesi Addae	paa@gmail.com	$2y$10$kpfzuyQkKhS1qbvgyVqqf.zQ6vnMx5AG43Z8c/h1kTY...

Appendix H: Video footage saved on user's computer test



Appendix I : Skaïgreen user questionnaire



Skaigreen User Questionnaire

This form contains details regarding the Skaigreen software, whereby you (our main user), provide us with feedback.
Definition of options available:
1 (Extremely satisfactory) - The selection of this option implies that there were no issues with the said feature.

2 (Very satisfactory) - The selection of this option implies that there were 1 or 2 issues with the said feature, but overall, it met the requirements.

3 (satisfactory) - The selection of this option implies that the system was functional, but could be improved upon in a number of criteria.

4 (unsatisfactory) - The selection of this option implies that there were multiple issues with the said feature, and it met little to no requirements.

5 (Extremely unsatisfactory) - The selection of this option implies that the said feature was not functional at all.

nkunimsarkodee10@gmail.com Switch accounts

Not shared

* Indicates required question

On a scale of 1 - 5 (1 being the most satisfactory and 5 being the least satisfactory), how was the user registration and login experience? *

- 1 (Extremely satisfactory)
- 2 (Very satisfactory)
- 3 (Satisfactory)
- 4 (Unsatisfactory)
- 5 (Extremely Unsatisfactory)

On a scale of 1 - 5 (1 being the most satisfactory and 5 being the least satisfactory), how satisfactory was your user experience? *

- 1 (Extremely satisfactory)
- 2 (Very satisfactory)
- 3 (Satisfactory)
- 4 (Unsatisfactory)
- 5 (Extremely Unsatisfactory)

On a scale of 1 - 5 (1 being the most satisfactory and 5 being the least satisfactory), how satisfactory was the autonomous drone flight function? *

- 1 (Extremely satisfactory)
- 2 (Very satisfactory)
- 3 (Satisfactory)
- 4 (Unsatisfactory)
- 5 (Extremely Unsatisfactory)

Appendix J – Gantt chart