ASHESI UNIVERSITY

CROP LAB: CROPS DISEASE DETECTION USING DEEP LEARNING INTEGRATED WITH A MOBILE APPLICATION

APPLIED PROJECT

B.Sc. Computer Science

Silas Sangmin

2023

ASHESI UNIVERSITY

CROP LAB: CROPS DISEASE DETECTION USING DEEP LEARNING INTEGRATED WITH A MOBILE APPLICATION

APPLIED PROJECT

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

Silas Sangmin

2023

DECLARATION

I hereby declare that this Applied Project is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

………………………………………………………………………………………………

Candidate's Name:

………………………………………………………………………………………………

Date:

………………………………………………………………………………………………

I hereby declare that the preparation and presentation of this Applied Project were supervised in accordance with the guidelines on the supervision of Applied Projects laid down by Ashesi University. Supervisor's Signature:

………………………………………………………………………………………………

Supervisor's Name:

………………………………………………………………………………………………

Date:

………………………………………………………………………………………………

ACKNOWLEDGEMENTS

First and foremost, I want to express my heartfelt appreciation to God Almighty for His divine guidance throughout this journey. Secondly, I sincerely thank my Capstone supervisor, Dr. Dennis A. Owusu, for his unwavering guidance, mentorship, and invaluable feedback. His expertise and dedication have been pivotal in shaping the project into its final form.

I am also grateful to my colleagues, friends, and family for their constant support, encouragement, and motivation. Their belief in my abilities and unwavering encouragement has been a driving force behind my perseverance. I genuinely appreciate the collective efforts of all those who have supported me in this endeavor and helped me realize my academic and personal goals.

Abstract

The farming sector has primarily sourced Ghana's economy. Its importance can never be underestimated, from feeding households to serving manufacturing industries. The agricultural sector suffers a continuous annual disease attack which predominantly affects crop output. However, most large- and small-scale farmers in Ghana fail to devise ways of automating the detection and control of disease infection. Instead, they rely on a manual approach to diagnose their crops. The manual process is time-consuming, and the outcome is mostly incorrect observations and interpretations. It results in low-quality production of food crops. Therefore, to address these challenges, Crop Lab Mobile Application is designed and integrated with an efficient deep-learning model to aid farmers in identifying and treating various diseases affecting their crop plants quickly. Crop Lab provides a rapid first-aid kit for farmers to diagnose and treat crop diseases without needing external agricultural officers.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Background

Crop diseases are significant challenge farmers face in Ghana, leading to low crop yield and economic losses. It is estimated that crop diseases can cause a 30-40% reduction in crop yields globally [1]. Identifying crop diseases early is critical for effective treatment and prevention of further spread. However, traditional methods of crop disease diagnosis involve visual inspection by agricultural experts, laboratory analysis, and field tests. These methods can be time-consuming, expensive, and require specialized knowledge.

Moreover, the limited availability of agricultural experts in rural areas can make it difficult for farmers to access the support they need to identify and treat crop diseases. Crop diseases are caused by various factors, including bacteria, fungi, viruses, and environmental stress. These diseases can manifest in different ways, including discoloration, stunted growth, and wilting, making it challenging to diagnose them accurately. Accurate and timely diagnosis of crop diseases is essential for effective treatment and prevention of the spread of the disease.

In recent years, there has been an increase in the use of artificial intelligence and machine learning techniques in agriculture to improve crop yield and combat crop diseases. Deep learning, a subfield of machine learning, has shown promising results in image recognition and classification tasks, making it a suitable approach for diagnosing crop diseases based on visual symptoms. The use of mobile applications in agriculture has also gained traction in recent years. Mobile applications offer farmers a convenient and accessible platform to access information on crop diseases, treatment options, and best practices for crop management. Moreover, mobile applications can give farmers real-time information on weather patterns, soil conditions, and market prices, enabling them to make data-driven decisions.

The project aims to leverage the power of deep learning and mobile applications to provide a rapid and cost-effective solution for crop disease diagnosis in Ghana. By integrating a deep neural network model with a mobile application, farmers can quickly diagnose crop diseases by simply taking pictures of plant leaves using their smartphones. The system also provides recommendations for effective treatments for specific diseases identified. This serves as a first-aid kit for farmers, allowing them to produce high-quality crops without needing external agricultural officers, which can be expensive and challenging to access in rural areas.

1.2 Proposed Solution

1.2.1 Crop Lab

Crop Lab is a mobile application-based solution that utilizes deep learning technology to diagnose crop diseases quickly and accurately. The Application is designed to be user-friendly, accessible, and cost-effective to guide farmers in making informed decisions about their crops. Using a smartphone camera to capture images of plant leaves, Crop Lab can analyze the images and diagnose the disease affecting the plant. The system is integrated with comprehensive data on plant diseases, enabling it to identify and recommend treatments for various crop diseases. Crop Lab is a valuable tool for farmers in Ghana, where access to agricultural experts is limited, and the costs of traditional diagnosis and treatment can be prohibitively high.

1.2.2 Core Functionalities of Crop Lab

The main objectives of this project are:

- An integrated deep neural network model that can accurately detect and diagnose crop diseases from images of crops.
- A mobile application that farmers in Ghana can use to scan and feed crop images to the Deep Learning model.

- Serving as a rapid first-aid kit for farmers to diagnose and treat crop diseases without needing external agricultural officers quickly and easily.

1.2.3 The code for this project can be found via the following links:

Mobile Application

Deep Learning API

Deep Learning Colab Notebook

1.3 Related Works

In recent years, machine learning has become an increasingly important tool in agriculture, with the potential to boost crop yields and increase economic output. Several studies have investigated the use of machine learning for early plant disease detection, a critical area of research given the significant impact that disease infestations can have on crop productivity. For example, Lakshmanarao et al. [2] developed a ConvNets model that achieved an accuracy of 98.3% in detecting potato, pepper, and tomato disease, while Militante et al. [3] trained a convolutional neural network with an accuracy rate of 96.5% for diagnosing diseases in apples, corn, grapes, potatoes, sugarcane, and tomatoes. These studies demonstrate the potential of machine learning to provide quick, early detection methods for plant diseases, which can help farmers take action before disease outbreaks significantly impact their crops.

Crop Research Institute Ghana (CRI Ghana) has also identified the issue of manual crop monitoring and disease detection. In tackling this challenge, they designed a similar mobile application intended to aid farmers in detecting their crops' diseases. The Application uses OpenCV to mark out unhealthy crops and leaves a section for farmers [11]. Figure 1.1 is a demonstration of the CRI Nuru App.

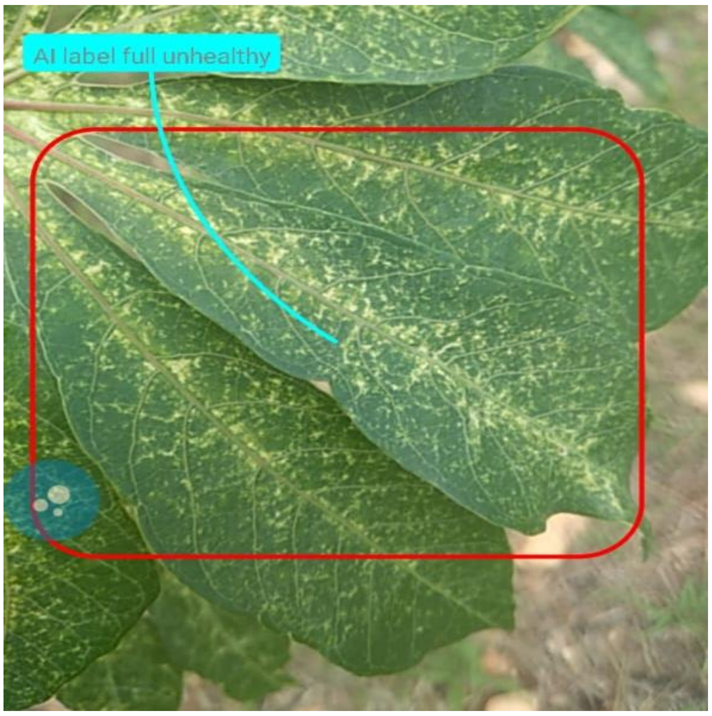


Figure 1:1 CRI Ghana Nuru APP demo: An unhealthy labeled cassava

Despite CRI's efforts in tackling this challenge, the Application fails to address the challenges farmers face. It only notifies the farmer that a portion of the crop is unhealthy but does not tell which disease affects the crop. Farmers are still stranded without knowing much about their crop conditions and how to treat them appropriately. Nuru tries to address this by providing a database of diseases to farmers to compare with their infected crop to know the condition of their crop. Still, the comparison approach can also lead to inappropriate interpretation.

Additionally, Nuru App only works better with maize crops and does not provide accurate predictions on other crops. On the other hand, Crop Lab works perfectly with eight different crops with infected diseases that give the farmers appropriate treatment measures. Crop Lab takes off the burden of allowing farmers to make manual comparisons.

Furthermore, collecting data on local crops in a particular region can be difficult, but it is essential for achieving accurate and effective results. In addressing this challenge, research was conducted to gather a suitable dataset for the project while covering Ghana's most prominent and dominant diseases affecting crops. With the correct data and machine learning tools, great model improvement will be achieved, cascading down to improve crop yields and promote economic growth in the agricultural sector.

In addition, precision agriculture, which involves using digital and information technology, sensors, microcontrollers, and communication technologies, can also help farmers to implement safe farming practices while reducing expenses. Singh and Sobti [4] suggest that precision agriculture can monitor farm conditions such as disease, soil, insect, and weather, providing farmers with greater knowledge and efficiency in producing crops. This approach has the potential to enable farmers to use resources on their farms more effectively, promote sustainability, and protect the environment. Using machine learning and precision agriculture in agriculture can revolutionize the industry, leading to increased productivity and a more sustainable future for farmers and the wider community.

Chapter 2: Requirements Analysis

The implemented solution for identifying crop diseases is an AI mobile application for farmers in Ghana. Usability requirements were obtained through survey questionnaires sent to randomly chosen farmers. This chapter outlines the functional and non-functional requirements of the system, which were influenced by the interview findings. The 4Ws and the H questions approach were essential in obtaining relevant farmer information.

The survey questionnaires provided a better understanding of the problem and the needs of the primary users. This information helped identify the type of plants and diseases to tackle, and the insights gathered guided the decision to choose crops and diseases to satisfy the user's needs. The information obtained from the farmers was used to determine the specific requirements for the Mobile Application, which will aid farmers in identifying and treating plant diseases efficiently. The user-centered approach adopted in the survey helped to ensure that the mobile Application meets the needs of the farmers and helps improve crop yields.

2.1 System Scope

The Application aims to assist farmers in detecting and diagnosing diseases affecting their crops, allowing them to take timely action to prevent significant yield loss. The project's primary objective is to provide farmers with an affordable and easy-to-use solution that can help them increase their productivity and income.

The mobile Application focuses on some of the common crops grown in Ghana, such as maize, cassava, Pepper, Potato, Orange, Tomato, Rice, and their most prevalent diseases. The scope also includes providing farmers with relevant information on preventing and managing crop diseases, including recommended treatments and best practices. The project's success is measured based on the Application's ability to effectively diagnose crop diseases, user adoption and satisfaction level, and its impact on improving crop yields and farmers' income. *Table 2* shows the scope of crops and diseases Crop Lab covers.

2.2 Users/Clients

The primary users of the AI mobile application for identifying crop diseases are farmers in Ghana. These users are the clients who will benefit from the system's features and functionalities. As such, the system was designed with their needs and requirements in mind to ensure that it is easy to use and provides accurate results. By targeting this specific user group, the system addresses a critical need in Ghana's agricultural sector, helping farmers improve their crop yields and overall livelihoods. Keeping these users at the forefront of the development process ensured the system met and fulfilled their expectations.

2.3 Functional Requirements

The system shall allow farmers to scan plant pictures with cameras using their mobile devices. Farmers will scan crop leaves with their smartphones; if the plant is infected, the name and type of infection will be identified. This requirement aims to provide farmers with an easy-to-use tool for detecting plant diseases. Farmers can quickly scan pictures of their crops using their mobile devices' cameras, and the system will identify the type of infection affecting the plant. This feature will help farmers detect plant diseases early, enabling them to take action quickly and prevent the spread of the disease.

The system shall accept image uploads. In addition to scanning plant pictures, farmers can also upload images of plants from their internal or external storage devices. This feature will allow farmers to identify diseases in plants that have been stored in their devices, such as images taken on previous visits to their farm. This requirement aims to provide a flexible solution that allows farmers to use the system most conveniently.

The system should be able to detect the specific diseases affecting plants. Precision is vital for this project, and the main goal is appropriately identifying the actual diseases without flaws or bias. This requirement focuses on the system's ability to identify specific diseases affecting plants. It is essential to ensure the system can accurately detect the disorders, as the entire project's success depends on its accuracy. To achieve this requirement, the system will be trained using a dataset of images of plants infected with various diseases.

Farmers should receive treatment recommendations for every disease detected. After identifying the disease, what is next? Farmers will also receive detailed step-by-step instructions on how to treat the disease. This will help reduce over-reliance on chemical substances for every illness identified, as seen with the traditional crop treatment approach. This requirement aims to provide farmers with recommendations for treating the diseases detected in their crop plants. This feature will help farmers to reduce over-reliance on chemical substances when treating their plants.

The system shall cover at least seven crops cultivated in Ghana. Even though there is limited data on local produce, the system shall include at least seven Ghanaian-based crops. This requirement focuses on the system's ability to cover crops cultivated in Ghana. This feature will help farmers to identify diseases affecting their crops and improve their productivity.

For each crop, the system shall cover at least two diseases. Each class of crops can be affected by different types of diseases. Therefore, the system should be able to handle more than one disease affecting each crop. This requirement focuses on the system's ability to cover multiple diseases affecting each crop. It is essential to ensure that the system can identify different diseases that affect crops to provide farmers with accurate and helpful recommendations for treating the diseases.

2.4 Use Case Diagram, Deep Learning Model, and Mobile Application

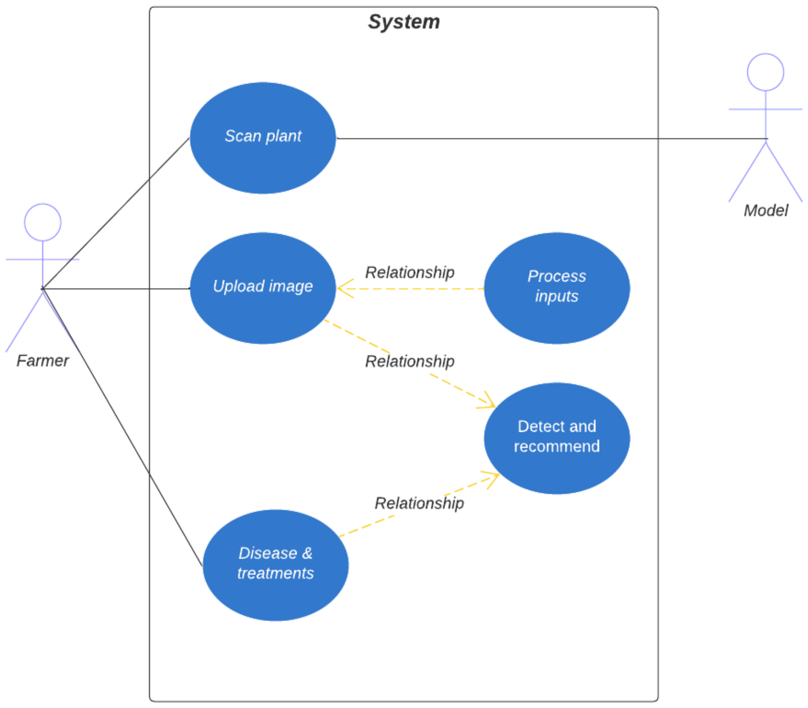


Figure 2:1 Use Case Diagram, Deep Learning Model, and Mobile Application

2.5 Non-functional Requirements

The system should be compatible with both Android and IOS devices. The system should also be cross-platform independent and work across multiple platforms and operating systems. This feature will help farmers use the system regardless of their device.

Security: The System shall be secured and shall protect users' privacy. This requirement ensures the system is secure and protects users' privacy. The system will be designed with security in mind and appropriate measures.

Accessibility: The System shall be accessible and available to every farmer using a smartphone. The system should be accessible and user-friendly for farmers with varying technological experiences. It should be easy to navigate and operate, with clear instructions and prompts for users to follow.

User-Friendliness: The System should be user-friendly. The system should be designed with the user in mind. It should be intuitive, easy to navigate, and user-friendly. The interface should be easy to understand and not require significant technical knowledge or expertise.

Availability: The System needs to be available for users to all farmers within the country. And should always be available for daily use. The system should always be available to farmers without downtime or interruptions. The system should be designed to function optimally and efficiently, even under high usage and heavy traffic.

Legal Compliance: The System shall comply with Ghana's laws and the cyber community rules and regulations. The system should adhere to all relevant legal and regulatory requirements within Ghana. It should also follow the best cybersecurity and data protection practices to ensure the system is secure and trustworthy.

2.6 Flow chart Diagram

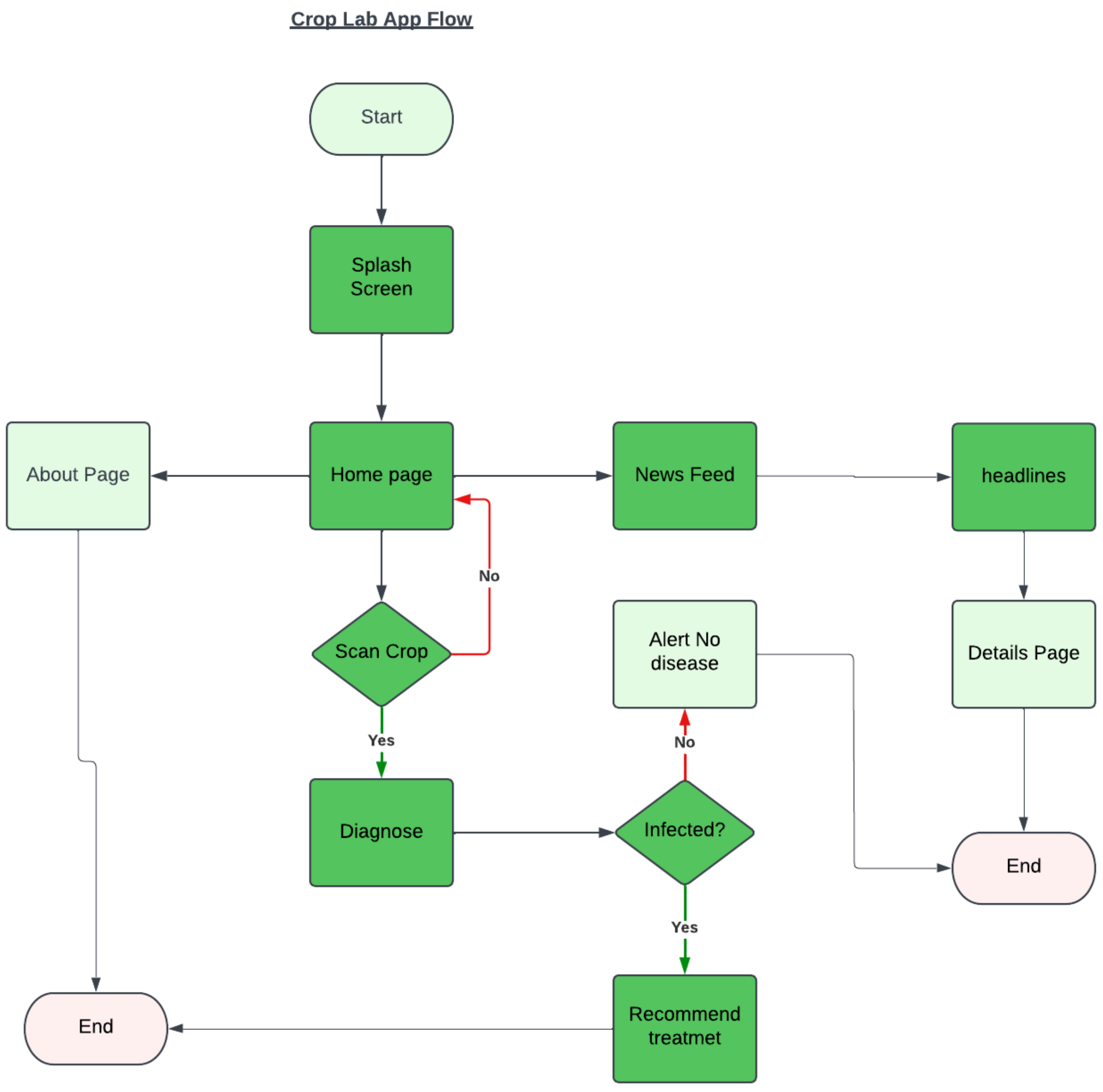


Figure 2:2 Flow Chat for Crop Lab

Chapter 3: Architectural Design

3.1 Client-Server Architecture and Model-View-Controller

According to Sommerville [5], the client-server architecture is a system that comprises a group of services provided by different servers that clients can access. In the case of Crop Lab, the services provided by the Application are the various functionalities it offers, such as crop analysis, disease diagnosis, and treatment recommendation. At the same time, the clients are the farmers who access these services using their mobile devices. The client-server model allows the clients to request the servers for the required services. For example, when farmers want to examine their crops, they can click the "Scan crop" button on the Application's home page. This click captures the image and initiates an HTTPS request to the server hosted on the Internet. The server processes the request and sends a JSON response to the client, displaying the results in a user-friendly format on the "Disease Information" page.

The client-server model offers a degree of independence and separation between the client and the server. This architecture allows the implementation of independent services on a single server or multiple servers. In the case of Crop Lab, the mobile Application can also be considered the client, which is separate and independent from the server, which is hosted remotely. This independence allows adding new features to the Application without affecting the server and vice versa. Additionally, this modular design offers opportunities for code reuse and maintenance. A potential disadvantage of the client-server architecture is that the system's performance is highly dependent on network strength, and performance may be unpredictable.

The Client-Server architecture handles communication between the mobile Application and the server. The client refers to the mobile Application running on the farmers' devices. In contrast, the server refers to the backend system responsible for processing the scanned images, identifying the diseases, and returning the results to the clients. The Client-Server architecture is a popular design pattern separating the Application into two parts. The client handles the presentation logic and user interaction, while the server handles the Application's data processing and storage. This separation ensures that the Application's logic is modular, easily scalable, and more secure. Figure 3.1 is an illustration of crop lab interaction via the client-server architecture.

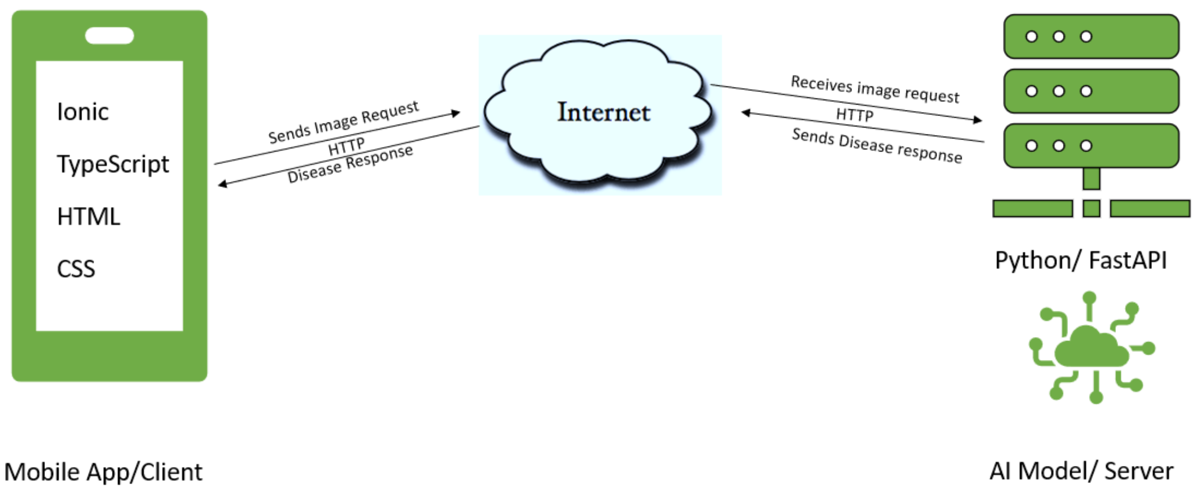


Figure 3:1 Client-Server Architecture for Crop Lab- Two-Tier

3.2 Model-View-Controller (MVC)

On the other hand, the Model-View-Controller (MVC) architecture separates the Application's data, user interface, and control logic [6]. The model represents the data and associated business logic in this design pattern, while the view represents the user interface. The controller acts as an intermediary between the model and the view, interpreting the user's actions and updating the model or view accordingly. The Model-View-Controller design pattern is used within the mobile Application of Crop Lab. The view represents the user interface (UI) the farmer interacts with, the model handles the data and logic related to the diseases, and the controller handles the communication between the farmer and the deep learning model. The MVC architecture makes the Application more modular and easier to maintain. Any changes made to the model or view do not affect the other components, and it is easier to debug and test the Application's components individually.

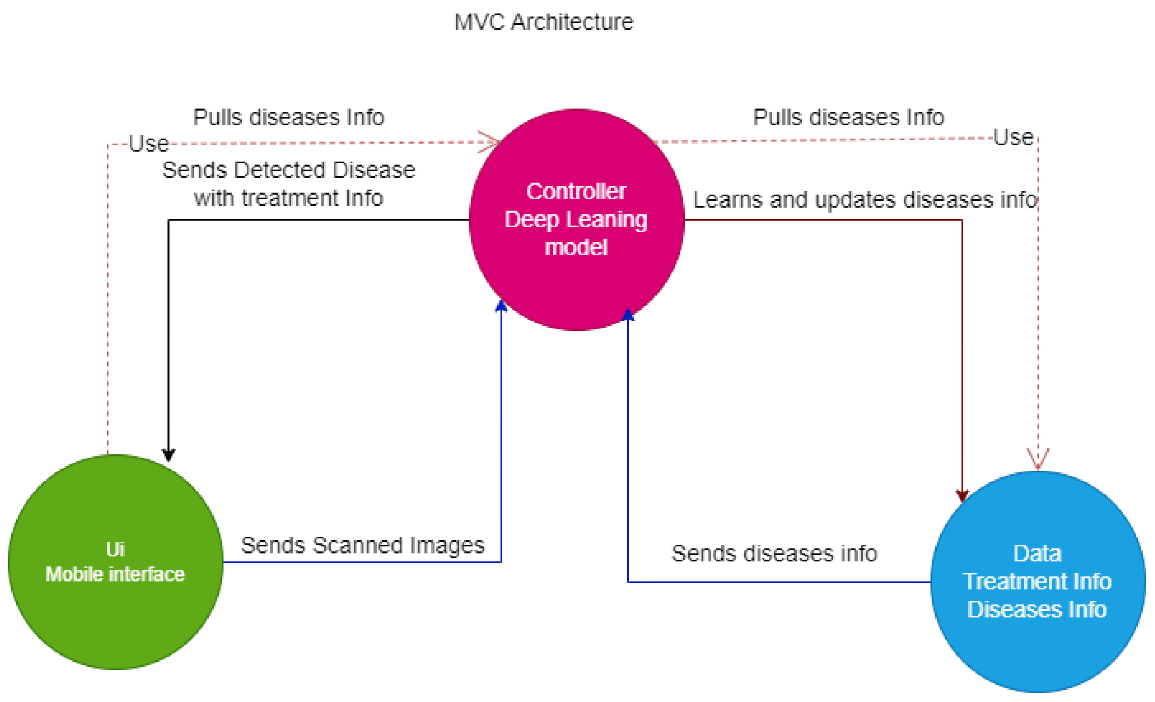

MVC Architecture
Pulls diseases Info
Use
Sends Detected Disease with treatment Info
Controller Deep Leaning model
Pulls diseases Info
Use
Learns and updates diseases info
Ui Mobile interface
Sends Scanned Images
Sends diseases info
Data Treatment Info Diseases Info

Figure 3:2 The MVC Architecture of Crop Lab

Combining the Client-Server architecture and the Model-View-Controller design pattern ensures this project is more scalable, secure, and maintainable.

Chapter 4: Implementation

4.1 Technologies for Crop Lab Mobile Application

4.1.1 Ionic Framework

Ionic is a popular open-source framework used for building cross-platform mobile applications. It is built on Angular, a robust JavaScript framework for building web applications. One of the significant advantages of using Ionic is that it allows developers to write code once and deploy it across different platforms, such as iOS, Android, and the Web. This means developers do not need expertise in different programming languages, such as Kotlin for Android and Swift for iOS. This saves time, effort, and resources for the development team.

Ionic also offers a wide range of pre-built UI components, which can be customized to fit the Application's requirements. These components are designed to provide a seamless user experience across different platforms. The framework also supports common mobile app features such as camera access, GPS location, and storage [7]. This helps me as a developer to focus on building the Application's core functionality without worrying about the low-level details of mobile development. Ionic is an excellent choice for building mobile applications, particularly those that need to run on multiple platforms. Table 4.1 is a comparison between Ionic (Angular) and Kotlin/Swift:

Table 4:1 Ionic Framework Versus Kotlin/Swift

Criteria	Ionic (Angular)	Kotlin/Swift
Programming Language	TypeScript, HTML, CSS	Kotlin (Android), Swift (iOS)
Platform	Cross-platform (Android, iOS, Web)	Native (Android, iOS)
Development	Faster development, single codebase for multiple platforms	Longer development, the separate codebase for each platform
Performance	Slower than native apps	Faster than cross-platform apps
UI/UX	Flexible and customizable, it uses web technologies	More native look and feel
Plugins/Tools	Extensive library of plugins and tools	Limited plugins and tools
Cost	Lower development cost, higher maintenance cost	Higher development cost, lower maintenance cost
Market Share/Users	Growing in popularity	Dominates mobile app development

Based on the comparison, Ionic (Angular) is a more suitable technology for this project. It supports Android and iOS and offers a faster development process with a single codebase for multiple platforms. This will save time and reduce development costs.

4.1.2 Crop Lab Mobile Application

4.1.2.1 Home Page

The home page of the Application is the landing page for the farmers. The user/farmer will see a call-to-action button named "Scan Crop" on the home page. Clicking on the button will open a pop-up for a camera or an image upload option. Either option will allow them to feed the send an image to the deep learning model on the server to get the necessary infection status; in addition to the scanning button, sections like recently scanned images and weather forecast guide the farmers' activities for the day. Figures 4.1 show the home page and the camera view, respectively.

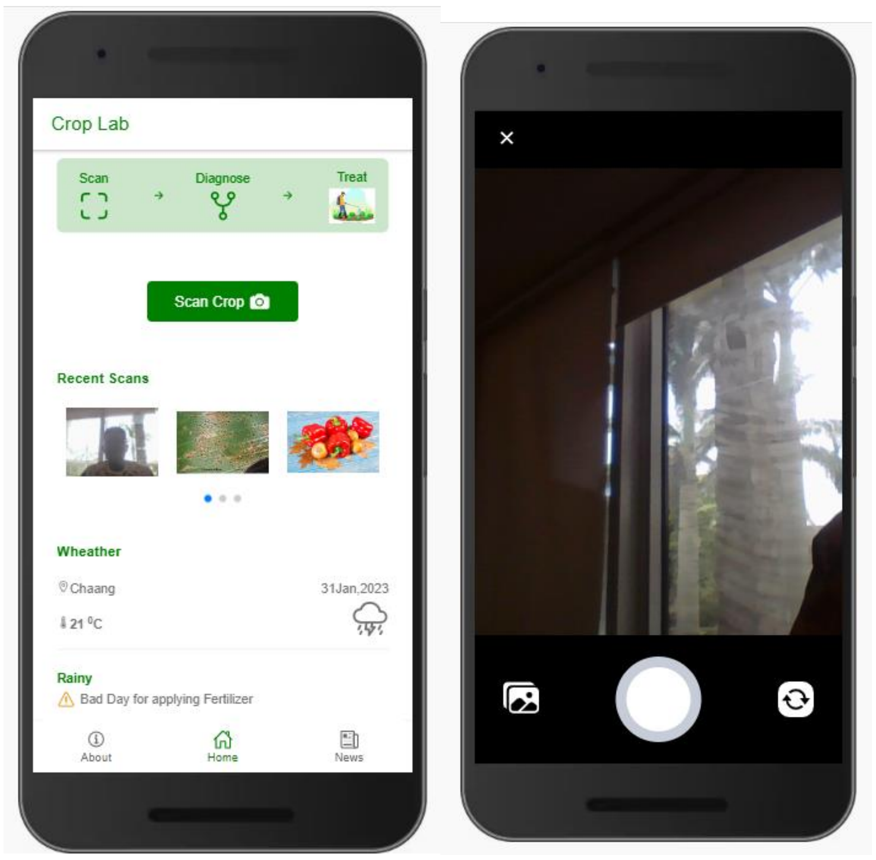


Figure 4:1: Home Page

4.1.2.2 Diagnose Page

The diagnosis page shows the status of the diagnosis. It displays the current image that has been diagnosed and some spinners indicating the processing of the image. It is meant to keep users updated on the waiting response. Figure 4.3 shows a screenshot of the diagnose page.

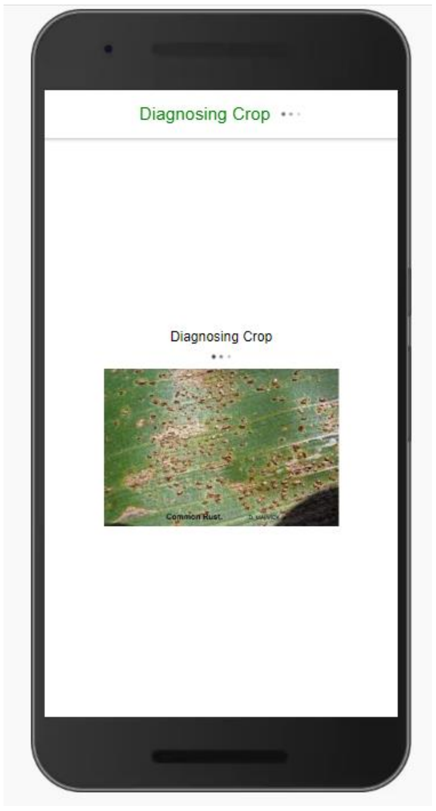
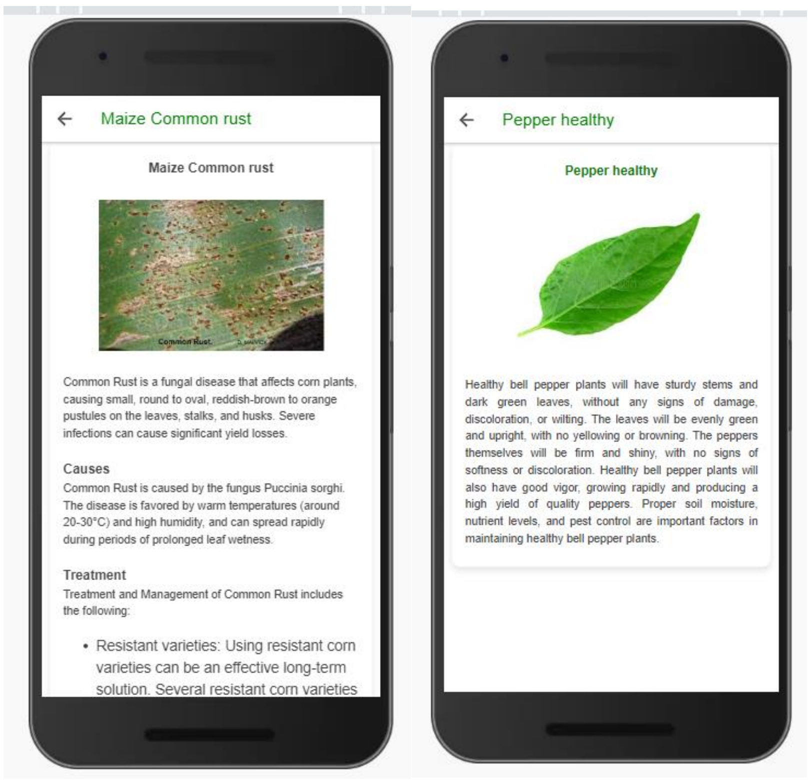


Figure 4:2: Diagnose Page

4.1.2.3 Diseases Treatment Page

This page displays information on the disease detected or otherwise. It also shows the recommended treatments and measures. Each disease's information is stored in JSON format. The Application fetches the appropriate data based on the detected disease using the MVC architecture's controller logic. Figure 4.4 shows this page.


Maize Common rust

Maize Common rust

Common Rust is a fungal disease that affects corn plants, causing small, round to oval, reddish-brown to orange pustules on the leaves, stalks, and husks. Severe infections can cause significant yield losses.

Causes

Common Rust is caused by the fungus Puccinia sorghi. The disease is favored by warm temperatures (around 20-30°C) and high humidity, and can spread rapidly during periods of prolonged leaf wetness.

Treatment

Treatment and Management of Common Rust includes the following:

- Resistant varieties: Using resistant corn varieties can be an effective long-term solution. Several resistant corn varieties

Pepper healthy

Pepper healthy

Healthy bell pepper plants will have sturdy stems and dark green leaves, without any signs of damage, discoloration, or wilting. The leaves will be evenly green and upright, with no yellowing or browning. The peppers themselves will be firm and shiny, with no signs of softness or discoloration. Healthy bell pepper plants will also have good vigor, growing rapidly and producing a high yield of quality peppers. Proper soil moisture, nutrient levels, and pest control are important factors in maintaining healthy bell pepper plants.

Figure 4:3 Disease /Treatment Pages

4.2 Technologies for the Deep Learning Model

4.2.1 PyTorch

Moving on to the deep learning model, PyTorch was used for the model implementation, and it was trained using Google Colab, a cloud-based development environment. PyTorch was chosen due to my knowledge of the framework, flexibility, ease of use, and excellent support for deep learning tasks. Google Colab was selected because it provides free access to powerful GPUs required for training deep learning models.

PyTorch is an open-source machine-learning library built on Python, one of the most popular programming languages in the world. This means that developers proficient in Python can quickly learn and use PyTorch. Python's simplicity and easy-to-learn syntax make it an ideal language for machine learning and data science applications. Additionally, Python has a vast and active developer community, which has contributed to the growth and popularity of

PyTorch. PyTorch's Python-based architecture provides several advantages over other deep learning frameworks. PyTorch is designed to be user-friendly and offer a simple and intuitive interface for building deep learning models.

4.3 Datasets

The crops and the type of diseases have been selected based on farmers' needs and data availability. Two main datasets were identified to contain these crops and diseases suitable for the task. These include PlantVillage (52,000+ data) and PlantDoc (approximately 17,000). Cassava and Rice data were obtained from a different source and merged with the other dataset from PlantVillage. The PlantDoc dataset was not used because it contains similar images as the PlantVillage. Hence the PlantVillage was used. The datasets contain some crops, such as strawberries, apples, and the like, which are not geographically cultivated in Ghana and were not considered for this project. During the data preprocessing phase, these foreign crops were removed. The final dataset used for training and validating the model was 52,747. Table 4.2 shows each crop and the diseases covered.

Table 4:2 Crops and their respective diseases covered by the system

Crop Plant	**Diseases**
1. Maize	i. Common rust ii. Northern Leaf Blight, iii. Cercosporin leaf spot, iv. gray leaf spot,
2. Tomato	i. Bacterial spot, ii. Leaf Mold, iii. Septoria leaf spot, Spider mites, iv. Two-spotted spider mite, v. Target Spot, vi. Yellow Leaf Curl Virus, vii. mosaic virus, viii. Early blight, Late blight
3. Pepper	i. bell Bacterial spot
4. Orange	i. Huanglongbing
5. Potato	i. Early blight, ii. late blight
6. Cassava	i. Cassava Bacterial Blight (CBB) ii. Cassava Brown Streak Disease (CBSD) iii. Cassava Green Mottle (CGM) iv. Cassava Mosaic Disease (CMD)
7. Rice	i. Rice Brown Spot ii. Rice Hispa iii. Rice Leaf Blast

4.4 Data Preprocessing

Data Preprocessing was one of the necessary steps to preprocess the crop dataset and prepare it for training the crop detection deep learning model. As part of the processing, Data augmentation was done. Data augmentation is an essential technique in machine learning, particularly in computer vision, which involves manipulating and transforming existing training data to create new samples. The importance of data augmentation lies in its ability to increase the size and diversity of the training dataset, which can help prevent overfitting and improve the model's ability to generalize new and unseen data [8]. By augmenting the existing dataset with variations of the same data, such as rotations, flips, and scaling, the model can learn to recognize patterns in the data regardless of their orientation, position, or size.

Furthermore, data augmentation can address a class imbalance in the dataset, where some classes have fewer samples than others, by generating synthetic samples of the underrepresented classes. Data augmentation is a crucial tool for improving the performance and robustness of machine learning models. For the training dataset, the following transformations are applied:

RandomResizedCrop - This transformation crops a random section of the cropped image and resizes it to 224 x 224 pixels. Random cropping helps create a different set of images that can help improve the model's ability to generalize to new images.

RandomHorizontalFlip - This transformation randomly flips the image horizontally with a probability of 0.5. This also helps in creating a more diverse set of images.

ToTensor - This transformation converts the image to a tensor.

Normalize - This transformation normalizes the pixel values of the image. The mean and standard deviation values used for normalization were obtained from the ImageNet dataset.

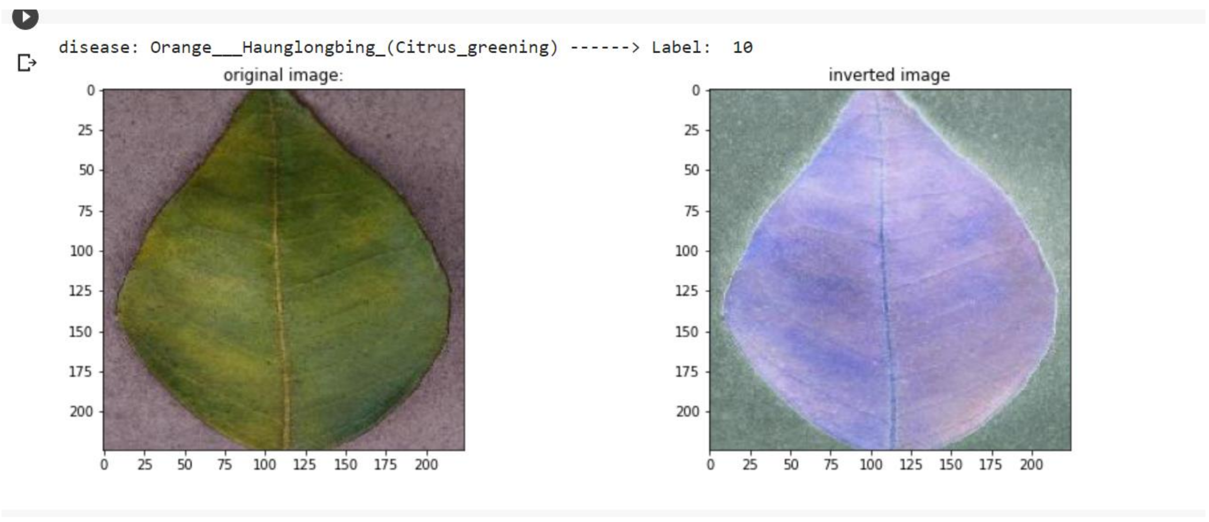
The formula for normalization typically involves subtracting the mean value of a dataset from each data point and then dividing the result by the standard deviation of the dataset. Mathematically, this can be represented as:

(x - mean) / standard deviation.

Where "x" is a data point, "mean" is the mean value of the dataset, and "standard deviation" is the standard deviation of the dataset. This process results in a new dataset with a mean of 0 and a standard deviation of 1, which can be easier to work with in some machine learning algorithms.

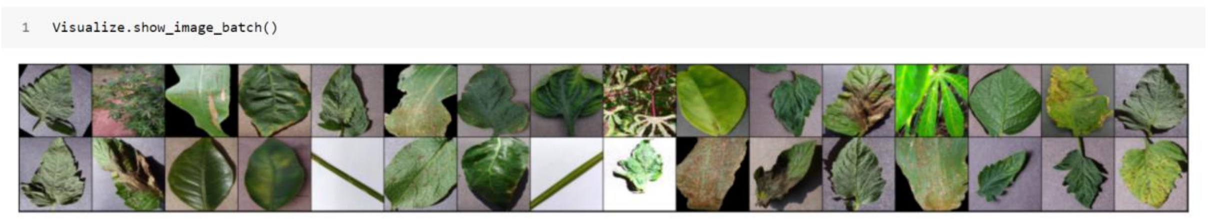
4.5 Data Visualization

Data visualization is essential during the preprocessing and training phases of the deep learning model. During the preprocessing phase, visualizing the data allowed for easy identification of data shape and possible imbalances in the data, which informs the data cleaning and preprocessing steps. Visualization is done for images, training progress, losses, and accuracies with the Python Matplotlib library. The following figures show some visualizations of the images.



```
disease: Orange___Haunglongbing_(Citrus_greening) ------> Label:  10
```

Figure 4:4 Sample Image Visualization with Inverted Image



```
Visualize.show_image_batch()
```

Figure 4:5 Image Batch Visualization

4.6 Deep Learning Model for Crop Lab

The model used for this project is the ResNet-18 model. ResNet stands for Residual Network and is a type of neural network architecture that utilizes residual connections, which allow information to bypass some layers and flow directly to later layers. This architecture enables the network to be deeper without the problem of vanishing gradients.

ResNet-18 is a specific ResNet architecture variant consisting of 18 layers. It has 11 million parameters and was pre-trained on the ImageNet dataset, an extensive database of labeled images.

The advantages of ResNet over other deep learning neural networks or CNN models are:

- ResNet can be deeper without encountering the problem of vanishing gradients. This means that deeper models can be trained without compromising performance.
- Residual connections allow for better information flow through the network and help avoid overfitting.
- ResNet has been shown to outperform other state-of-the-art models on several benchmark image classification tasks.

4.7 Model Training

The training was initially done on a custom-built Convolutional Neural Network. However, some challenges encountered with the custom CNN included overfitting, low validation accuracy (around 65%), and time-consuming during training. Hence, transfer learning was adopted. Transfer learning is a technique used in machine learning and deep learning where a pre-trained model on a large dataset is fine-tuned on a different dataset for a specific task. The idea is to use the knowledge gained from the pre-training to improve the model's performance on a new, related task. Transfer learning has several advantages over training a model from scratch. First, it can significantly reduce the time and computational resources needed to train a model, as the pre-trained model has already learned the underlying patterns and features of the data. Second, it can help to overcome the problem of limited data, as the pre-trained model has been trained on a large dataset, making it better at generalizing to new, unseen data. Finally, transfer learning can improve the overall performance of a model, as the pre-trained model has already learned to recognize complex patterns and features that may be useful for the new task.

4.7.1 Hyper Parameters

Learning rate: This hyperparameter determines the step size at each iteration while moving toward a minimum loss function during the training of the deep learning model. An initial learning rate of 0.001 was used. Then a learning rate scheduler was used to update the learning rate of the optimizer during training. The learning rate scheduler reduces the learning rate by a gamma factor for every step-size epoch. An optimizer was then used to update the neural network's weights during training, an instance of the stochastic gradient descent (SGD) optimizer with a momentum of 0.9. In this case, the learning rate will be multiplied by 0.1 every 7 epochs (step size=7 and gamma=0.1), a common practice to prevent the model from overfitting and improve its generalization ability.

Batch size: This hyperparameter determines the number of samples for each batch to train a neural network. The training process used a batch size of 32 because a larger batch size is computationally expensive. 32 was used due to the limited GPU on the free version of Google Colab.

The number of epochs: This hyperparameter determines the number of times the entire training dataset is passed through the neural network during training. An epoch size of 15 was used due to limited time.

Training, testing, and Validation Dataset sizes: The training dataset was 80% of the dataset. Thus, 42190 of 52747, the remaining 20%, was reserved for validation. That 10557 of 52747. The data size for testing was also 2,594 images kept in a test folder hidden from the model during the training and evaluation phases. The testing data was later used to evaluate how the model would perform with a new dataset. The test data was also used to obtain and plot a confusion matrix, F1 Score, precision, and recall which will be discussed later. The table below summarizes the datasets for training, validation, and testing.

Table 4:3 Summary of dataset sizes

Dataset	Size
Training	39,596
Validation	10,557
Testing	2,594

Class/label Size: Crop lab covers 7 crops with 30 labels. Each label represents a disease, except for one class called background without leaves, which enables the model to detect when invalid crop leaves are captured.

4.8 Model Evaluation & Testing

4.8.1 Overfitting

One major problem during the training and evaluation phase was overfitting. Overfitting is a common issue in machine learning and statistical modeling where a model performs very well on the training data but fails to generalize to new, unseen data. In other words, an overfit model may have learned to memorize the training data instead of learning the underlying patterns, resulting in poor performance on new data. The model performed well on the training data with a training accuracy of 90%. However, it failed to generalize the validation data, leading to poor performance and inaccurate predictions. Figures 4.7 show the graphs on the overfitting model, illustrating the loss and accuracy plotted against the epochs

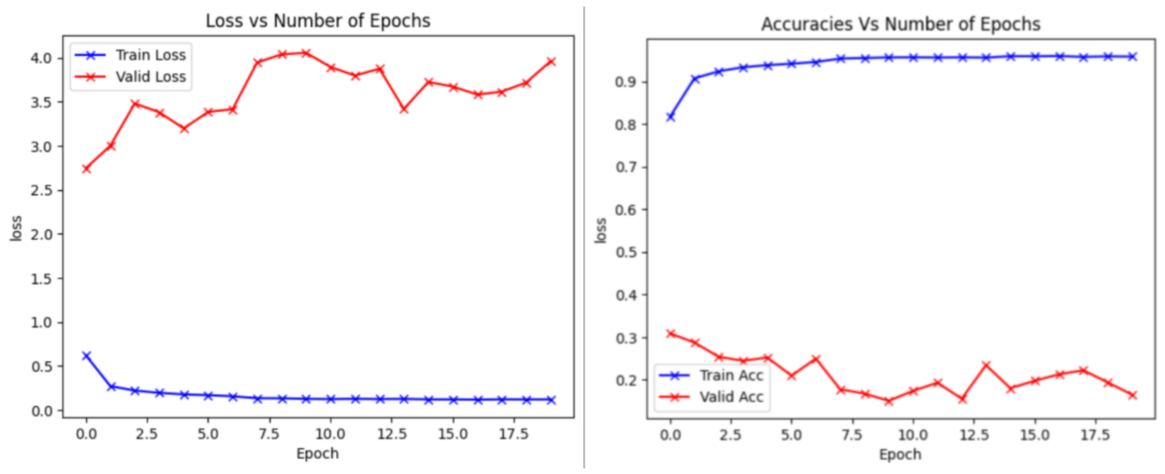
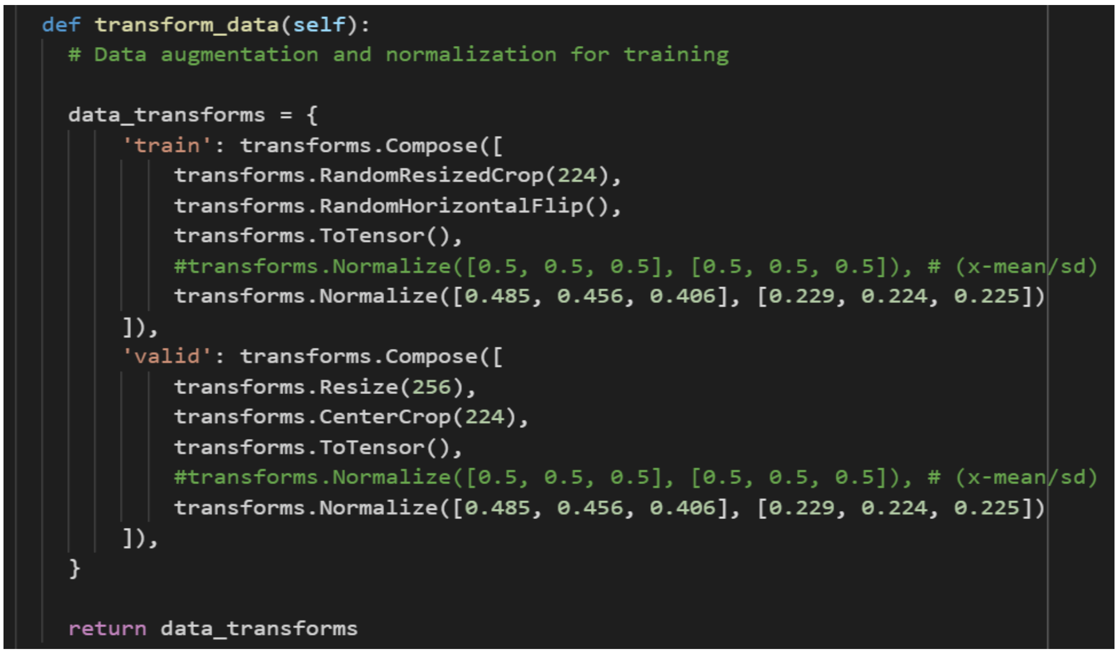


Figure 4:6 Plot of losses and accuracies against epochs

To mitigate overfitting, various techniques were employed, such as simplifying the model architecture from restNet50 to restNet18, changing the batch size to 32, and adopting a scheduler for the learning rate. Proper augmentation techniques were also applied to the training and validation datasets.

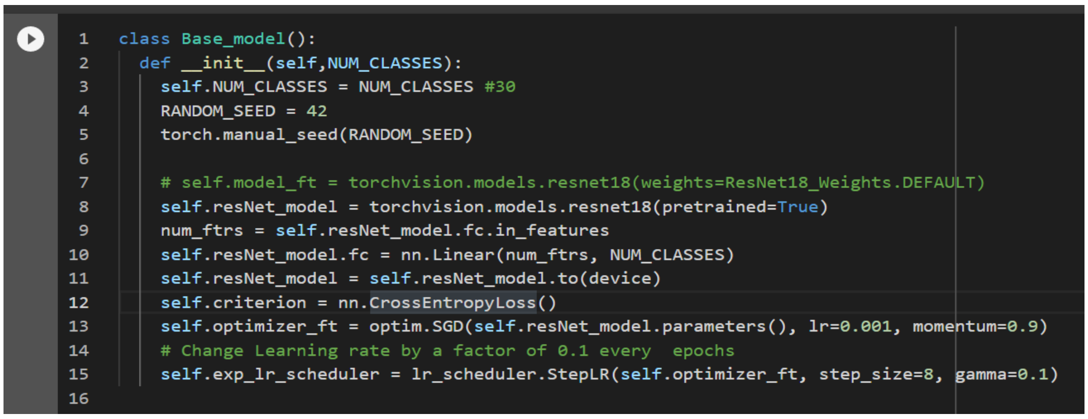


```python
def transform_data(self):
  # Data augmentation and normalization for training

  data_transforms = {
      'train': transforms.Compose([
          transforms.RandomResizedCrop(224),
          transforms.RandomHorizontalFlip(),
          transforms.ToTensor(),
          #transforms.Normalize([0.5, 0.5, 0.5], [0.5, 0.5, 0.5]), # (x-mean/sd)
          transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
      ]),
      'valid': transforms.Compose([
          transforms.Resize(256),
          transforms.CenterCrop(224),
          transforms.ToTensor(),
          #transforms.Normalize([0.5, 0.5, 0.5], [0.5, 0.5, 0.5]), # (x-mean/sd)
          transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
      ]),
  }

  return data_transforms
```

Figure 4:7 Data argumentation and normalization to prevent overfitting



```python
class Base_model():
  def __init__(self,NUM_CLASSES):
    self.NUM_CLASSES = NUM_CLASSES #30
    RANDOM_SEED = 42
    torch.manual_seed(RANDOM_SEED)

    # self.model_ft = torchvision.models.resnet18(weights=ResNet18_Weights.DEFAULT)
    self.resNet_model = torchvision.models.resnet18(pretrained=True)
    num_ftrs = self.resNet_model.fc.in_features
    self.resNet_model.fc = nn.Linear(num_ftrs, NUM_CLASSES)
    self.resNet_model = self.resNet_model.to(device)
    self.criterion = nn.CrossEntropyLoss()
    self.optimizer_ft = optim.SGD(self.resNet_model.parameters(), lr=0.001, momentum=0.9)
    # Change Learning rate by a factor of 0.1 every  epochs
    self.exp_lr_scheduler = lr_scheduler.StepLR(self.optimizer_ft, step_size=8, gamma=0.1)
```

Figure 4:8 ResNet18 Model using transfer learning

After the hyperparameters tuning and training of the model on 15 epochs, the model achieved a training accuracy of 96%, the validation accuracy was 96%, and test accuracy was 94%. Figure 11 and Figure 12 show the graphs for both losses and accuracies for training and validation.

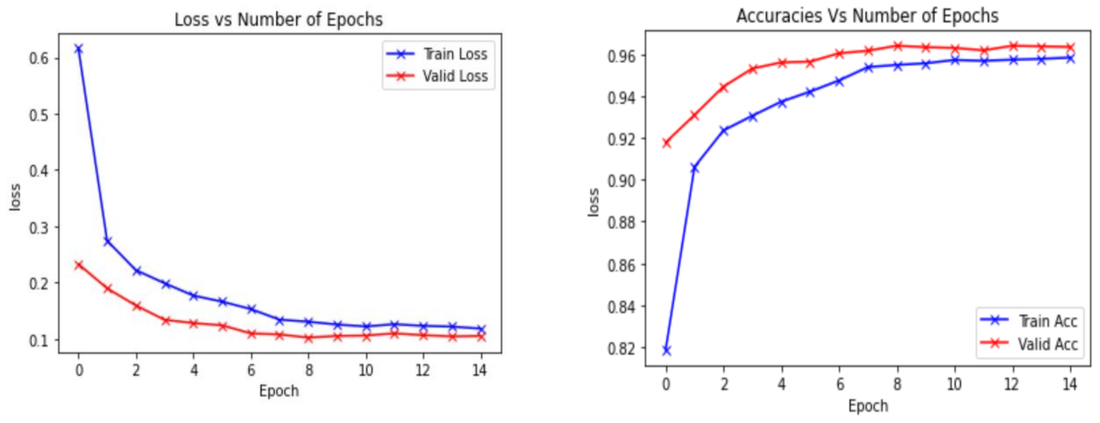


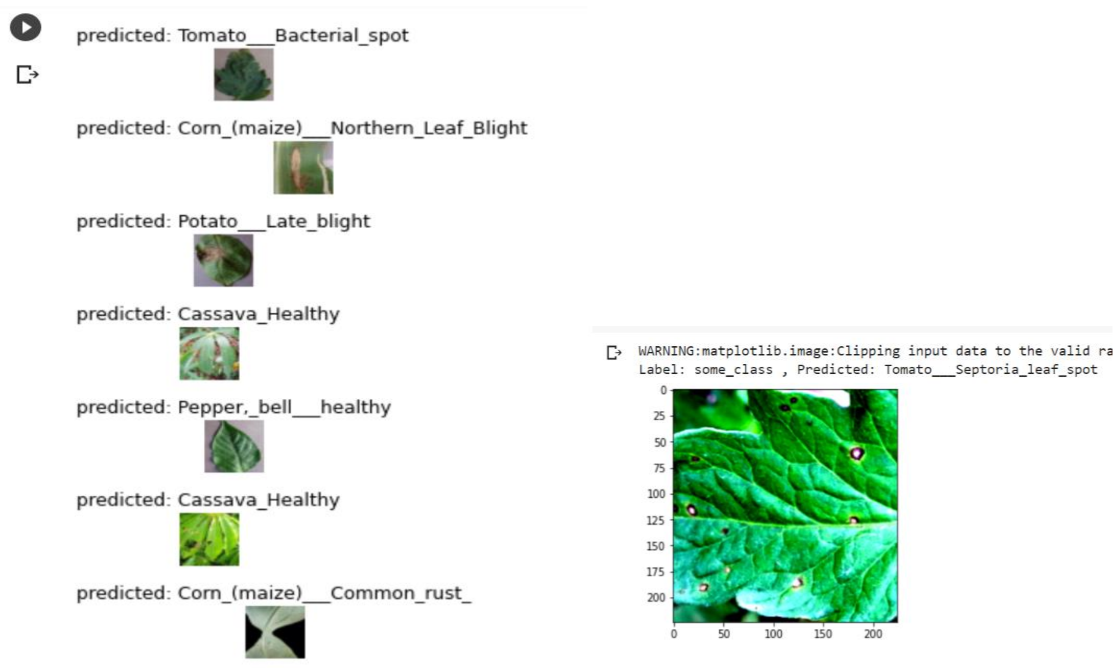
Figure 4:9 Plots of Training and Validation Losses and accuracies respectively

Table 4:4 Datasets and their respective accuracies

Dataset	**Accuracy**
Training	96%
Validation	96%
Testing	93%

The graphs indicate no overfitting over the hyperparameter tuning, but it is unusual to have higher validation accuracy than training. A possible cause could come from the way the validation is done. The validation logic is added to the training to set model evaluation to true if it detects that the data loader is coming from the validation folder. The learning rate scheduler enables the model to apply the learned parameters on the evaluation after each epoch; hence, the model optimizes using the scheduler after each epoch and adopts the weights in the evaluation phase.

Despite this strange observation, the testing result of the test data was accurate. The test dataset contained 2000+ images held outside training and was not known by the model.



```
predicted: Tomato___Bacterial_spot
predicted: Corn_(maize)___Northern_Leaf_Blight
predicted: Potato___Late_blight
predicted: Cassava_Healthy
predicted: Pepper,_bell___healthy
predicted: Cassava_Healthy
predicted: Corn_(maize)___Common_rust_
```

```
WARNING:matplotlib.image:Clipping input data to the valid ra
Label: some_class , Predicted: Tomato___Septoria_leaf_spot
```

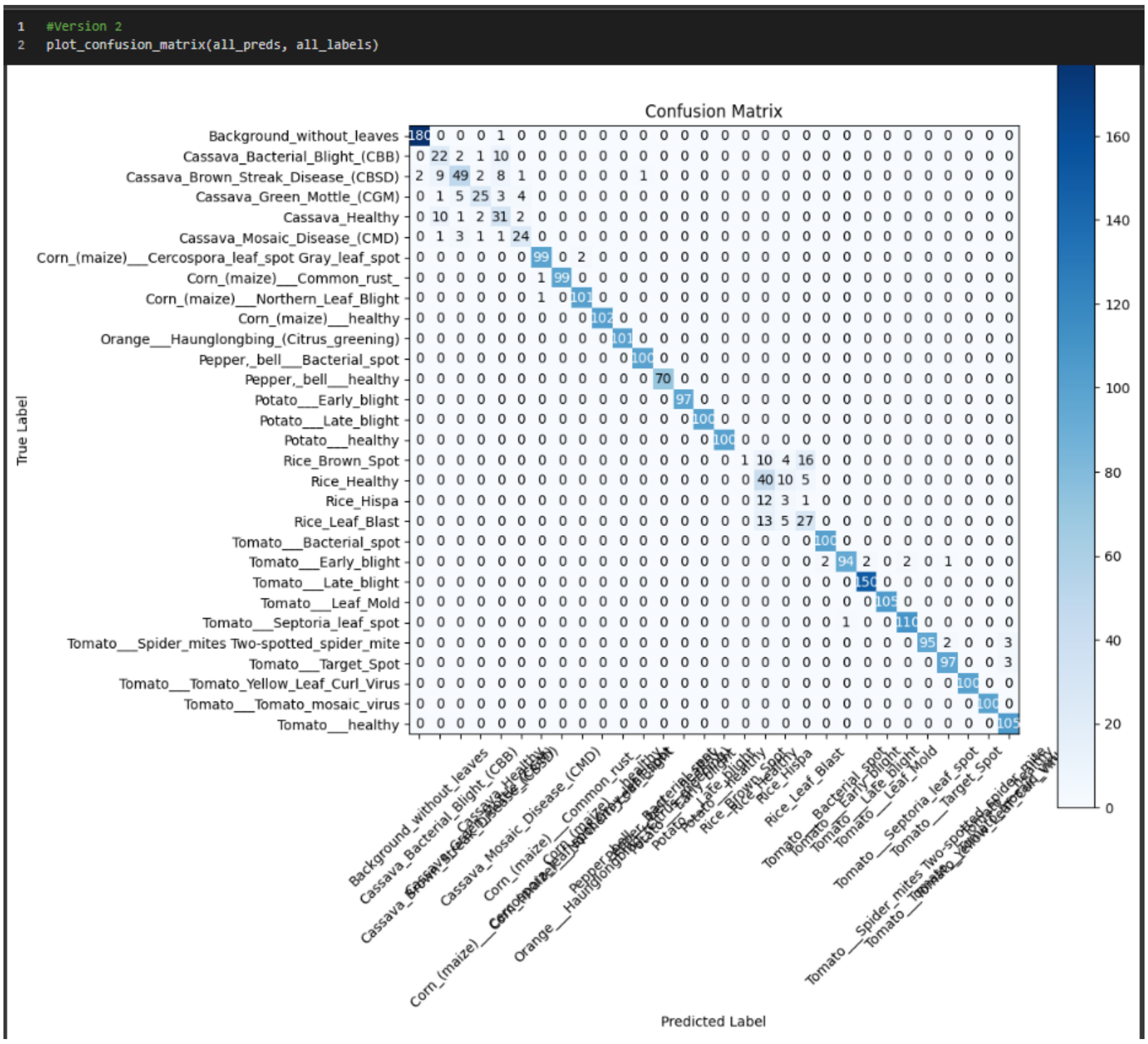
Figure 4:10: Sample Test Predictions on multiple images

Crop diseases can have a significant impact on agricultural yield and quality. Detecting crop diseases early and accurately is crucial for effective disease management strategies. Therefore, further evaluations such as Confusion Matrix, Precision, Recall, and F1 Score were performed to reveal that the deep learning model learned and generalized. This section illustrates and evaluates the model's performance based on the provided evaluation metrics: Precision, Recall, and F1-Score.

Precision, Recall, and F-Score have commonly used evaluation metrics in classification tasks, including crop disease detection. These metrics provide insights into the performance of a model in terms of its ability to predict positive and negative cases correctly. The next section analyzes the results obtained from evaluating the ResNet18 model.

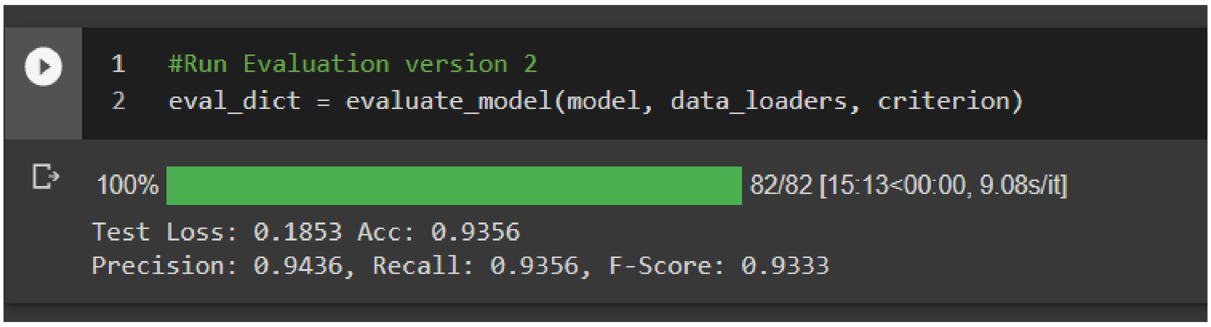
4.8.2 Confusion Matrix

A confusion matrix for the deep learning model was obtained from a test dataset size of 2594 samples. This dataset was carefully curated and prepared to represent a diverse and representative sample of the disease undercover. The test dataset was fed into the deep neural network model implemented in the system, which then made predictions for each sample based on the learned patterns from the training dataset. The predicted results were compared with the ground truth labels in the test dataset to compute the confusion matrix, which provided insights into the system's performance in terms of true positives, true negatives, false positives, and false negatives. The test dataset size of 2594 samples ensured that the system's performance was statistically robust and reliable, providing a comprehensive assessment of the accuracy and effectiveness of the system in identifying crop diseases in real-world conditions. The results obtained from the confusion matrix were used to validate the system's performance and make informed decisions for further improvements and future work. Figure 4.14 of the confusion matrix shows that the model predictions were nearly perfect, indicating that it has learned and can generalize.



```
#Version 2
plot_confusion_matrix(all_preds, all_labels)
```

Figure 4:11 A confusion matrix on the test dataset



```
#Run Evaluation version 2
eval_dict = evaluate_model(model, data_loaders, criterion)
```

```
100% 82/82 [15:13<00:00, 9.08s/it]
Test Loss: 0.1853 Acc: 0.9356
Precision: 0.9436, Recall: 0.9356, F-Score: 0.9333
```

Figure 4:12 Model evaluation on test data, including precision, recall, and F1 Score

Precision: 0.94

Precision is a measure of the accuracy of positive predictions made by a model. It is calculated as the ratio of true positive predictions to the total number of positive predictions (i.e., true positives and false positives). The model achieved a precision score of 0.94. This indicates that the model's positive predictions are accurate 94% of the time.

Recall: 0.93

Recall, also known as sensitivity or true positive rate, measures the ability of a model to correctly identify positive cases out of the total actual positive cases. It is calculated as the ratio of true positive predictions to the sum of true positive and false negative predictions. A recall score of 0.93 suggests that the model can correctly identify 93% of the positive cases.

F-Score: 0.93

F-Score, or F1-Score, is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance by considering precision and recall. F-Score is calculated as two times the product of precision and recall, divided by the sum of precision and recall. A higher F-Score indicates a better trade-off between precision and recall. In this case, the ResNet18 model has an F-Score of 0.93, meaning a relatively balanced performance regarding precision and recall.

Based on the evaluation results, the ResNet18 model has shown promising performance for crop disease detection with a precision of 0.94, a recall of 0.93, and an F-Score of 0.93. The model can accurately predict positive cases with high precision and identify many positive cases with high recall. The F-Score of 0.93 indicates a good balance between precision and recall, which is desirable in crop disease detection tasks.

The evaluation of the ResNet18 model for crop disease detection based on the provided metrics suggests that the model is performing well with high precision, recall, and F-Score. The experimentation and validation of the test datasets helped strengthen the reliability and applicability of the model for practical crop disease detection purposes.

Table 4:5 – Evaluation scores table

Evaluation Metrics	**Score**
Precision	94%
Recall	93%
F1 Score	93%

4.9 Model Deployment

After achieving above 96% validation, and 94% test accuracies and having justified that the deep learning model had learned and was able to generalize, the model was downloaded from Google Colab. Then an API was generated and deployed on Render using FastAPI. This process is explained further in the next sub-topic, called FastAPI.

4.9.1 FastAPI

FastAPI is a modern web framework used to build APIs quickly, efficiently, and with high performance. It is based on the Asynchronous Server Gateway Interface server, which is faster than the traditional Web Server Gateway Interface server used in Python web frameworks. FastAPI is built on top of Starlette for the web parts and Pydantic for the data validation and serialization parts [9].

In this project, FastAPI was used to deploy the model API on Render.com. Render provides a platform to deploy and manage the backend of web applications. FastAPI was chosen as the framework for the model API because it is lightweight, fast, and easy to use.

With FastAPI, an API for the deep learning model is built efficiently. FastAPI allowed the system to handle requests efficiently and quickly process responses, making it ideal for deploying the deep learning model on a web server. The ease of use and scalability of FastAPI made it the perfect choice for developing the model API on Render.com.

Chapter 5: Testing & Results

5.1 Development Testing

5.1.1 Unit Testing

Unit testing is an essential part of software development that involves testing individual units or components of a software system to ensure they function as intended. It is a fundamental practice that helps developers detect and fix errors early in the development cycle, leading to higher-quality software. Unit testing has been found to improve the quality of software systems, reduce the cost of testing, and reduce the number of defects in software [10]. To perform unit testing, code was written to test some key functions and the individual units of the Crop Lab software system. This was done to ensure that each unit performs as expected and that any changes made do not introduce errors.

The tool for unit testing in Ionic is called "Jasmine." It is a behavior-driven development (BDD) framework for testing JavaScript code, and it is specifically designed to work with Angular and Ionic. Jasmine provides a clean and intuitive syntax for writing tests, making it easy to define test suites, individual tests, and test expectations. It also includes built-in test runners that make running tests locally or in a continuous integration (CI) environment easy.

Jasmine can test all aspects of an Ionic application, including components, services, and directives. It allows developers to write tests that simulate user interaction with the Application, such as clicking buttons or entering text. This makes catching errors early in the development process possible before they make it into production. Jasmine is a powerful and flexible tool for unit testing Ionic applications, and the Ionic community widely uses it to ensure the quality and reliability of their code. Therefore, Unit testing for the crop lab was done with Jasmine.

5.1.2 Component Testing

Component testing is software testing that focuses on testing individual components or modules of a larger software system. The purpose of component testing is to ensure that each system component functions correctly and meets the required specifications. Components can be tested in isolation or in conjunction with other components to ensure that they integrate and work together as expected. During the development of Crop Lab's mobile Application, component testing was conducted by examining all pages that transfer data to other pages or components. For instance, on the home page, where users can view the application interface, clicking the "scan crop" button would open the camera page, which captures or uploads images and sends them to the diagnosis page. The image is formatted and sent to the model API for a disease response as JSON on the diagnosis page. The response is sent to the disease information and treatment page. These actions are illustrated in Figures 5.1 to 5.3 below.

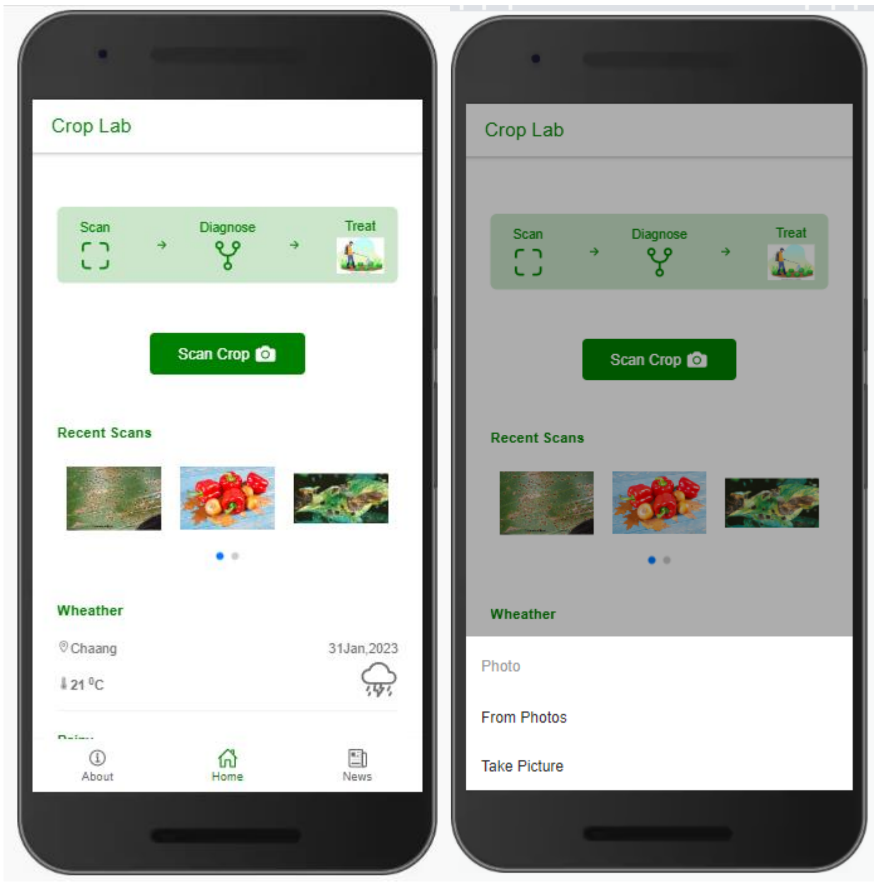
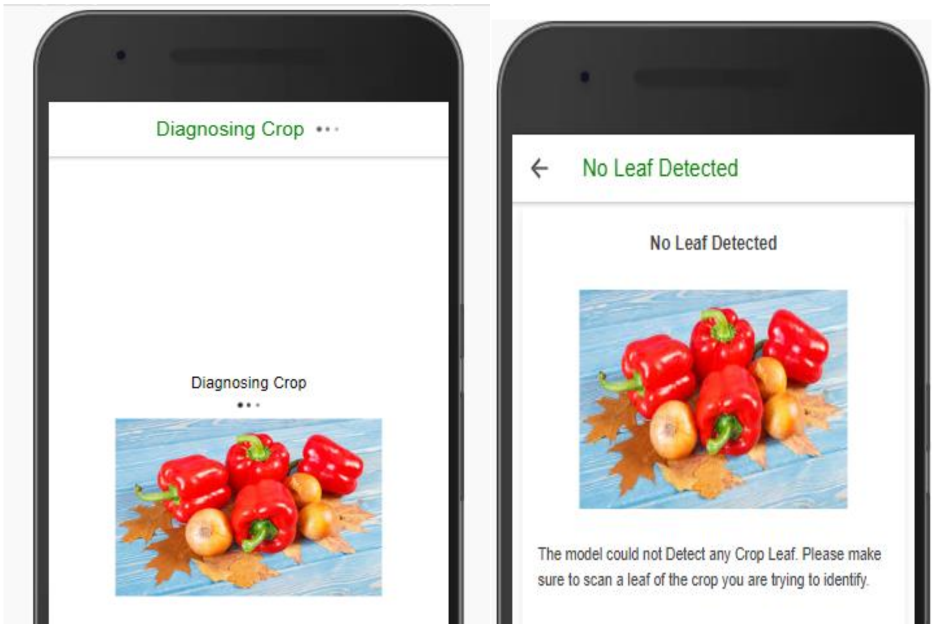


Figure 5:1: Click the “Scan Crop” Button action

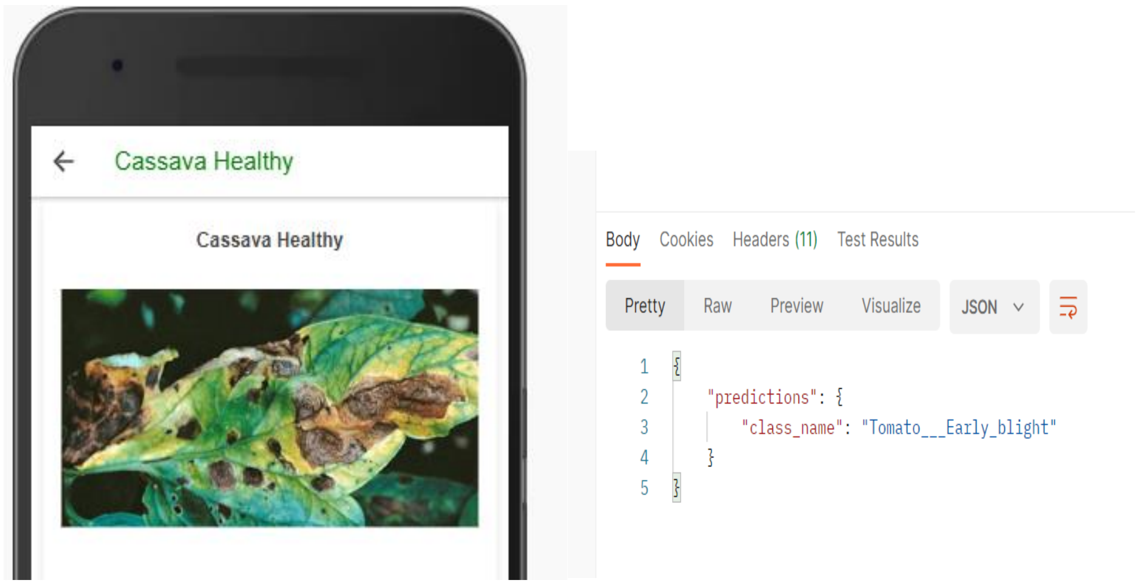


The model could not Detect any Crop Leaf. Please make sure to scan a leaf of the crop you are trying to identify.

Figure 5:2: Image data sent for Diagnosing

5.1.3 Integration & System Testing

During the system testing of Crop Lab, all the application components were integrated with the deep learning model API and tested as a whole. According to Sommerville, [5] system testing ensures that all the components interact correctly, transfer the right data across their interfaces, and are compatible. The complete Application was built and tested on Android and is yet to be tested on iOS devices to ensure that it works seamlessly on both platforms. Some bugs that were discovered during the testing process were addressed. For instance, predicting using the Postman API tester worked very well, but the same leaf prediction on the Android Application gave wrong results, as shown in Figure 5.5. This was caused by how images were formatted and passed through the HTTP request.



```
{
    "predictions": {
        "class_name": "Tomato___Early_blight"
    }
}
```

Figure 5:3: Wrong prediction on Android

5.1.4 User and Beta Testing

Release testing, according to Sommerville [6], involves testing a version of an application that has already been released by individuals who are not part of the development team. The primary objective of release testing is to guarantee that the Application is of acceptable quality and ready for use. On the other hand, user testing is a phase in which application users employ the Application and offer feedback or input on the Application [5].

Beta testing was performed on the mobile Application. Beta testing is a stage in which the Application is released to a limited number of users to assess the Application's performance and collect feedback. Beta testing is a critical aspect of the testing process since it allows developers to assess the Application's usability and identify any defects or issues that may have been overlooked in earlier testing stages. An APK of the Application was sent to people to install and interact with the system. Through the beta testing phase, user feedback was incorporated into the system. Some include allowing farmers to crop the image before uploading, changing the font size on the home page, and many other iterations.

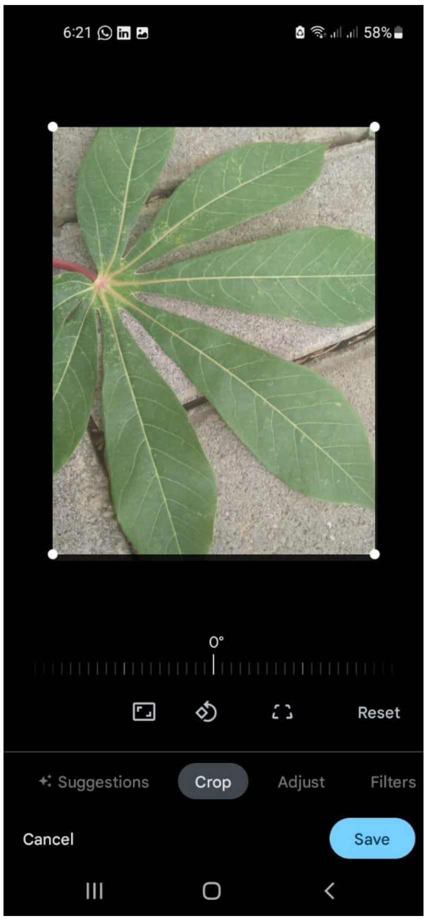


Figure 5:4 Incorporated cropping and editing image feature from test feedback

Chapter 6: Conclusion

Crop Lab, through a deep neural network model integrated with a mobile application, has shown promising results for farmers in Ghana. The system has demonstrated high accuracy (96% validation accuracy and 94% test accuracy) in identifying crop diseases using simple camera scans of plant leaves. This has the potential to significantly impact the agricultural sector in Ghana by providing farmers with timely and accurate information, eliminating the need for costly external agricultural officers. It has also helped to address the hurdle of manual inspection of crops by farmers.

6.1 Limitations

However, like any system, there are limitations to be addressed. One limitation is the reliance on clear and high-resolution images of plant leaves for accurate disease identification. Poor image quality may affect the system's performance. Additionally, the system's accuracy may vary depending on the specific crop species and disease types. Crop Lab only works best on seven crops, covering about twenty-nine different diseases. But there are more than seven crops and even many diseases which affect crop yield. Therefore, further research into obtaining data about these crops and diseases may be needed to overcome this limitation.

Another limitation of the system for rapid crop disease identification is the requirement of access to the Internet. The system relies on a mobile application that uses deep neural network models, which may require internet connectivity for real-time processing and analysis. In areas with limited or unreliable internet access, such as remote rural areas or regions with poor network coverage, farmers may face challenges in accessing the system and obtaining timely information about crop diseases. This limitation could hinder the system's effectiveness in reaching and assisting farmers who do not have reliable internet access. The image upload feature helps to overcome this limitation by allowing farmers to take pictures of crops even if there is no internet and upload them later for diagnosis.

6.2 Future Work.

Future work could expand the system to include more crop species and diseases. Further efforts could also be made to make the system more accessible to farmers with limited resources, such as by optimizing the mobile Application for low-cost devices or incorporating offline functionality. Image processing techniques and model training can be improved to overcome randomly wrong predictions. Despite achieving a promising accuracy of 94% on the test data, future work should strive to improve to attain even higher accuracies with more training epochs and more dataset.

In conclusion, Crop Lab will revolutionize crop disease management in Ghana, empowering farmers with a cost-effective and efficient tool to protect their crops and improve their economic prosperity. Further research and development efforts can build upon the system's successes and address its limitations, paving the way for a more sustainable and technology-driven agricultural sector in Ghana.

References

[1] S. Savary, L. Willocquet, S. J. Pethybridge, P. Esker, N. McRoberts, and A. Nelson, "The global burden of pathogens and pests on major food crops," *Nature Ecology & Evolution*, vol. 3, no. 3, pp. 430–439, Feb. 2019, doi: https://doi.org/10.1038/s41559-018-0793-y.

[2] A. Lakshmanarao, M. R. Babu, and T. S. R. Kiran, "Plant Disease Prediction and Classification using Deep Learning ConvNets," *2021 International Conference on Artificial Intelligence and Machine Vision (AIMV)*, Sep. 2021, doi: https://doi.org/10.1109/aimv53313.2021.9670918.

[3] S. V. Militante, B. D. Gerardo, and N. V. Dionisio, "Plant Leaf Detection and Disease Recognition using Deep Learning," *2019 IEEE Eurasia Conference on IOT, Communication, and Engineering (ECICE)*, Oct. 2019, doi: https://doi.org/10.1109/ecice47484.2019.8942686.

[4] D. K. Singh and R. Sobti, "Role of Internet of Things and Machine Learning in Precision Agriculture: A Short Review," *2021 6th International Conference on Signal Processing, Computing and Control (ISPCC)*, Oct. 2021, doi: https://doi.org/10.1109/ispcc53510.2021.9609427.

[5] I. Sommerville, *Software engineering*, 10th ed. Boston, Mass. Amsterdam Cape Town Pearson Education Limited, 2016.

[6] T. Iulia-Maria and H. Ciocarlie, "Best practices in iPhone programming: Model-view-controller architecture — Carousel component development," *2011 IEEE EUROCON - International Conference on Computer as a Tool*, Apr. 2011, doi: https://doi.org/10.1109/eurocon.2011.5929308.

[7] Ionic, "Ionic - Cross-Platform Mobile App Development," *Ionic Framework*, 2019. https://ionicframework.com/

[8] "Data augmentation," *Engati*. https://www.engati.com/glossary/data-augmentation#:~:text=%E2%80%8D- (accessed Apr. 03, 2023).

[9] "FastAPI," *fastapi.tiangolo.com*. https://fastapi.tiangolo.com/

[10] "What Is Unit Testing?," *smartbear.com*. https://smartbear.com/learn/automated-testing/what-is-unit-testing/#:~:text=A%20unit%20test%20is%20a

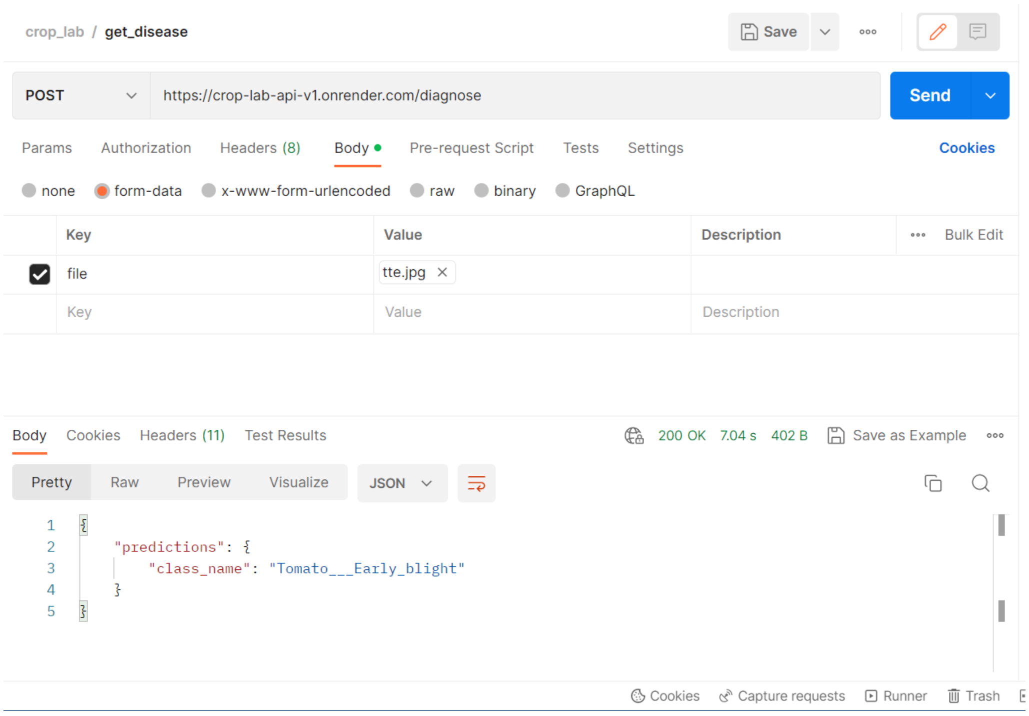
[11] "Home," Crops Research Institute. http://cri.csir.org.gh/ (accessed Apr. 20, 2023).

Appendix

6.3 Questionnaires: Farmers

1. What do you do for a living?
2. What is your highest level of education?
3. Which type of crops do you cultivate?
4. What is the major contributing factor to your crop yield?
5. How do you identify/detect infectious crops on your farm?
6. Are you satisfied with that approach?
7. How do you handle and treat infected crops?
8. Do you use or have access to a smartphone?

6.4 Testing Model API on Postman



crop_lab / get_disease
Save
POST
Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL
Body Cookies Headers (11) Test Results
Save as Example
Pretty Raw Preview Visualize JSON
Cookies Capture requests Runner Trash

POST https://crop-lab-api-v1.onrender.com/diagnose

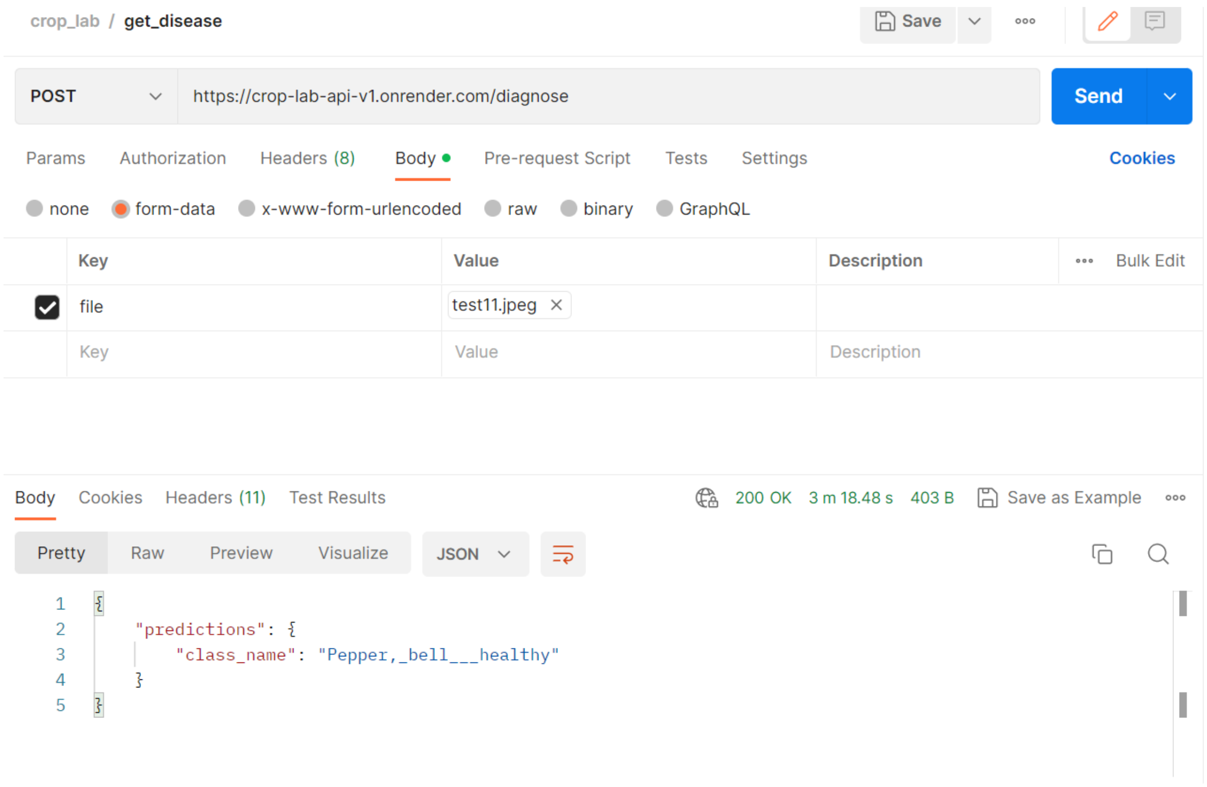
Body: form-data

Key	Value	Description
file	tte.jpg	

200 OK 7.04 s 402 B

```
{
    "predictions": {
        "class_name": "Tomato___Early_blight"
    }
}
```

Figure 0:1 - Testing Model API with Postman



crop_lab / get_disease
Save
POST
Send
Params Authorization Headers (8) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL
Body Cookies Headers (11) Test Results
Save as Example
Pretty Raw Preview Visualize JSON

POST https://crop-lab-api-v1.onrender.com/diagnose

Body: form-data

Key	Value	Description
file	test11.jpeg	

200 OK 3 m 18.48 s 403 B

```
{
    "predictions": {
        "class_name": "Pepper,_bell___healthy"
    }
}
```

Figure 0:2 - Testing Model API on Postman

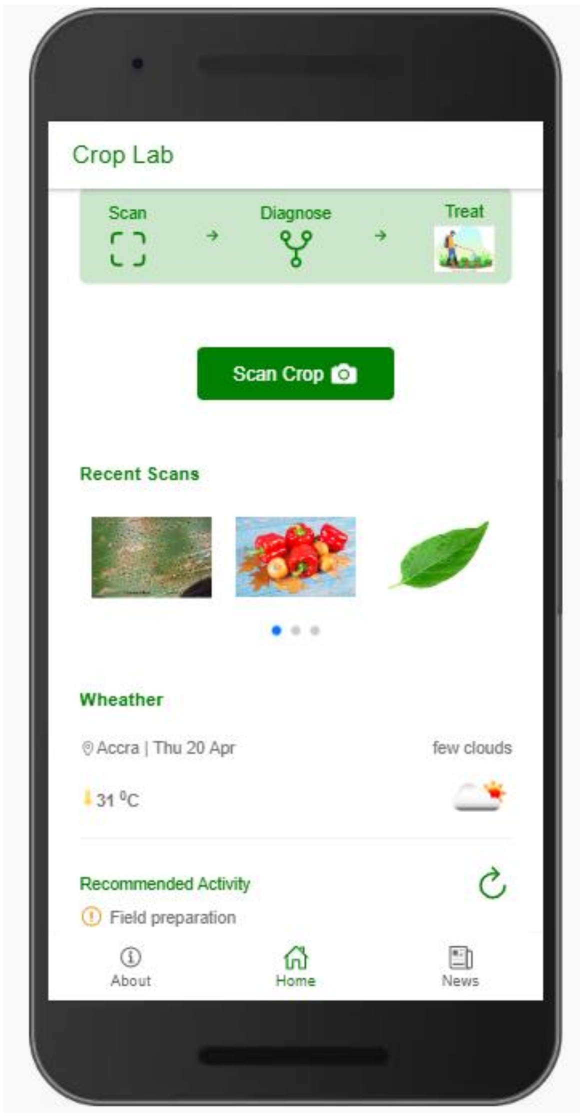
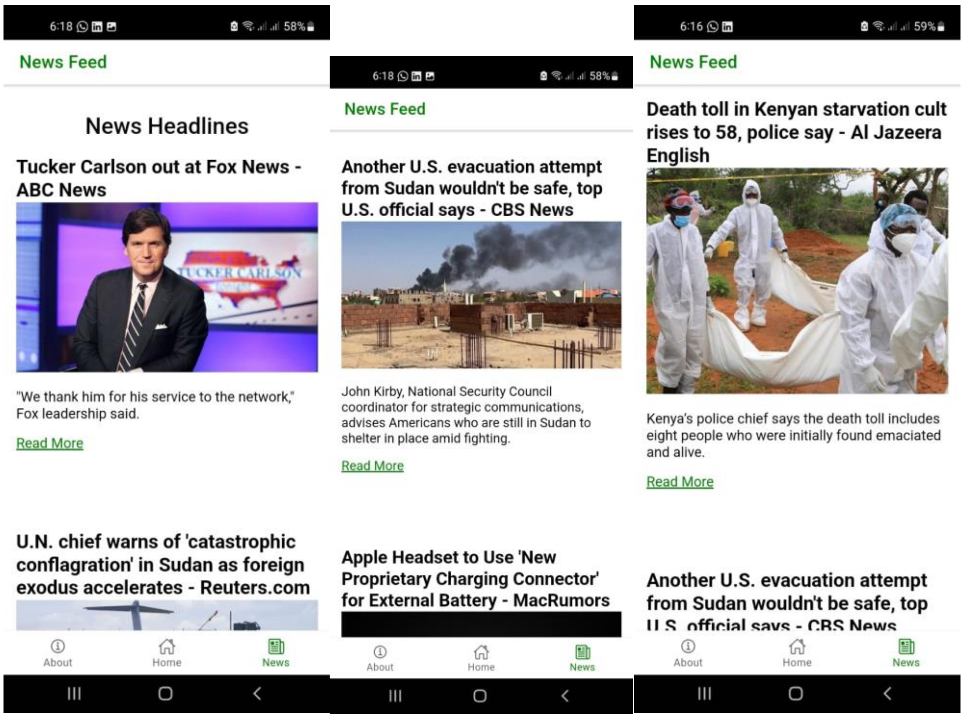


Figure 0:2 – Crop Lab home page with weather updates and recommended activity

6.5 Crop Lab News Feed Pages


6:18 58%

News Feed

News Headlines

Tucker Carlson out at Fox News - ABC News

"We thank him for his service to the network," Fox leadership said.

Read More

U.N. chief warns of 'catastrophic conflagration' in Sudan as foreign exodus accelerates - Reuters.com

About Home News

6:18 58%

News Feed

Another U.S. evacuation attempt from Sudan wouldn't be safe, top U.S. official says - CBS News

John Kirby, National Security Council coordinator for strategic communications, advises Americans who are still in Sudan to shelter in place amid fighting.

Read More

Apple Headset to Use 'New Proprietary Charging Connector' for External Battery - MacRumors

About Home News

6:16 59%

News Feed

Death toll in Kenyan starvation cult rises to 58, police say - Al Jazeera English

Kenya's police chief says the death toll includes eight people who were initially found emaciated and alive.

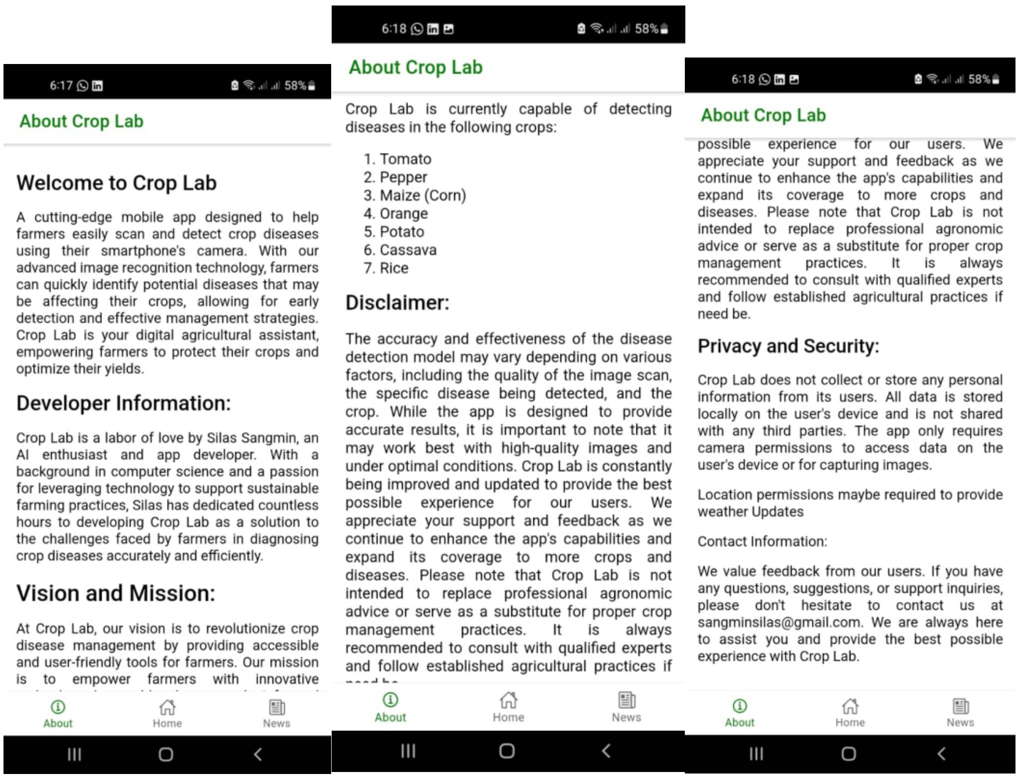
Read More

Another U.S. evacuation attempt from Sudan wouldn't be safe, top U.S. official says - CBS News

About Home News

Figure 0:3 News feed pages

6.6 About Page



6:17

About Crop Lab

Welcome to Crop Lab

A cutting-edge mobile app designed to help farmers easily scan and detect crop diseases using their smartphone's camera. With our advanced image recognition technology, farmers can quickly identify potential diseases that may be affecting their crops, allowing for early detection and effective management strategies. Crop Lab is your digital agricultural assistant, empowering farmers to protect their crops and optimize their yields.

Developer Information:

Crop Lab is a labor of love by Silas Sangmin, an AI enthusiast and app developer. With a background in computer science and a passion for leveraging technology to support sustainable farming practices, Silas has dedicated countless hours to developing Crop Lab as a solution to the challenges faced by farmers in diagnosing crop diseases accurately and efficiently.

Vision and Mission:

At Crop Lab, our vision is to revolutionize crop disease management by providing accessible and user-friendly tools for farmers. Our mission is to empower farmers with innovative

About Home News

6:18

About Crop Lab

Crop Lab is currently capable of detecting diseases in the following crops:

1. Tomato
2. Pepper
3. Maize (Corn)
4. Orange
5. Potato
6. Cassava
7. Rice

Disclaimer:

The accuracy and effectiveness of the disease detection model may vary depending on various factors, including the quality of the image scan, the specific disease being detected, and the crop. While the app is designed to provide accurate results, it is important to note that it may work best with high-quality images and under optimal conditions. Crop Lab is constantly being improved and updated to provide the best possible experience for our users. We appreciate your support and feedback as we continue to enhance the app's capabilities and expand its coverage to more crops and diseases. Please note that Crop Lab is not intended to replace professional agronomic advice or serve as a substitute for proper crop management practices. It is always recommended to consult with qualified experts and follow established agricultural practices if

About Home News

6:18

About Crop Lab

possible experience for our users. We appreciate your support and feedback as we continue to enhance the app's capabilities and expand its coverage to more crops and diseases. Please note that Crop Lab is not intended to replace professional agronomic advice or serve as a substitute for proper crop management practices. It is always recommended to consult with qualified experts and follow established agricultural practices if need be.

Privacy and Security:

Crop Lab does not collect or store any personal information from its users. All data is stored locally on the user's device and is not shared with any third parties. The app only requires camera permissions to access data on the user's device or for capturing images.

Location permissions maybe required to provide weather Updates

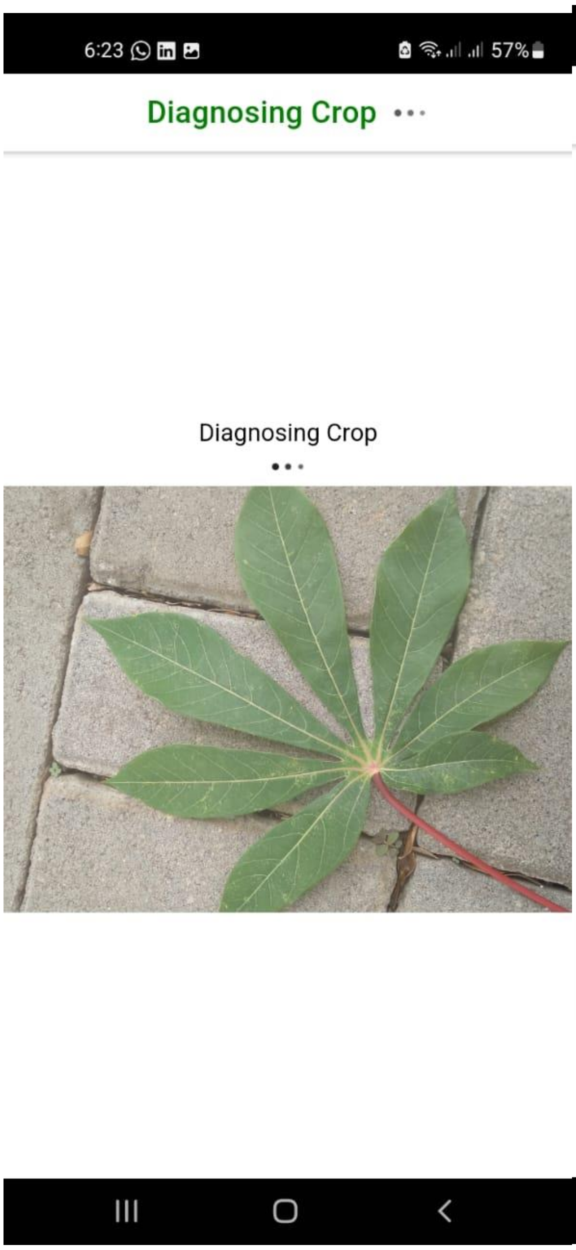
Contact Information:

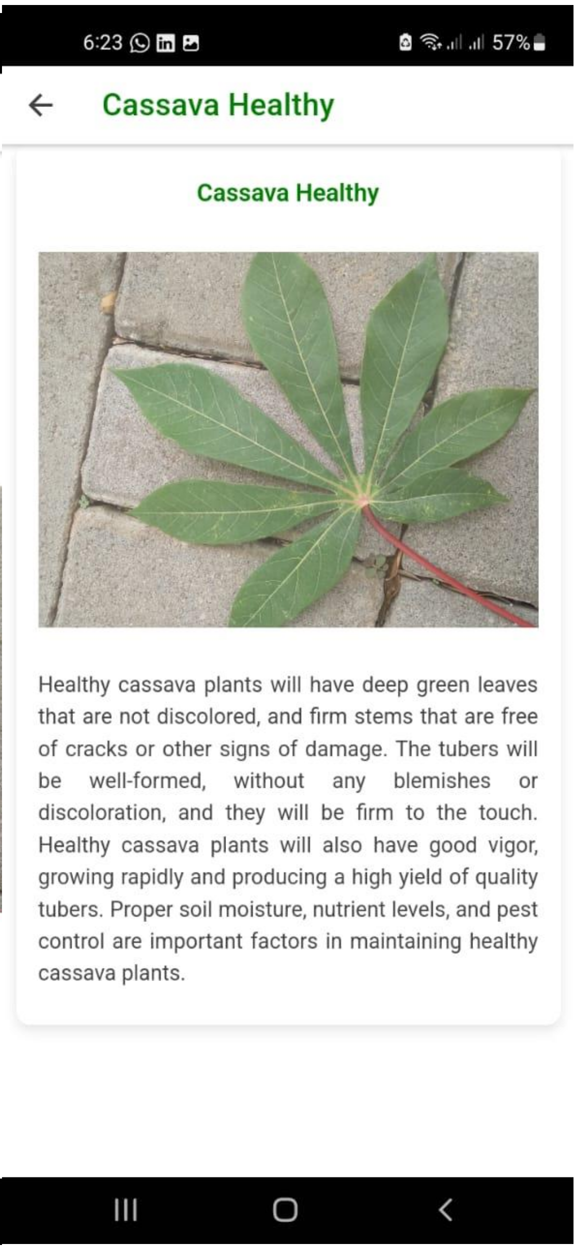
We value feedback from our users. If you have any questions, suggestions, or support inquiries, please don't hesitate to contact us at sangminsilas@gmail.com. We are always here to assist you and provide the best possible experience with Crop Lab.

About Home News

Figure 0:4 About page

6.7 Crop Lab – Other pages





Cassava Healthy

Healthy cassava plants will have deep green leaves that are not discolored, and firm stems that are free of cracks or other signs of damage. The tubers will be well-formed, without any blemishes or discoloration, and they will be firm to the touch. Healthy cassava plants will also have good vigor, growing rapidly and producing a high yield of quality tubers. Proper soil moisture, nutrient levels, and pest control are important factors in maintaining healthy cassava plants.

Figure 0:5 Sample diagnosed results