



UNIVERSITY OF NAIROBI
SCHOOL OF COMPUTING AND INFORMATICS
FOURTH YEAR FINAL REPORT

NAME: BRIAN KIPTOO

REG NO: P15/85932/2017

PROJECT TITLE: TOMATO DISEASE DETECTION AND DIAGNOSTIC APP USING
MACHINE LEARNING
(TOMATO LIVING)

SUPERVISOR: DR. MIRITI

Submitted in partial fulfillment of the requirements for award of BSc. Computer Science

DECLARATION

I hereby affirm that this project report, as presented in this report, is entirely my own work, and has to the best of my knowledge, not been submitted to any other institution of higher learning.

Student: Brian Kiptoo

Reg. Number: P15/85932/2017

Signature:

Date: ...20/05/2022.....

This project report has been submitted as a partial fulfillment of requirements for the Bachelor of Science in Computer Science of the University of Nairobi with my approval as the University Supervisor.

Supervisor: Dr. Evans Miriti

Signature:

Date:

ACKNOWLEDGEMENT

I thank the Almighty God for the grace and strength throughout this project period.

I am also very grateful to my supervisor, Dr. Miriti for guiding and supporting me. Without his intelligence and remarkable generosity in sharing ideas and telling me what works best and what does not, this project wouldn't be complete.

Not forgetting my dear friends and classmates, who also undergoing similar pressure, took time to assist me in undertaking this project – not once, but severally – whenever necessary.

I would also like to express my sincere gratitude to Mr. Simon Nyaga, a research expert from Kenya Agricultural Research Institute with whom I interacted with throughout the development of this project.

Many thanks to the School of Computing and Informatics staff, as well as my colleagues, for the support given in realizing the objectives of this project.

Finally, my deep gratitude to my family who supported me, gave words of encouragement whenever I was in need of them, honored my needs for the project, and never wavered in that strange and beautiful faith born of love.

Thank you all.

ABSTRACT

This is a project document that details an attempt to identify and diagnose disease-infected tomatoes using machine learning and computer vision techniques. The use of computer systems in automation, mechanization, digitization, and the development of expert systems employed in smart agriculture has increased over time. In agriculture, there has also been a considerable surge in the usage of new technologies such as AI and IOT. However, considerable work remains to be done because professional knowledge in agriculture, particularly tomato cultivation, has yet to be properly captured. Experienced diagnosis remains a difficulty for farmers, who must rely primarily on field experts and agricultural cooperatives, both of which are in short supply in the field. This results in a low yields and farmers abandoning tomato farming.

Various deep machine learning approaches, such as K-Nearest Neighbor (KNN), Google's TensorFlow Framework for image recognition, and OpenCV, can be used to diagnose diseases. Farmers may get professional advice with just a few touches of a few buttons thanks to the machine learning algorithms used. They can diagnose tomato problems with a 90+ percent accuracy, allowing them to improve the quality and quantity of their yields.

Table of Contents

Contents

DECLARATION	2
ABSTRACT.....	3
Table of Figures	6
1.1 BACKGROUND.....	7
1.2 PROBLEM STATEMENT	8
1.3 PROJECT GOALS AND OBJECTIVES	9
1.3.1 Research Objectives.....	9
1.3.2 System Objectives.....	9
1.4 Justification	9
1.4.1 Stakeholders.....	9
1.4.2 Technologies	10
1.5 PROJECT JUSTIFICATION.....	10
1.6 EXPECTED OUTCOME.....	10
1.7 ASSUMPTIONS	10
CHAPTER 2: LITERATURE REVIEW	11
2.1 INTRODUCTION.....	11
2.1 Review on existing systems in farming.....	12
2.4 The Gap	14
2.3 Proposed solution	15
2.5 Key Concepts	15
2.5.1 Deep Learning Concept	15
.....	16
1.5.2 Deep Learning Algorithm(s).....	18
CHAPTER 3: SYSTEM ANALYSIS AND DESIGN	23
3.1 METHODOLOGY	23
3.1.1 APPROACH.....	24
3.2 SYSTEM ANALYSIS	24
3.2.1 Data Collection	25

3.2.2 Feasibility Analysis	30
3.1.4 System Analysis Models	32
3.1.5 Function Specifications	37
3.2 SYSTEM DESIGN	37
3.2.1 System Architecture	39
3.2.2 Data Flow Diagrams	40
3.2.3 Database Design.....	41
3.3 IMPLEMENTATION AND TESTING.....	42
3.3.1 Resources.....	42
References	59
APPENDIX 1	60
Project Structure	60
Gantt Chart	60
APPENDIX 2.....	62
Sample Code	62

Table of Figures

Figure 2 plant disease mobile app	12
Figure 3 crop disease ml.....	13
Figure 4 model building.....	17
Figure 5 VGG19 architetcure.....	20
Figure 6 dataset preview.....	21
Figure 7 agile vs traditional.....	23
Figure 8 farmers pie-chart.....	26
Figure 9 internet availability pie-chart.....	27
Figure 10 smart phone usage pie-chart.....	28
Figure 11 crop loss.....	29
Figure 12 disease identification methods	30
Figure 13 farmer use case.....	32
Figure 14 admin usecase.....	33
Figure 15 farmer activity diagram.....	34
Figure 16 admin activity diagram.....	35
Figure 17 system architetcure.....	39
Figure 18 component level diagram.....	40
Figure 19 data flow diagram.....	41
Figure 20 database design.....	42
Figure 21 accuracy vs val accuracy.....	45
Figure 22 loss vs val_loss.....	45
Figure 23 model accuracy.....	46
Figure 24 confusion matrix.....	46
Figure 25 model test.....	47
Figure 26 sign up.....	48
Figure 27 verify sign up.....	49
Figure 28 Home screen.....	50
Figure 29 disease diagnosis results.....	51
Figure 30 disease treatment.....	52
Figure 31 share results to social media.....	53
Figure 32 project structure.....	60
Figure 33 Gantt chart.....	61
Figure 34 sample code 1.....	62
Figure 35sample code 2.....	63
Figure 36 main.dart.....	64
Figure 37 homepage.....	65
Figure 38 home.dart.....	66
Figure 39 call flask api.....	67
Figure 40 labels.txt.....	68

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND

Kenya continues to rely on agriculture for food and economic development (Ochilo et al., 2019). The sector is a key economic pillar contributing 24% of the gross domestic product and about 65% of exports (Nyamwamu, 2016). Horticulture forms the bulk of agriculture with vegetables accounting for 80% of growers and 60% of exports (Yabs and Awuor, 2016). Tomato is one of the most widely grown vegetable food crops not only in Kenya but also in East Africa and the whole world at large, second only to potato (FAO 2005; Maerere et al. 2006). The crop is one of the most highly consumed vegetable crops in Kenya (National farmers information service). Very large quantities of tomatoes are processed and preserved in a variety of forms. Much of the volume of processed tomatoes is packaged as tomato paste and used to make products such as juice, sauces and soups (FAO 2000). In Kenya, tomato is an important economic activity for small holder farmers. However, tomato farming has been greatly affected by the prevalence of pests and diseases. This is brought about by the lack of enough resources to diagnose and accurately identify the diseases, since most farmers rely on the traditional eye disease identification techniques.

Farmers have a rich and in depth knowledge of tomato production through experience, agricultural extension officers in the locality and other channels such as mass media and agro-vet shops, however there are still major gaps in knowledge, especially concerning the control of wilt diseases nematodes and blight. Therefore there is need to adopt artificial intelligence and smart agriculture to effectively manage the nematodes and wilt diseases. There is need to adopt methods of pest and disease control that is environmentally friendly and that do not affect human health.

The major issue affecting tomato farmers is intertwined with issues of sustainable agriculture and climate change. According to the findings, climatic change can affect pathogen development stages and rates, as well as host resistance, resulting in physiological alterations in host-pathogen interactions. The situation is made even more complicated by the fact that diseases are now more easily transmitted globally than ever before. New diseases can emerge in regions where they have never been seen before, and where there is, by definition, no local competence to combat them.

Inexperienced pesticide application in disease control can result in pathogens developing long-term resistance, severely limiting their ability to fight back. One of the cornerstones of precision agriculture is timely and precise identification of plant diseases. By tackling the long-term pathogen resistance development problem and minimizing the negative consequences of climate change, it is critical to avoid wasteful waste of financial and other resources, resulting in healthier production.

In today's ever-changing climate, accurate and fast disease detection, as well as early disease prevention, has never been more critical for tomato growers. The naked eye examination of a qualified professional is the primary technique used in practice for plant disease diagnosis because most diseases that impact tomato production reveal themselves in the visible spectrum.

Innovations in machine learning and deep learning techniques have the potential to broaden and improve the practice of precision plant protection while also expanding the market for image processing techniques in precision agriculture.

1.2 PROBLEM STATEMENT

Agriculture is the backbone of all civilizations. Agricultural productivity has already had a substantial impact on the economic development of industrialized countries, and its position in countries which are still developing is crucial. According to the FAO (Food and Agricultural Organization), more than half of the world's population (62%) dependent upon agriculture for preservation, making agriculture integral to the global economy. It is found that globally, grain losses of 10% - 20%, rice losses of 25% - 41%, maize losses of 20%, tomato losses of 8% etc. due to pests and mainly due to crop diseases. And it is clear that plant diseases makes a tremendous impact on global economy. Agriculture plays a critical role in providing food supply for growing population in the world. Annual global food supply loss due to plant diseases is 40% on average each year. In Kenya, small scale farmers generate more than 80% of the country's agricultural production. To them, loss of crops due to pests and diseases has devastating consequences.

Detection of plant diseases therefore plays a key role in agriculture. Plant diseases are unavoidable if adequate precautions are not taken in this field, cause enormous effects on plants and affect quality, abundance or productivity. Crop loss monitoring and treatment involves a multidisciplinary approach since pathogens (fungi, bacteria, viruses, and nematodes) not only communicate with each other, but also with other biotic and abiotic factors that affect yield. Developing countries such as Kenya with limited access to disease control regulations and average losses of between 30% -50% are common for main crops of the country. Naked eye observation by specialists is the primary method used in plant disease recognition. Nonetheless, this needs continuous supervision of experts that can be incredibly expensive and hard to find. A great accomplishment and a real move forward would be to be able to reliably determine the effect of pests and plant diseases on crop production by resolving the above issue in a proper manner.

There are not enough resources to provide farmers with expert advice on proper diagnosis and identification of diseases which results in untimely application of pesticides due to assumptions and misdiagnosis. This often results in reduced quality and quantity of the tomatoes produced in the country and a high cost of production.

1.3 PROJECT GOALS AND OBJECTIVES

The main objective of this project is to deliver an effective and consistent autonomous crop disease detection application which identifies and recognizes various types of crop diseases. The application should diagnose tomato crop disease by examining the state of the plant leaf contaminated by the disease. This would constitute the first phase of the app by diagnosing tomato diseases. The application would then be scaled to include more crops such as maize etc. in order to obtain optimal precision, the system will be tested in Kenya agricultural environments such as tomato farms and home gardens.

1.3.1 Research Objectives

- i. To understand problems that affect tomato farming
- ii. To understand the reason why farmers are abandoning tomato farming in many areas in Kenya.
- iii. To understand the production process of tomato, the benefits and downfalls.
- iv. To know how farmers, make diagnosis of tomato diseases
- v. To find out if agricultural officers still exists and how they offer help, and if they are privately or publicly employed
- vi. To understand how often and how many farmers benefit from the field officers

1.3.2 System Objectives

- i. The proposed solution should give farmers an accurate disease identification and diagnosis.
- ii. Give farmers information about other pests and diseases they might be facing
- iii. Keep the data gathered from the app.
- iv. Give the farmers advice on the best farming practices for tomato
- v. Help suppliers to know what pesticides and chemicals to stock from the data obtained from the farmers

1.4 Justification

- i. Using this system will give farmers expert information on tomato diseases when they need it wherever they are without the presence of an expert field officer.
- ii. There will not be inexperience application of pesticides and farm chemicals
- iii. The system shall provide accurate and waranted disease diagnosis.
- iv. It will also help in collection of data which can help in improvement of farming and thus production in future

1.4.1 Stakeholders

The key stakeholders of the proposed solution include farmers (users) and an application administrator

1.4.2 Technologies

The key technologies to be used in the system include the web technologies and mobile technologies for the frontend implementation, artificial intelligence and deep learning for image classification and disease classification and Mysql Database and flask for the backend and api.

1.5 PROJECT JUSTIFICATION

The use of the proposed system will readily provide farmers with expert information on tomato diseases when they need it without the need for the presence of a field expert. This will ensure that there will be minimum inexperienced application of pesticides and farm chemicals. The data collected from disease analysis will help improve farming practices and aid in predicting future trends

1.6 EXPECTED OUTCOME

An intelligent system that diagnoses diseases automatically with an accuracy of above 50%. The system should also communicate with the various key players like agricultural officers and agrovet outlets as well as providing analysis of the diseases diagnosed.

1.7 ASSUMPTIONS

The assumption of this system is that farmers will have access to internet enabled phones, camera enabled phones and can afford internet once in a while. It is also assumed that most farmers can read or at least have someone to read for them.

CHAPTER 2: LITERATURE REVIEW

2.1 INTRODUCTION

In the country, there is a huge gap between technology and agriculture. In Kenya, there are few expert systems that provides farmers with expert advice or at least, not any that I know. Pest and disease outbreaks have become a major concern globally and extension service providers and policy makers are highly challenged in their management (MOAL&F, 2012). Over the past few years, farmers have slowly adapted to technology. Such technologies include use of IOT for smart farming, GPS measuring applications and the introduction of Mpesa for mobile money transfers and the introduction of digitized systems for soil testing.

In the past, people used to judge the class of tomato disease subjectively through experience, but the ability to distinguish amongst multiple diseases is limited and the process is time consuming.

Nowadays, there has been an accelerated increase in the use of computer vision and machine learning to identify and detect diseases. Machine learning image processing technology is developing rapidly and is widely used in all aspects, including the agricultural field. Applying machine learning and image processing technology to crop disease recognition has incomparable advantages over traditional manual diagnosis and recognition methods. People only need to collect a small number of disease image samples. The process involves the following steps: firstly, the dataset is pre-processed; secondly, the feature extraction algorithm is used to extract the features of the disease area in the image; lastly, the obtained feature information is sent to the classifier for training and the model parameters are obtained. The generated model can be used to detect the disease category. Training with a large number of datasets is time consuming due to the lack of datasets and the poor generalisation ability of the model. Moreover, the development of agricultural modernization towards the direction of intelligence has highlighted the shortcomings of these traditional image detection methods.

2.1 Review on existing systems in farming

Plant Disease Identification Mobile Application

As (Naveen Chandra Gowda, 2019) has established, the system consists of an android mobile application that provides a feature in which farmers can scan the suspicious plant for disease from the application camera and identify the plant disease.

The front-end module consists of the 'detect a disease' option and the method to detect various plant types. The user can enter all the details of the infected plant and can get remedies accordingly. Following figure an example of a plant image scanning process of the application.

Input: Image of the diseased plant
result

Output: Diagnosis

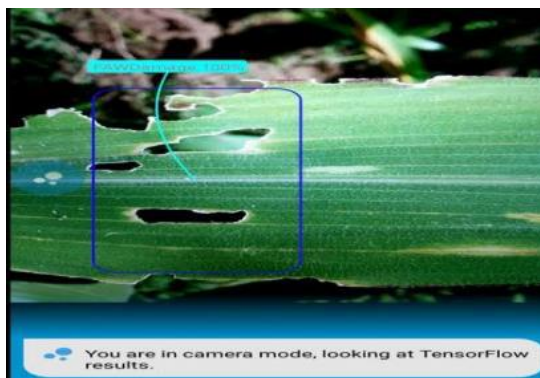


Figure 1 plant disease mobile app

(Naveen Chandra Gowda, 2019)

A Tensorflow model specially developed on the dataset of infected plants to perform plant recognition process. The approach used to identify crop disease is the use of the Convolutional Neural Network (CNN). Data set has been separated in to 10:1 ratio as training data and validation data. The *clustering algorithm* of machine learning is used which are prewritten in the Tensorflow framework. The model is optimized by using the optimization techniques in the Tensorflow such as *model optimization, pruning, stemming*. It is found that the deep learning model in the system has performed a 97.61% accuracy.

Crop disease recognition System Using Machine Learning

Contrary to (al., 2018) Pathogens and crop diseases can lead to reduce in food production leading to food malnutrition. In order to address above issue a desktop application which is capable of detecting plant diseases was developed using machine learning.

The application provides the feature to the farmer to upload a picture of the diseased crop leaf and the system returns the condition of the plant. Some steps have been taken to determine if the

leaf is diseased or healthy. Ex: pre-processing, Extraction of Features, Classifier Learning and Identification.

The system has been used *random forests* classifier. Dataset is divided in to two parts such as training and validation data. Using HOG feature extraction technique feature vector is produced and trained using Random Forest algorithm. Feature vector generating using HOG feature descriptor for testing data and testing data can be represented as the given image below.

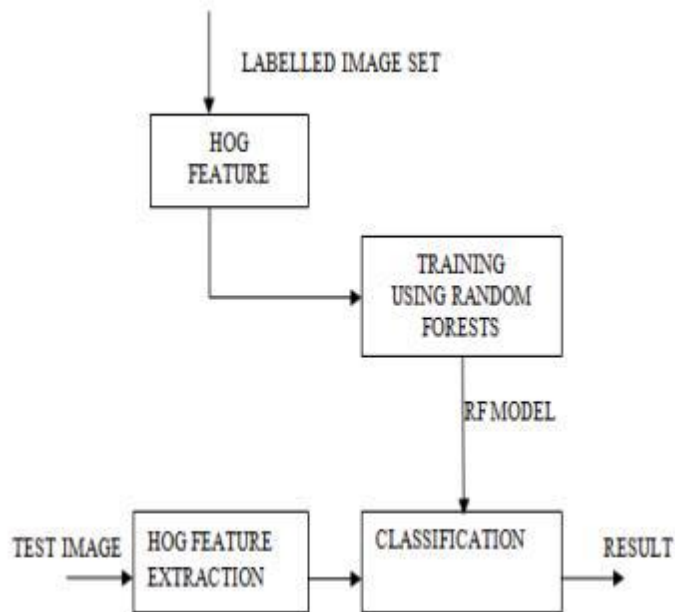


Figure 2 crop disease ml

Application could classify diseased plants with approximate 70% percent accuracy.

PlantVillage

This is a user moderated Q & A forum helps farmers solve their plant related questions. It was formed by researchers from Penn State.

MyFugo:

This app helps farmers detect when animals are on heat even when they are away from home and keep digital records.

Weed Manager:

This is an app developed by the ICAR-Directorate of Weed Research, Jabalpur. It is a user-friendly mobile App for farmers, Agriculture department officials, students, other stakeholders and industry professionals.

Nuru:

This is a phone app developed by PlantVillage - a department of researchers at Penn State University- uses artificial intelligence and machine learning to detect common diseases in cassava and maize.

Mfarm:

Is an app that connects buyers and farmers, to help farmers sell their products. It provides farmers with market for their products

Plantix

Is a global android application that assists farmers in diagnosis of plant diseases using their phones. The application contains only a few crops and tomatoes are not included.

Farm Drive

Helps small scale farmers to access finance. It mostly deals with the financial needs for small scale farmers.

Advantages and Disadvantages of the existing systems

- i. Diseases and soil nutrients are different from one environment to another thus using the global applications may not be applicable to farmers in Kenya
- ii. The existing systems have not yet achieved a better accuracy in diagnosis of the diseases.
- iii. The field officers, considering there are very few are not always available when the farmers need them
- iv. The plant clinics are only available in very few counties which makes it difficult to farmers to access them
- v. Most systems in Kenya are focusing on providing financial needs other than the diseases the farmers experience from their farms
- vi. User should supply all the details of the plant manually in order to start the scanning process.

2.4 The Gap

Based on the above existing systems, it is quite clear that there is no current system that does a diagnosis for plant diseases, and one that does is not freely available to Kenyan farmers and the content is not localized.

2.3 Proposed solution

The proposed tomato disease identification and diagnostic application, seeks to help farmers to make the right diagnosis of the diseases faced by their crops and be given advice on management.

The system would also help farmers in farm management, where data from the farmers would be stored. The process would involve application of deep learning and machine learning algorithms on images of disease infected tomato leaves, to generate image classification models that would be applied in the identification and diagnosis of disease infected plants. From the large data generated, stakeholders in the industry such as agro-chemical stockers, agrovets and cooperatives would know which are the most prevalent diseases and stock accordingly.

2.5 Key Concepts

2.5.1 Deep Learning Concept

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to “learn” from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy.

Deep learning drives many artificial intelligence (AI) applications and services that improve automation, performing analytical and physical tasks without human intervention. Deep learning technology lies behind everyday products and services (such as digital assistants, voice-enabled TV remotes, and credit card fraud detection) as well as emerging technologies (such as self-driving cars).

Deep Learning Vs Machine Learning

If deep learning is a subset of machine learning, how do they differ? Deep learning distinguishes itself from classical machine learning by the type of data that it works with and the methods in which it learns.

Machine learning algorithms leverage structured, labeled data to make predictions—meaning that specific features are defined from the input data for the model and organized into tables. This doesn't necessarily mean that it doesn't use unstructured data; it just means that if it does, it generally goes through some pre-processing to organize it into a structured format.

Deep learning eliminates some of data pre-processing that is typically involved with machine learning. These algorithms can ingest and process unstructured data, like text and images, and it automates feature extraction, removing some of the dependency on human experts. For example, let's say that we had a set of photos of different pets, and we wanted to categorize by “cat”, “dog”, “hamster”, et cetera. Deep learning algorithms can determine which features (e.g. ears) are most important to distinguish each animal from another. In machine learning, this hierarchy of features is established manually by a human expert.

Then, through the processes of gradient descent and backpropagation, the deep learning algorithm adjusts and fits itself for accuracy, allowing it to make predictions about a new photo of an animal with increased precision.

Machine learning and deep learning models are capable of different types of learning as well, which are usually categorized as supervised learning, unsupervised learning, and reinforcement learning. Supervised learning utilizes labeled datasets to categorize or make predictions; this requires some kind of human intervention to label input data correctly. In contrast, unsupervised learning doesn't require labeled datasets, and instead, it detects patterns in the data, clustering them by any distinguishing characteristics. Reinforcement learning is a process in which a model learns to become more accurate for performing an action in an environment based on feedback in order to maximize the reward.

How deep learning works

Deep learning neural networks, or artificial neural networks, attempts to mimic the human brain through a combination of data inputs, weights, and bias. These elements work together to accurately recognize, classify, and describe objects within the data.

Deep neural networks consist of multiple layers of interconnected nodes, each building upon the previous layer to refine and optimize the prediction or categorization. This progression of computations through the network is called forward propagation. The input and output layers of a deep neural network are called *visible* layers. The input layer is where the deep learning model ingests the data for processing, and the output layer is where the final prediction or classification is made.

Another process called backpropagation uses algorithms, like gradient descent, to calculate errors in predictions and then adjusts the weights and biases of the function by moving backwards through the layers in an effort to train the model. Together, forward propagation and backpropagation allow a neural network to make predictions and correct for any errors accordingly. Over time, the algorithm becomes gradually more accurate.

The above describes the simplest type of deep neural network in the simplest terms. However, deep learning algorithms are incredibly complex, and there are different types of neural networks to address specific problems or datasets. For example,

- [*Convolutional neural networks \(CNNs\)*](#), used primarily in computer vision and image classification applications, can detect features and patterns within an image, enabling tasks, like object detection or recognition. In 2015, a CNN bested a human in an object recognition challenge for the first time.
- [*Recurrent neural network \(RNNs\)*](#) are typically used in natural language and speech recognition applications as it leverages sequential or times series data.

Model Building Process

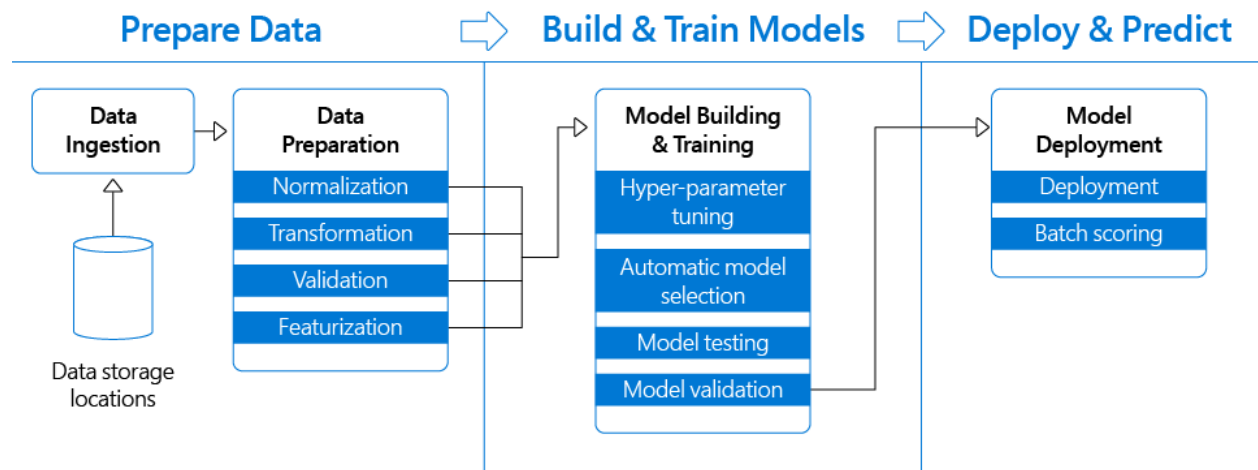


Figure 3 model building

1.5.2 Deep Learning Algorithm(s)

Transfer Learning (VGG19)

Transfer learning consists of taking features learned on one problem, and leveraging them on a new, similar problem. For instance, features from a model that has learned to identify racoons may be useful to kick-start a model meant to identify tanukis.

Transfer learning is usually done for tasks where your dataset has too little data to train a full-scale model from scratch. The process often involves taking the pre-trained weights in the first layers which are often general to many datasets and initializing the last layers randomly with and training them for classification purpose. Thus, in transfer learning approach, learning or backpropagation occurs only at the last layers initialized with random weights. Meanwhile, there are several approaches to transfer learning and the approach we use is dependent on the nature of our new dataset we want to classify with respect to the dataset of the pre-trained models. There are four main scenarios or cases of transfer learning.

The most common incarnation of transfer learning in the context of deep learning is the following workflow:

- i. Take layers from a previously trained model.
- ii. Freeze them, so as to avoid destroying any of the information they contain during future training rounds.
- iii. Add some new, trainable layers on top of the frozen layers. They will learn to turn the old features into predictions on a new dataset.
- iv. Train the new layers on your dataset.

A last, optional step, is **fine-tuning**, which consists of unfreezing the entire model you obtained above (or part of it), and re-training it on the new data with a very low learning rate. This can potentially achieve meaningful improvements, by incrementally adapting the pre trained features to the new data.

Keras applications have given users access to architectures such as VGG16, VGG19, RESNET and a lot more. There are 25+ models available and a user can choose their preferred application based on the amount of data their dataset holds, the top accuracies on ImageNet classifications, parameters, depth and size.

For the proposed project, VGG19 is used to train the tomato leaf disease detection model. VGG19 has a size of 549mb, a Top-1 Accuracy of 0.713, a Top-5 Accuracy of 0.900 on the ImageNet classification, 143,667,240 parameters, a depth of 26 and a time per inference step 84.75, 4.38 ms for CPU and GPU respectively.

For my case, I chose the VGG19 model for some reasons. First, even though it didn't win ILSVRC, it took the 2nd place showing nice performance. I only had 10 classes of images, so I thought VGG19 is enough for CIFAR-10. Second, VGG19 architecture is very simple. If you understand the basic CNN model, you will instantly notice that VGG19 looks similar. Third, I used google colab that provides GPU with Ram and Disk storage. This one is not the best choice,

but I thought it would be enough to run VGG19 even though VGG19 is a big in size. Lastly, since a lot of people uses VGG16, I wanted to try VGG19.

VGG19 Architecture

VGGNet is a Convolutional Neural Network architecture proposed by Karen Simonyan and Andrew Zisserman from the University of Oxford in 2014. The input to VGG based convNet is a 224×224 RGB image. Preprocessing layer takes the RGB image with pixel values in the range of 0–255 and subtracts the mean image values which is calculated over the entire ImageNet training set.

VGGNet 19 has 19 weight layers consisting of 16 convolutional layers with 3 fully connected layers and same 5 pooling layers. It also consists of two Fully Connected layers with 4096 channels each which is followed by another fully connected layer with 1000 channels to predict 1000 labels. Last fully connected layer uses softmax layer for classification purpose.

The first two layers are convolutional layers with 3×3 filters, and first two layers use 64 filters that results in $224 \times 224 \times 64$ volume as same convolutions are used. The filters are always 3×3 with stride of 1, After this, pooling layer was used with max-pool of 2×2 size and stride 2 which reduces height and width of a volume from $224 \times 224 \times 64$ to $112 \times 112 \times 64$. This is followed by 2 more convolution layers with 128 filters. This results in the new dimension of $112 \times 112 \times 128$. After pooling layer is used, volume is reduced to $56 \times 56 \times 128$. Two more convolution layers are added with 256 filters each followed by down sampling layer that reduces the size to $28 \times 28 \times 256$. Two more stack each with 3 convolution layer is separated by a max-pool layer. After the final pooling layer, $7 \times 7 \times 512$ volume is flattened into Fully Connected (FC) layer with 4096 channels and softmax output of 1000 classes.

VGG 19

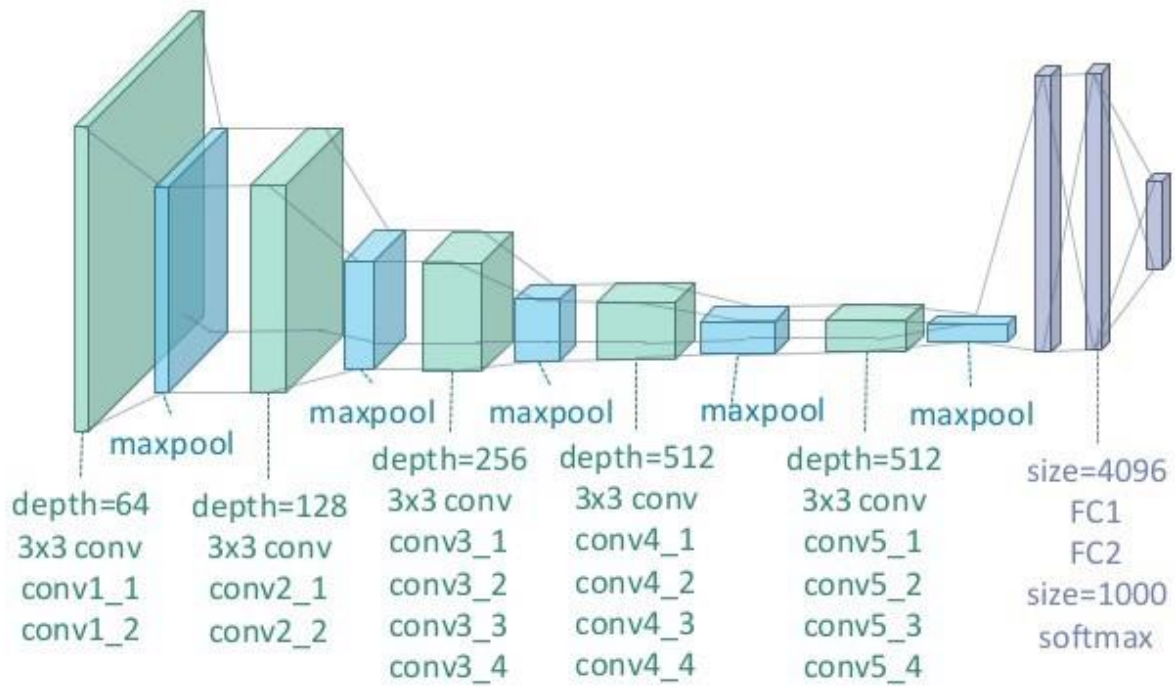


Figure 4 VGG19 architecture

To begin the model building process using VGG19, the required libraries are first imported and then the inputs are then preprocessed. The dataset used on tomato leaf recognition had 21,317,794 images categorized into 10 classes. The image size used in the network is 256x256. Once the model is imported and the output layer changed, to avoid retraining the layers, the model is then trained for specified epochs and an accuracy of 74% on the validation data is achieved. An accuracy vs loss for VGG19 is then plotted to give a visual picture of the disparity. If the model reaches the desired accuracy on required changes, the model is saved and is ready for deployment.

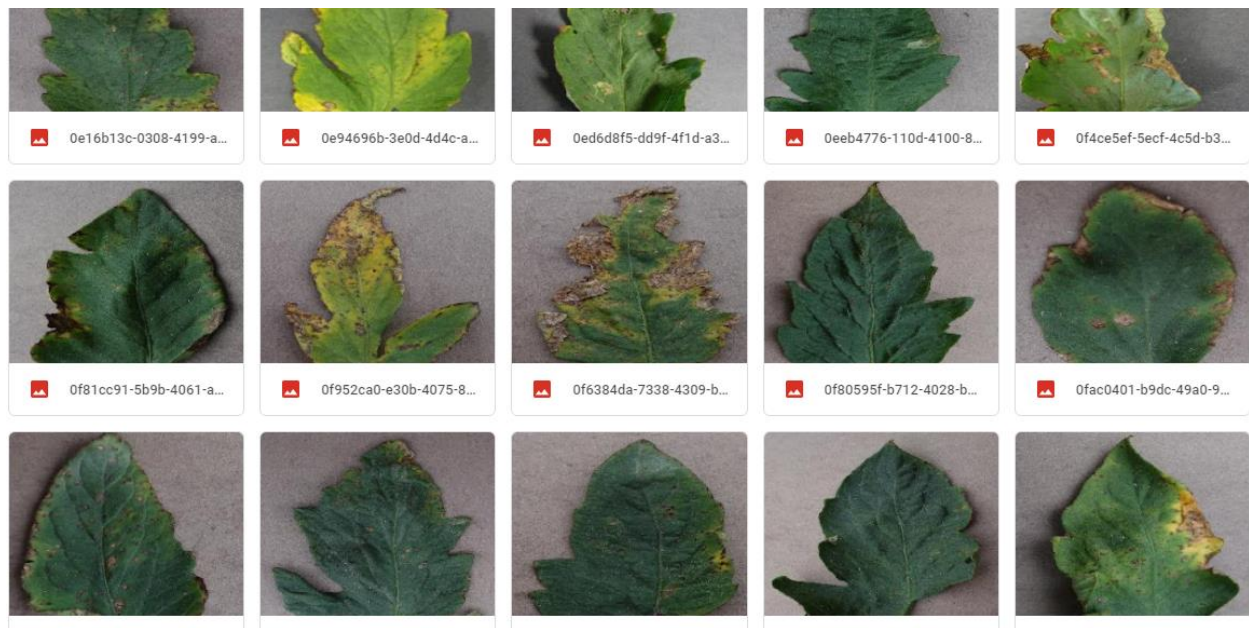


Figure 5 dataset preview

TensorFlow Deep learning Library

Tensor Flow is an open source software library that uses data flow graphs to do numerical computations. The mathematical processes are represented by the graph nodes, while the multidimensional data arrays (tensors) that flow between them are represented by the graph edges. Without rewriting code, this flexible architecture allows you to distribute computing over one or more CPUs or GPUs in a desktop, server, or mobile device.

TensorFlow was developed by researchers and engineers at Google's Machine Intelligence Research organization's Google Brain team for the aim of conducting machine learning and deep neural network research. The system is generic enough to be used in a variety of other fields as well.

To train my model online, I used a Google Colab GPU with 4GB RAM. To attain the best results, the training required around an hour.

CHAPTER 3: SYSTEM ANALYSIS AND DESIGN

3.1 METHODOLOGY

Considering the limited timeframe of the project and limited resources, it is compulsory to follow a carefully guided plan. Following components should be consider when deciding a software development methodology is being decided.

- The size of the venture
- How explicit necessities are
- How much the client will need to change requirements?
- How large the development team is?
- Due date

The project should be designed and developed with in a very limited time framework and the design, development, testing and deployment all done by a single person, it is important to follow a well-chosen software development approach in order to reduce the possibility of failing to fulfill the planned project deadlines. As stated above the project will be developed by one person with in a very tight time framework it is anticipated that there will be regular meetings with the supervisor when designing the program to ensure that the system is implemented in compliance with standards and specifications. Following table presents a comparison between traditional SDLC methodologies and agile methods.

	Traditional Methodologies(Ex: waterfall)	Agile methods
Adaptability to change	Change sustainability	Change adaptability
Development approach	Predictive	Adaptive
Development Orientation	Process oriented	People oriented
Project Size	Large	Small/Medium
Planning Scale	Long term	Short term
Management Style	Command and control	Leadership and collaboration
Learning	Continuous learning while development	Learning is secondary to development
Documentation	High	Low

Figure 6 comparison between agile and traditional methods

Waterfall methodology is not a fit to the project since the time frame is very limited, waiting each step to finish is not practical and ideal. RAD methodology can be excluded because of the limited resources and hence only one person is developing the system. Considering all the most widely used SDLC methodologies Agile's SCRUM methodology would be a perfect fit for this project.

3.1.1 APPROACH

3.1.1.1 Requirement Specification and Analysis

This phase of the system development life cycle entailed gathering important information for the system's development. The data and information are also analyzed in order to create a logical model of the system that will be used in the design phase. Informal interviews, observations, and online research were used in the collection of data and information on system needs. The information gathered was utilized to create a logical model that explains how the system works.

3.1.1.2 Design

This step entails the building of a working prototype that meets all of the system's defined specifications. It entails creating a user interface and determining the system's inputs, outputs, and procedures.

3.1.1.3 Implementation and Coding

This stage comprises the translation of the developed designs into an actual system utilizing proper programming language. Python, flask is used to develop the deep learning web api while react native cross platform mobile app programming is used for frontend development.

3.1.1.4 Testing

This is the testing of the inputs, outputs and functionality of the system during the various stages of the software development life cycle..

3.2 SYSTEM ANALYSIS

This stage entails analyzing a system, identifying flaws, and utilizing the data to enhance the system. It also aids in determining how things are done and whether or not certain requirements are worthwhile to pursue, as well as the potential rewards of doing so.

3.2.1 Data Collection

Research

Performing a primary research is important in order to get along with the system specifications and a good design of a system because system requirement specifications and client needs can be established through a proper primary research. This form of research was necessary to determine client needs, problems that they currently face and how much impact would hold if the project succeed.

Through surveys that performed with farmers and locals highly important and critical information was discovered in this project. This information can be emphasized as given below. Surveys were performed using paper surveys since some of the old farmers in the area didn't know how to use google docs. The surveys consists with a set of questions which were divided in

two sections addressing the problem and how their attitude towards current systems and how the users of the proposed system would think that the application would be advantageous. Physical copies of following questionnaires were shared with 21 people in the area including mainly targeted participants in order to obtain necessary information.

All 21 participants has responded to the survey questions and answers were successfully recorded. Following snap shots will represent one of the responded survey questionnaires by the participants.

Obtained information through the survey can be divided in to sections and visualized as below.

Are you farming?

As following pie chart emphasized 100% of the participants do cultivation according to the survey reports. Hence it is proved that the appropriate participants has been selected regarding the completion of the survey

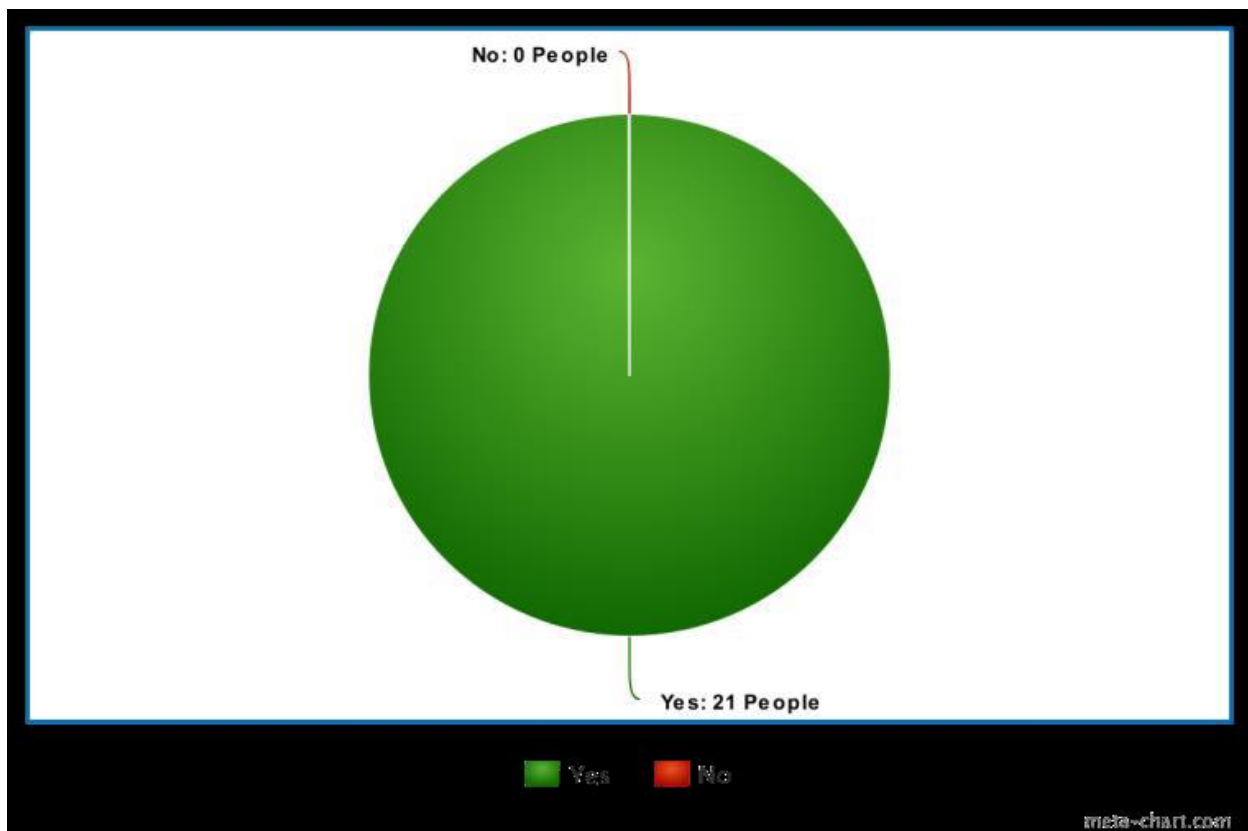


Figure 7 farmers pie-chart

Internet Availability

Above 97% had a stable internet facility in their homes which is really necessary when deciding system requirement specifications. Even though others didn't have a stable internet connection all of them had access to internet through their smartphones.

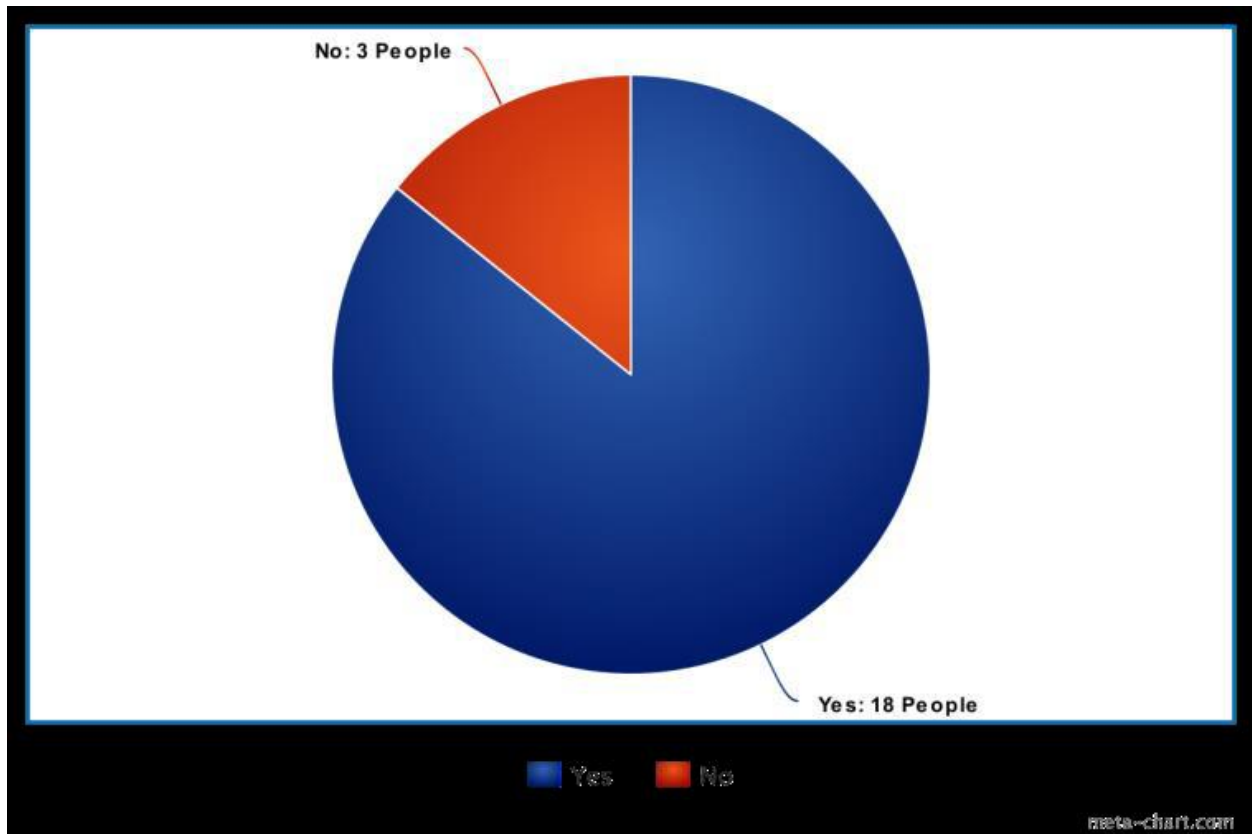


Figure 8 internet availability pie-chart

Smart device usage

According to the survey 94% used a smartphone more often which is necessary when deciding the platform for the application.

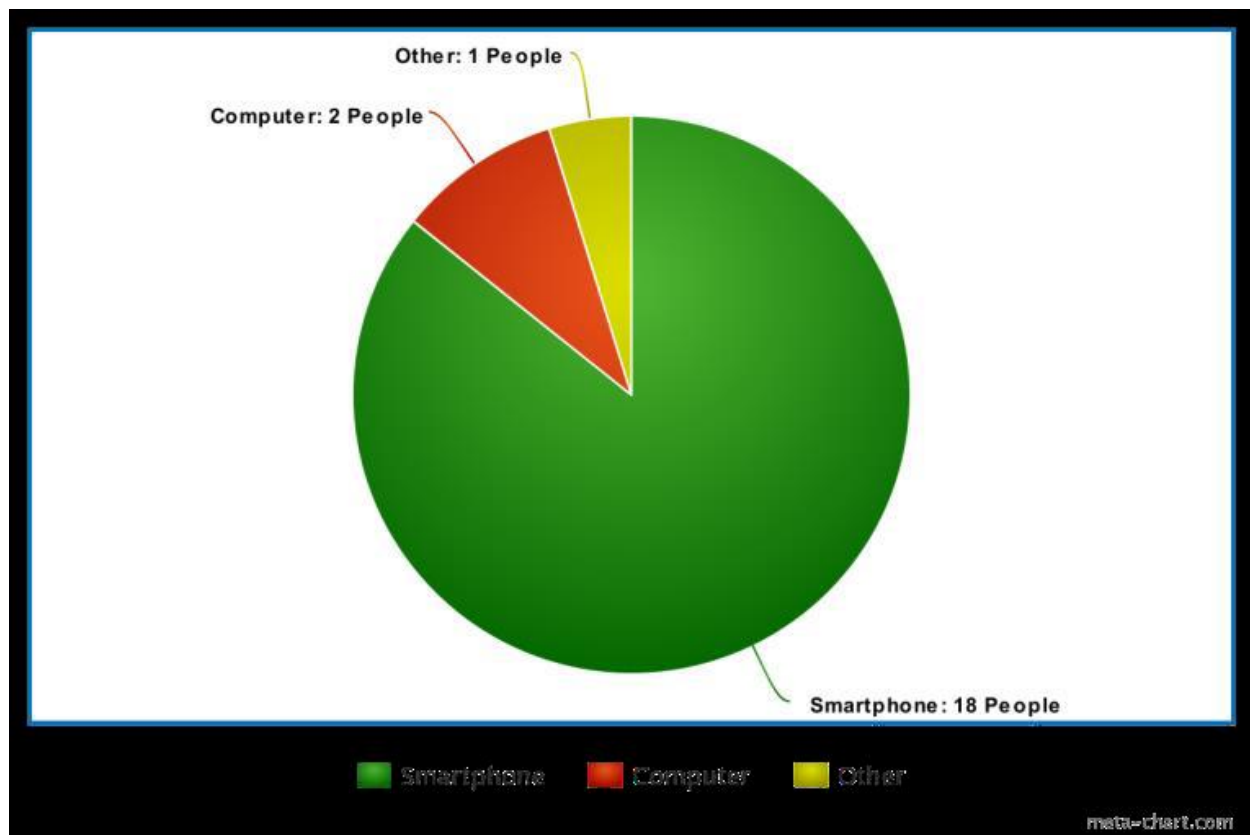


Figure 9 smart phone usage pie-chart

Crop losses because of plant diseases

Over 87% of the participants have faced to crop losses because of various kinds of plant diseases.

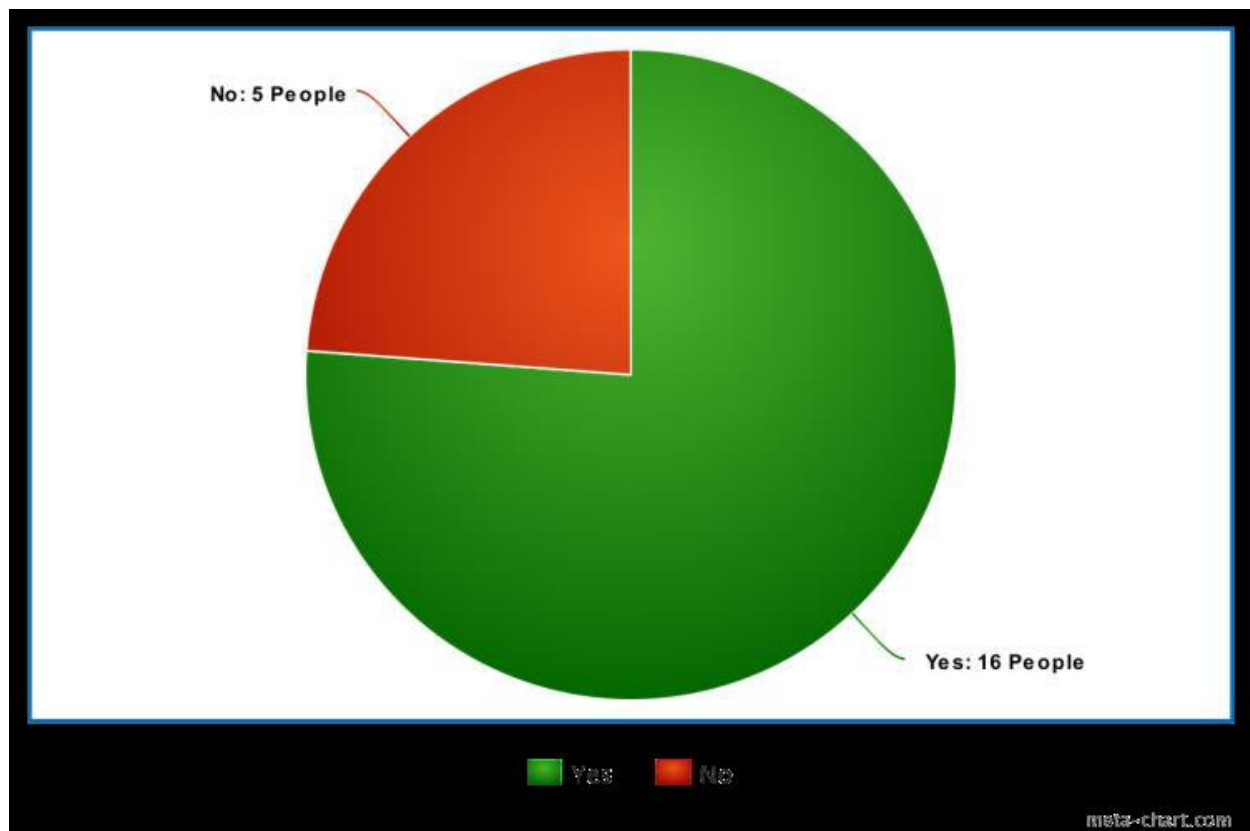


Figure 10 farmers who suffered from crop loss due to diseases

Plant disease identification methods

According to reports of the surveys following pie chart represents the method that participants chose in order to identify plant diseases.

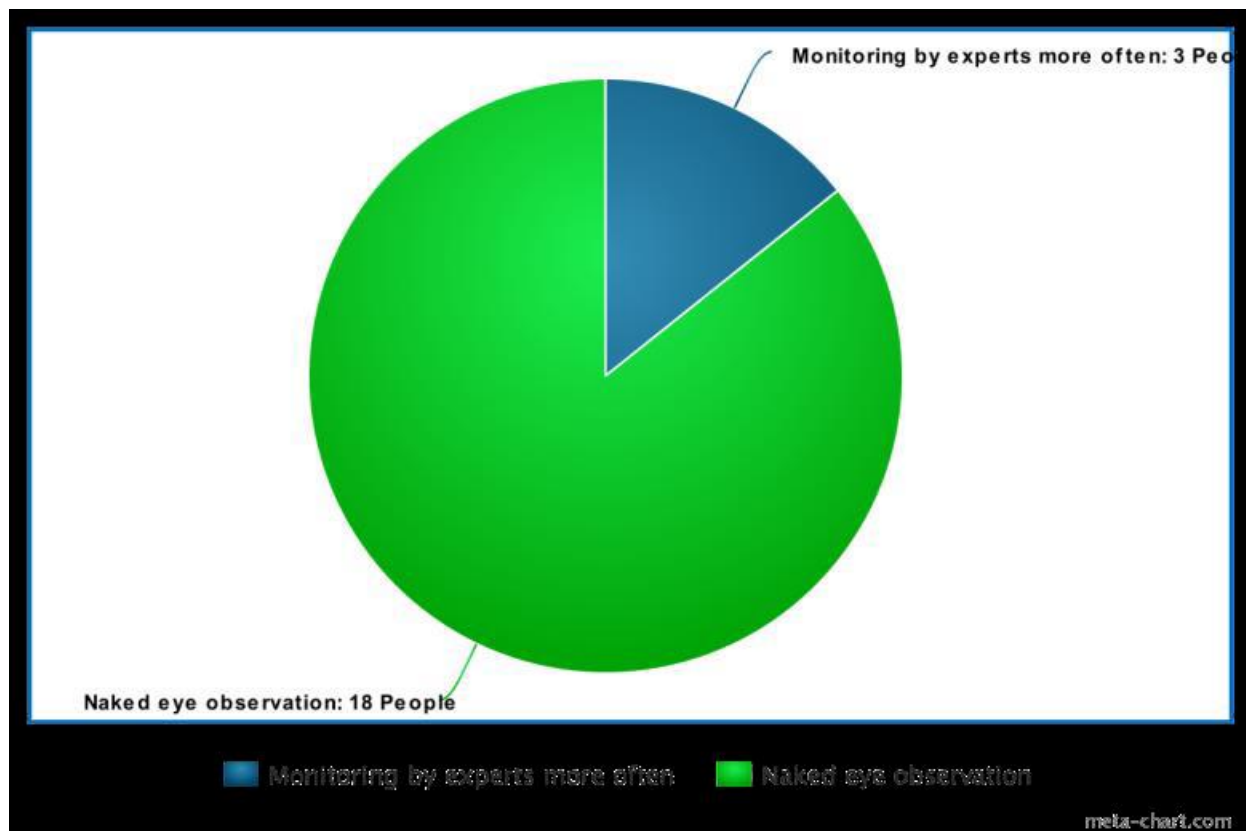


Figure 11 disease identification methods

Informal Interviews

I distributed a questionnaire to various agriculture students and interviewed a few of them at the start of the study. I also interviewed SAWE agrovet sellers. The interview was aimed at understanding the problem and finding out if the problem was valuable to be tackled. I also did one-on-one interviews with tomato farmers in Nakuru in order to gain a better understanding of the situation from their perspective, as well as the tomato growing process and the problems they experienced from planting through harvesting.

Online Research

The internet was the major source of information through online materials such as research papers, publications, agricultural journals, online videos and tutorials. Online data was preferred since the information is readily available and in abundance.

3.2.2 Feasibility Analysis

This is the process of measuring the viability of the project in various aspects which include:

Economic Feasibility

This is a metric for determining the project's or solution's cost-effectiveness. It entails determining how much money will be spent on the diagnostic system's development as well as

the system's benefit or returns. Because the majority of the tools, resources, and tools required are freely available and accessible, the project is both economical to construct and maintain.

Technical Feasibility

This entails ensuring that the software, hardware, and personnel needed to build and run the system are all available. The proposed system can be implemented with the help of experience and tools.

Operation Feasibility

This is a criterion for determining the urgency of a problem or the acceptability of a solution, as well as assessing the solution's effectiveness. Its goal is to determine whether the problem is worth fixing, the solution's viability, and the users' thoughts and attitudes regarding the solution being developed.

Based on the responses to the interviews and questionnaires, it was determined that the proposed solution would be viable, since it would assist farmers in making accurate disease diagnoses and agricultural chemical dealers and wholesalers in stocking the appropriate items.

Schedule Feasibility

The analysis determines whether the project deadline is feasible and whether the solution can be conceived and implemented in a reasonable amount of time. According to the project's work plan set during the project start, there will be sufficient time to develop, test, and assess the system successfully.

3.1.4 System Analysis Models

Use Case Diagrams

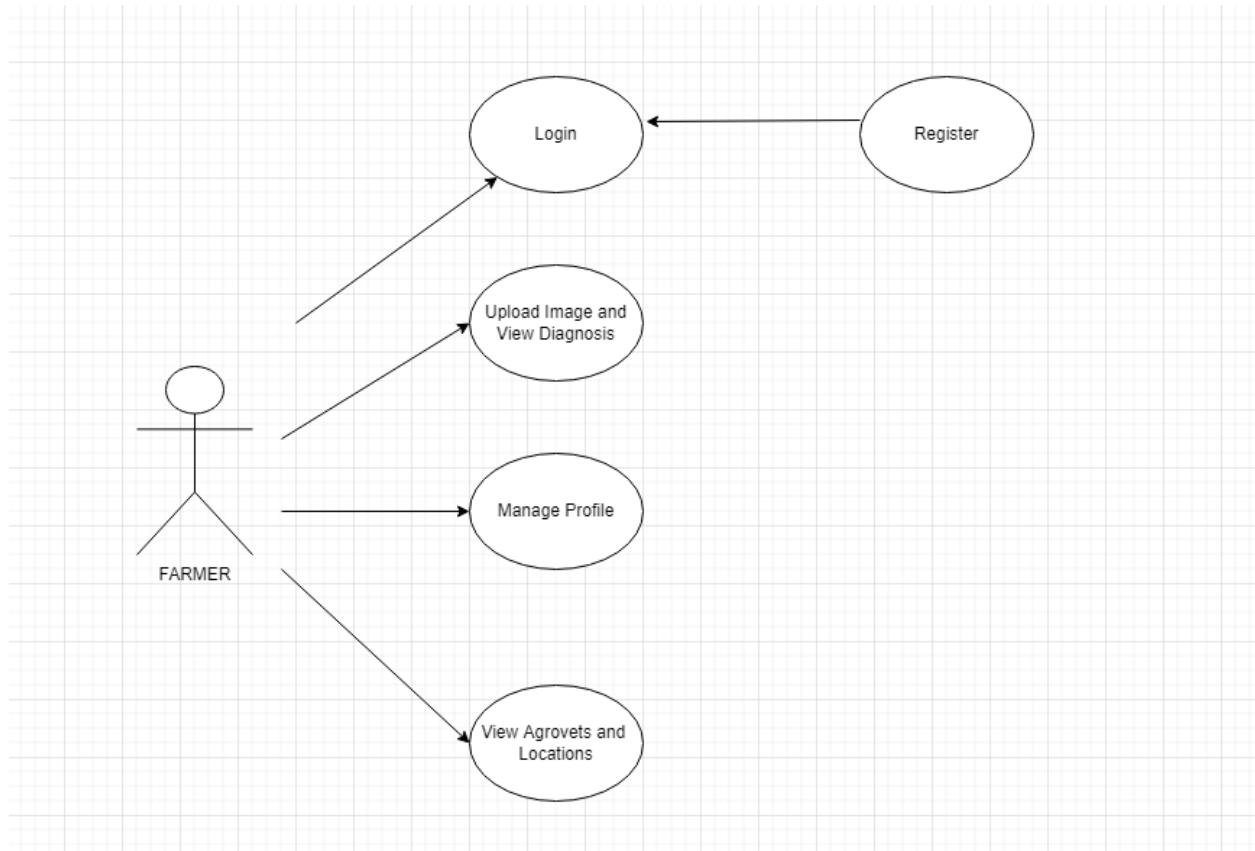


Figure 12 farmer use case

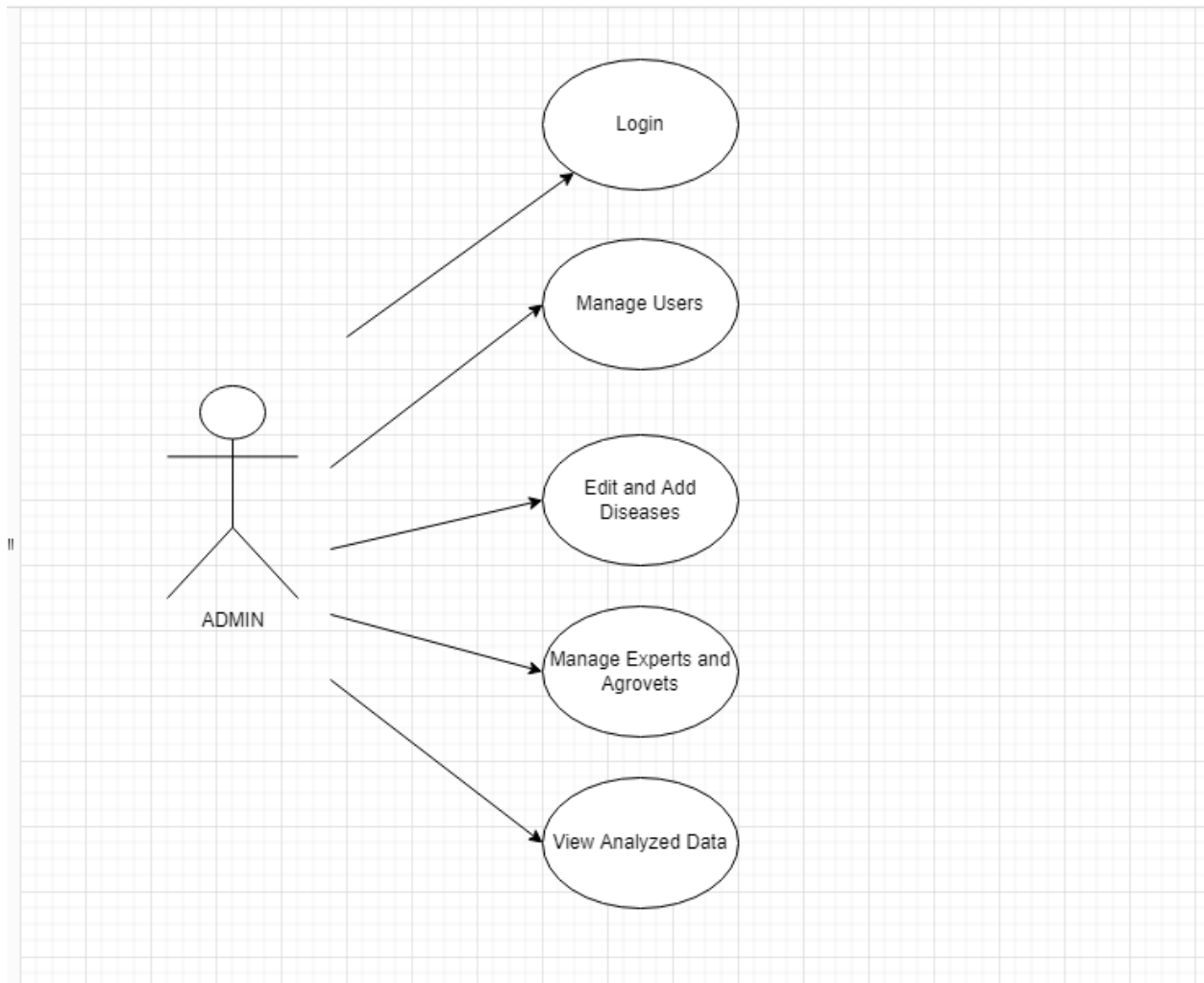


Figure 13 admin usecase

Activity Diagram

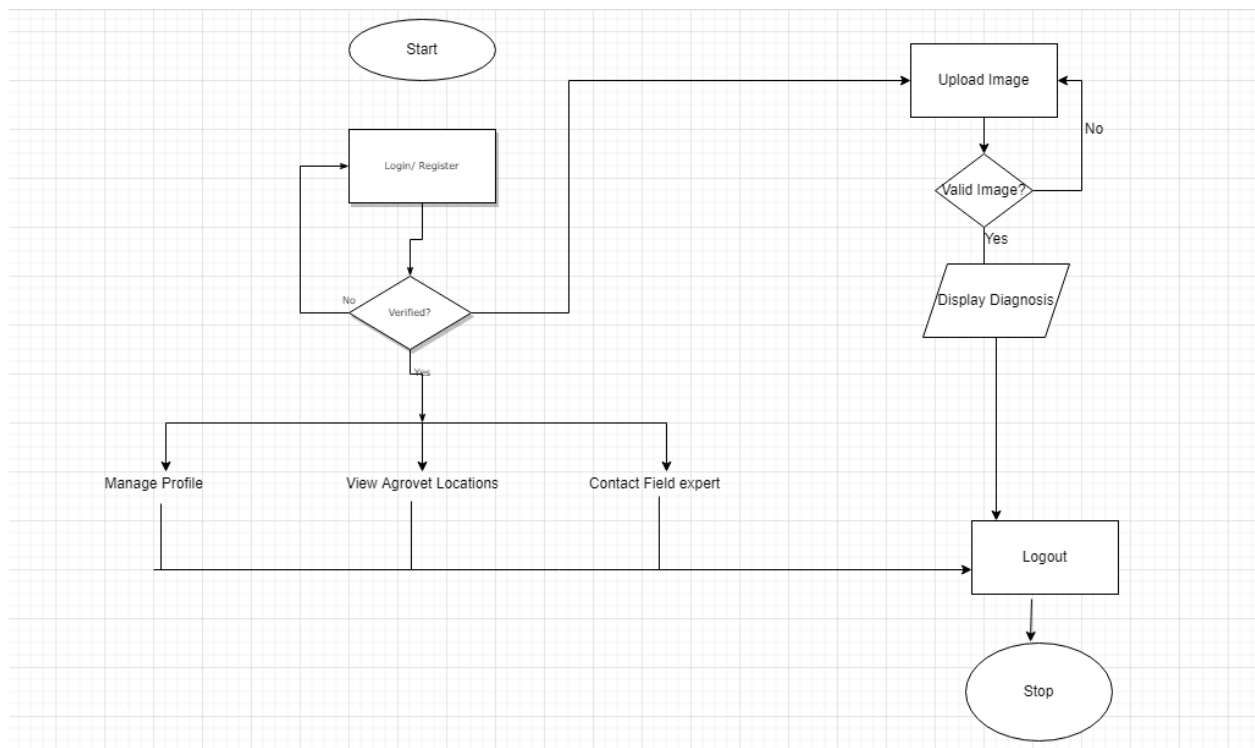


Figure 14 farmer activity diagram

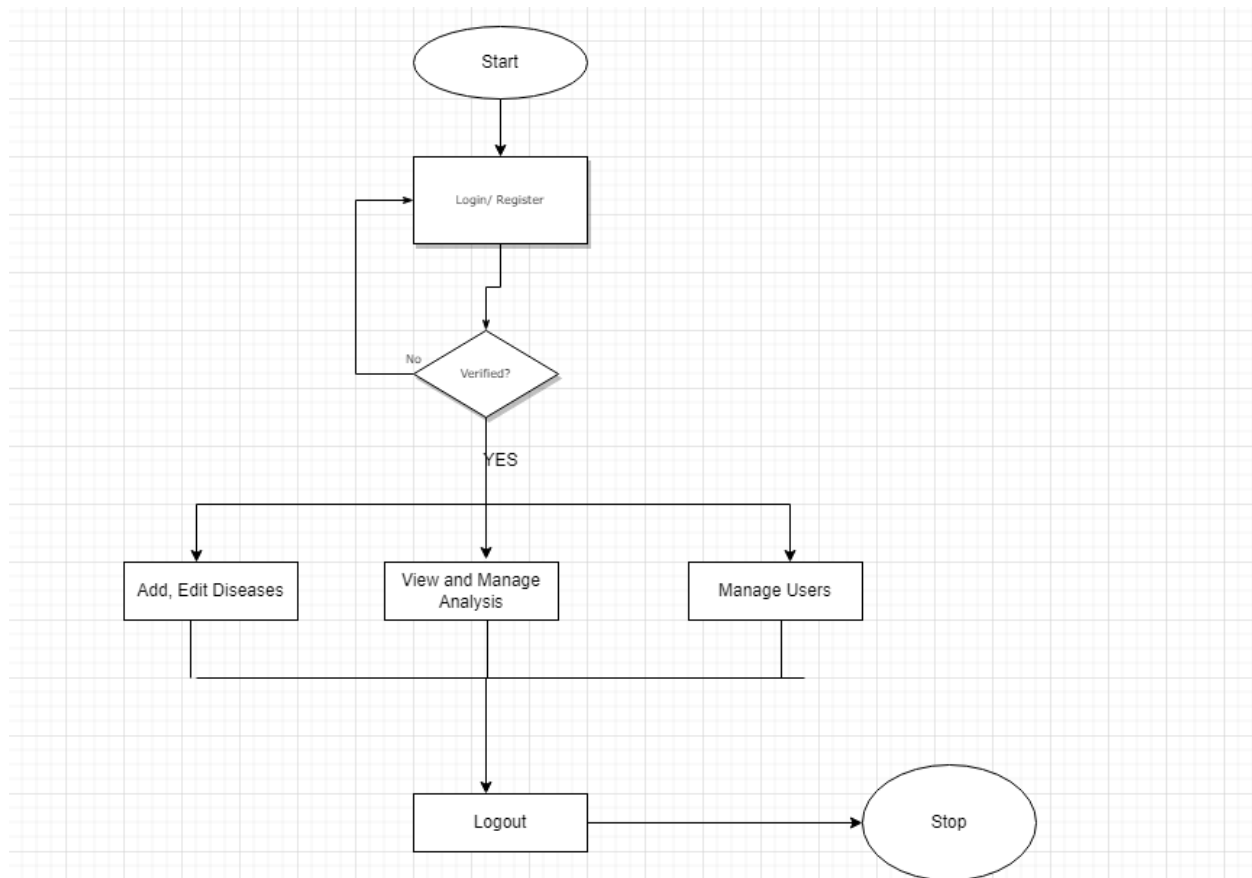


Figure 15 admin activity diagram

Use case specifications

Use Case Name:	Upload plant leaf image
Description:	A user uploads an image of a plant leaf.
Primary Actors:	User
Main_Flow:	<ol style="list-style-type: none"> 1. The use case begins when the user presses “Take a picture” button. 2. The device camera screen is opened and instructions are displayed. 3. User takes a photo of the plant leaf. 4. User presses “✓” button.

	<p>5. The uploaded image is sent to the server side application (back-end application).</p> <p>6. The backend application recognizes the image and detect plant condition (plant disease/ healthy) along with the accuracy level and sends as the response to the client-side app.</p> <p>7. The use case ends while displaying received response (sent image, confidence level/accuracy level and plant condition) in the application UI.</p>
Alternative-Flow:	<p>1A Selecting some other function</p> <p>1. Exits from the use case.</p> <p>2A Presses “Choose from gallery” button</p> <p>1. The device gallery is opened.</p> <p>2. User selects image/images from the gallery.</p> <p>3. Directs to BF 4</p> <p>4A Presses “Cancel” button</p> <p>1. Exits from the use case.</p> <p>6A If the image is not a plant leaf image</p> <p>1. System recognized provided image and user will be notified</p> <p>2. Display a warning message saying “Plant leaf image is required”</p> <p>3. Directs to BF 2</p> <p>6B If the image is corrupted</p> <p>1. Display Error message</p> <p>2. Directs to BF 2</p>
PostConditions:	<ul style="list-style-type: none"> • The application database is updated with the relevant details. • User can upload more images to prediction.

3.1.5 Function Specifications

This is the process of describing functionalities and features expected to be implemented in a system as well as the constraints expected during the development of the proposed solution.

Functional Requirements

These are the characteristics or qualities that must be present in the system in order to meet a certain requirement. They are quantifiable and stated regardless of implementation.

Upload image of plant leaf

The application should be able to upload a single or multiple images of plant leaves to the application through device camera or choosing from gallery.

Detect plant leaf

The application must be able to identify whether the image given is a plant leaf image or something else. If the image is not a plant leaf image the user should be notified.

Recognize plant disorder

The application must be able to recognize crop and crop disease by evaluating them with data that has been stored in the dataset and the database and predict plant disorder precisely. If plant type is not supported in the system user should be notified.

Search plant diseases

The system should provide features to the user to search for various kinds of plant diseases and find relevant information through the system.

Non - Functional Requirements

The system's user-visible features are usually qualitative in nature.

Accuracy

Application must be accurate with the predictions of plant diseases to avoid misleading the users.

User-Friendly

The system must be consists with an easy to use GUI because the users of the application may do not have technical expertise.

Performance

Hence, the application is mainly about detection crop disorders the process heavily relies on time disease recognition process should be much efficient and accurate.

Scalable

The system must be flexible with the future changes and upgrades.

3.2 SYSTEM DESIGN

It entails expounding on the research carried out in order to piece each component into a coherent application or system model. It emphasizes technical or implementation problems while

focusing on how the system will be created by describing structural components. Mainly the system can be divided in to three main parts as follow;

- Diseased plant image uploading process
- Plant leaf recognition process
- Disease identification process

The following figure explains the key components of the system.

3.2.1 System Architecture

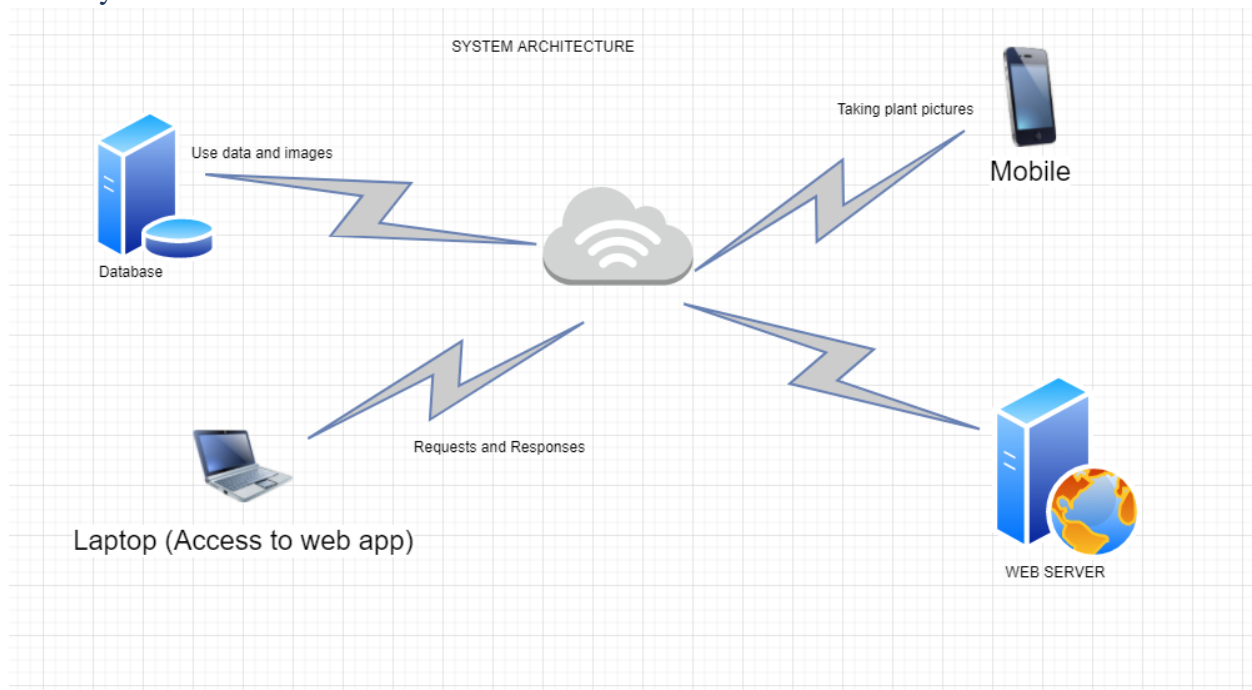


Figure 16 system architetcure

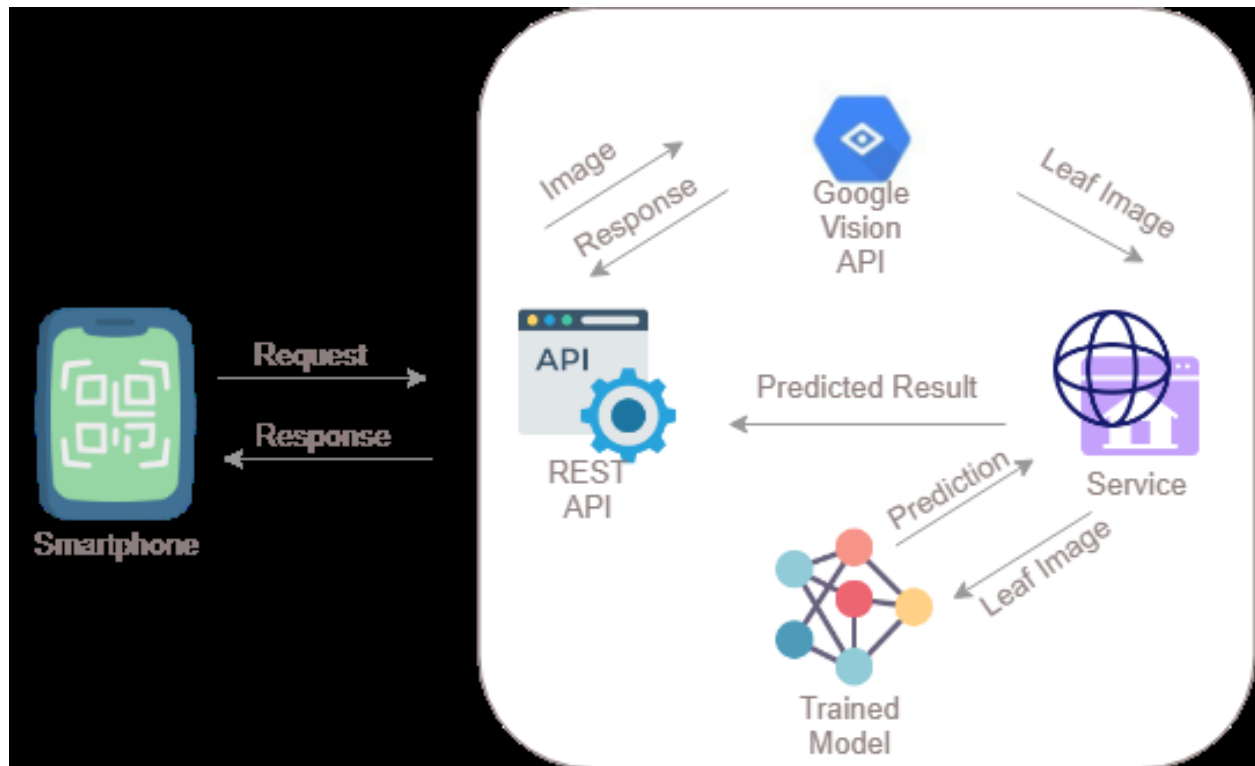


Figure 17 component level diagram

3.2.2 Data Flow Diagrams

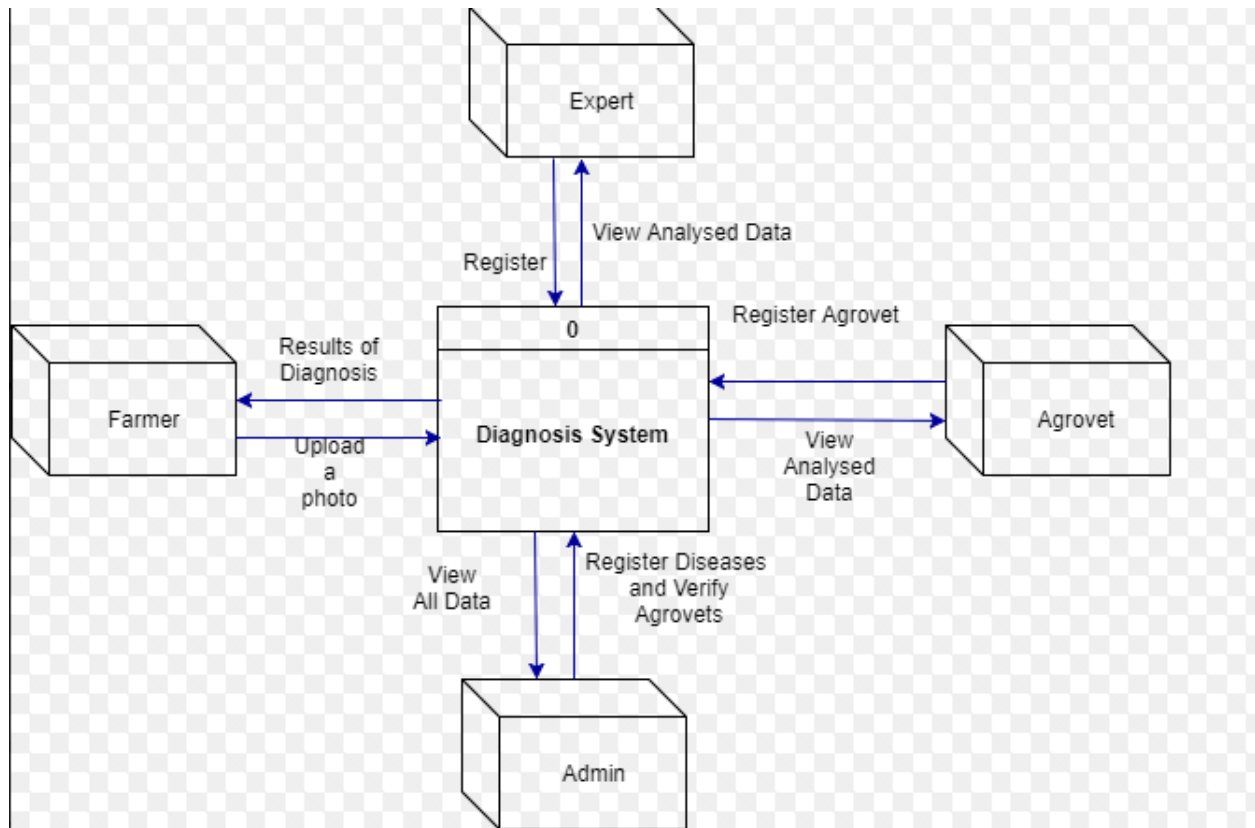


Figure 18 data flow diagram

3.2.3 Database Design

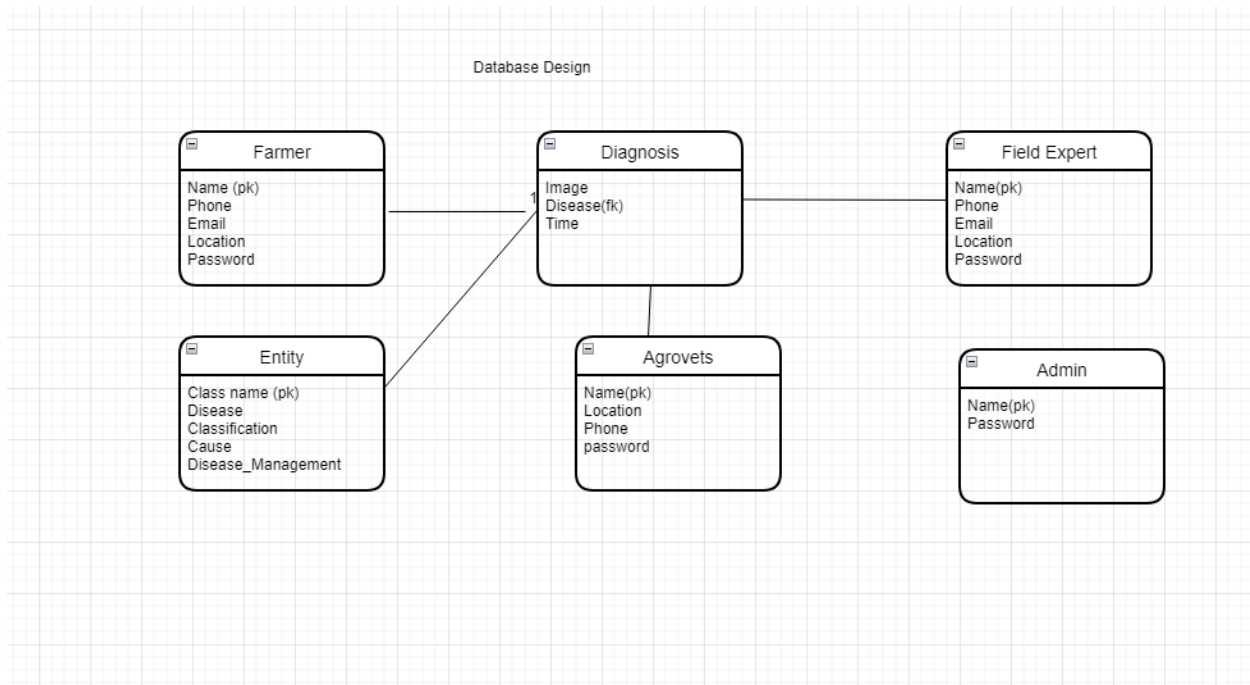


Figure 19 database design

3.3 IMPLEMENTATION AND TESTING

The implementation of the proposed system is covered in this phase. It entails the development and testing of the software, as well as the hardware platform used, the programming language used, and the test data and outputs.

3.3.1 Resources

3.3.1.1 Hardware Resources

1. HP Elite book 840- Laptop
2. RAM - 64GB
3. Processor - Intel Core i5
4. Hard Disk – 1TB
5. GPU 4GB

3.3.1.2 Software Resources

Operating systems:

- Microsoft Windows 10
- Android 8.1 emulator

Languages:

- Python
- JavaScript
- Flutter

Libraries:

- Tensorflow
- OpenCV
- NumPy
- Flask
- Sea-born
- Pandas
- Split-Folders

Frameworks

- Flutter

Database:

- Mysql + hive

Virtual Machine:

- Google Colabs (For model training purposes)

IDEs:

- Pycharm
- Visual Studio Code

3.3.2 Choice of Programming Tools and Techniques

3.3.3 Testing

Testing is the process where one tests that all the components and modules building the system are actually working.

Test plan

Every test method is designed to detect a particular type of product malfunction. However, all test types are designed to achieve a common objective of "early identification of all defects before the product is released to the user".

The following types of testing should be performed prior to publication of the application.

Unit test – Since the application will be developed feature by feature testing the smallest piece of verifiable software is essential.

API test – Test the flask API's built for the backend of the application.

Integration testing – Individual application components will be combined together and will be tested as a one component.

System test – Conducted on a complete, integrated platform to check the compliance of the system with its particular requirements.

The following processes were performed on various system units

1. Unit Testing
2. Integration Testing
3. System Testing

4. User Acceptance Testing

3.3.3.1 Unit Testing

Unit testing tests each program part to ensure each single program is fully tested since it addresses the testing of functional units within a system as the main building blocks. A test plan is created first which consists of a number of test runs, such as the valid paths through the code, and the exception and error handling paths. For each test run there is a list of conditions tested, the test data used and results expected. This ensures then each unit of the project under development is fully functional.

3.3.3.2 Integration Testing

For a system to be complete the separate modules have to be interfaced so that they can work as one. This form of testing was carried out systematically using the modified sandwich integration testing where a functional group is extensively tested before moving on to another functional group.

3.3.3.3 System Testing

Usually done after all other components have been successfully integrated together. Test data is used to assume the normal use of the system. Black box testing was also done to ensure all the functionalities of the system were comprehensively covered and work as expected.

3.3.3.4 User Acceptance Testing

These are formal testing with respect to user needs, requirements and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system. Any component that failed to satisfy the acceptance criteria was modified

Accuracy Testing

Parameter Evaluation

In a well performing machine learning model the difference between actual predicted data and the training data should be minimum. In order to archive this, the machine learning model should be trained avoiding model overfitting and model underfitting. To avoid model overfitting and underfitting it should determine the suited parameters of the training model. To find that it should keep changing the parameter values and check the final output.

In this scenario the final model has been selected after 72 attempts. In an underfitting model the model can't perform well in both validation and training datasets while in an overfitting model it can perform very well on the training dataset but perform poorly on the validation dataset. In a good machine learning model the accuracy of the model should be increased in epoch and the loss should be decreased. The accuracy of the model can be depend on various parameters ex. Number of layers, data splitting ratio, filter values and etc. Following graphs represents the accuracy and the loss of the finally selected model over the number of epochs.

```
plt.plot(h['accuracy'])
plt.plot(h['val_accuracy'], c='red')
plt.title('acc vs val-acc')
plt.show()
```

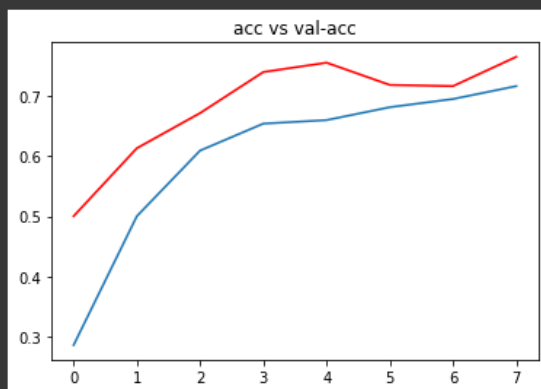


Figure 20 accuracy vs val accuracy

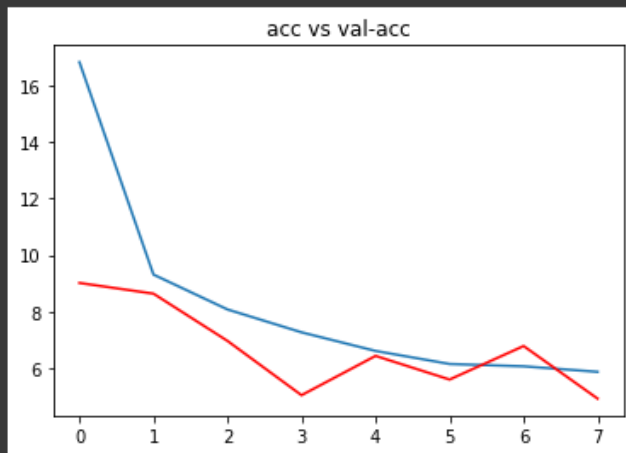


Figure 21 loss vs val_loss

As the above graphs shows the accuracy of the training model has been increased from 1st epoch to 10th epoch and the accuracy of the model in the final epoch is 93.75% and the loss is getting decreased in each epoch, the loss of the 10th layer of the model is 16%. In order to gain this results it has used 4 convolutional layers and 2 dense layers

After training model, I tested it with test images given in dataset which is around 1000. Accuracy you can see below i.e., 94.5%

```
[ ] print("[INFO] Calculating model accuracy")
    scores = newModel.evaluate(testing, y)
    print(f"Test Accuracy: {scores[1]*100}")

[INFO] Calculating model accuracy
32/32 [=====] - 10s 266ms/step - loss: 0.2798 - accuracy: 0.9460
Test Accuracy: 94.59999799728394
```

model accuracy

Figure 22 model accuracy

I have also generated confusion matrix and classification report. As confusion matrix can give you a better idea of what your classification model is getting right and what types of errors it is making. And classification report displays the precision, recall, F1, and support scores for the model. This gives a deeper intuition of the classifier behavior over global accuracy which can mask functional weaknesses in one class of a multiclass problem.

```
Confusion Matrix
[[ 97   2   0   0   1   0]
 [  2 286   2   3   6   1]
 [  0   2  95   0   2   1]
 [  0   1   0  91   2   6]
 [  0  11   0   0 183   6]
 [  1   2   0   0   3 194]]
```

Figure 23 confusion matrix

After concluding report, we can observe how much accurate that will give true values as that belongs to particular class. I tested model by giving image from internet and it shows below:

```
img = "/content/gdrive/MyDrive/tomatoleaf/example/leafmold.jpg"
image1 = convert_image_to_array(img)
image1 = image1.reshape((1,) + image1.shape)
pred = newModel.predict(image1)
pred1 = np.argmax(pred)
print(lenc.classes_[pred1])
plt.imshow(cv2.cvtColor(image1[0], cv2.COLOR_BGR2RGB))
plt.title(lenc.classes_[pred1])
plt.show()
```

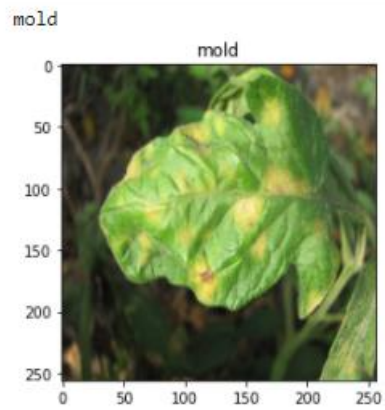
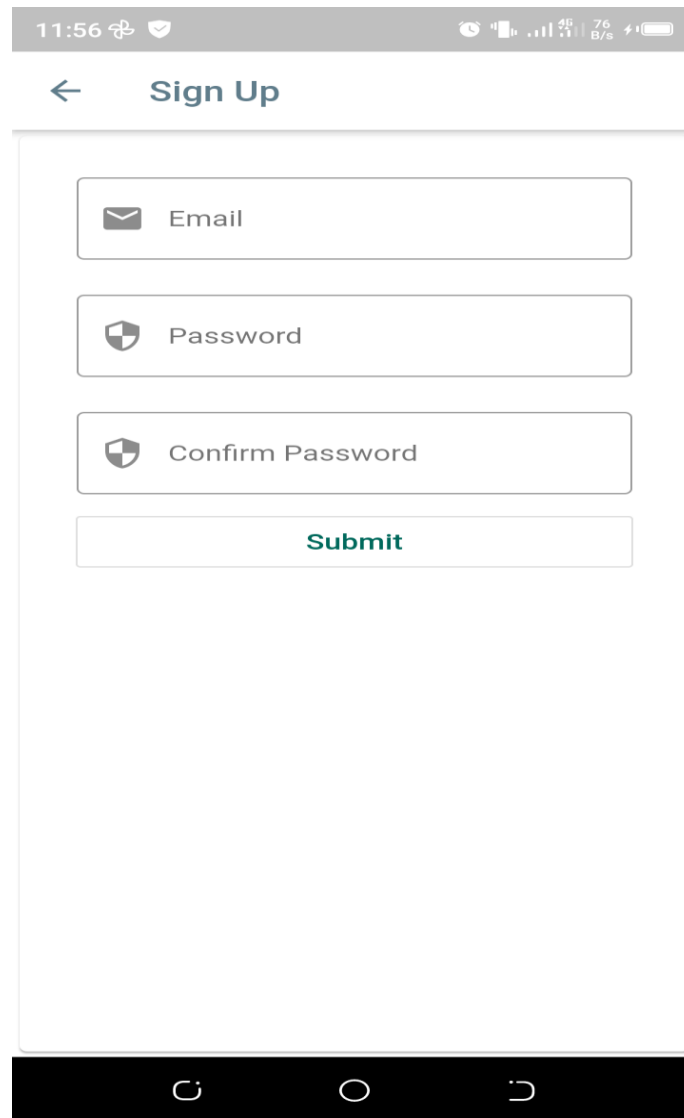


Figure 24 model test

App Screens



The image shows a mobile application screen for signing up. At the top, there is a status bar with the time 11:56, a signal strength indicator, a battery level indicator, and a 76% battery percentage. Below the status bar is a navigation bar with a back arrow and the text "Sign Up". The main content area contains three input fields: "Email" with an envelope icon, "Password" with a shield icon, and "Confirm Password" with a shield icon. Below these fields is a "Submit" button. The bottom of the screen features a black navigation bar with three white icons: a home button, a search button, and a list button.

11:56

← Sign Up

Email

Password

Confirm Password

Submit

Figure 25 sign up

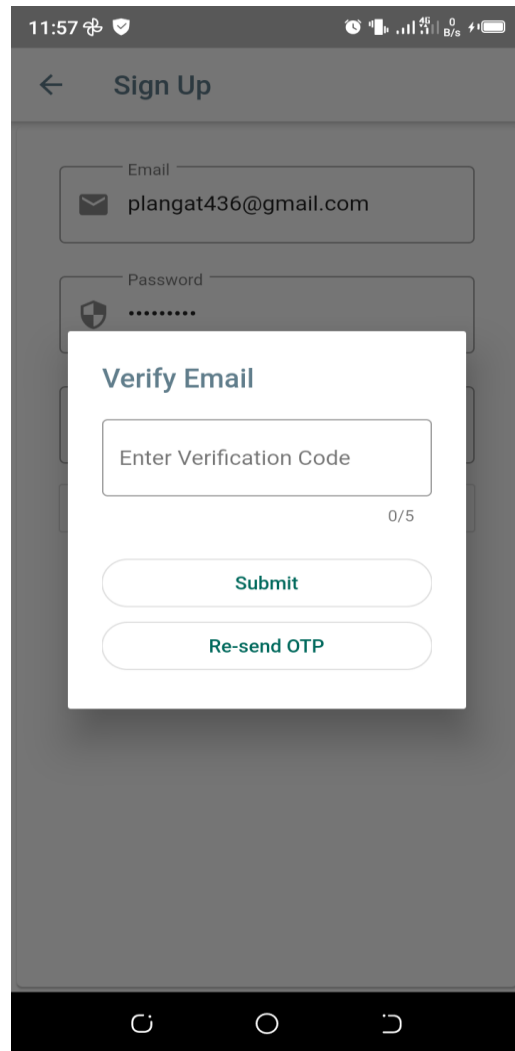
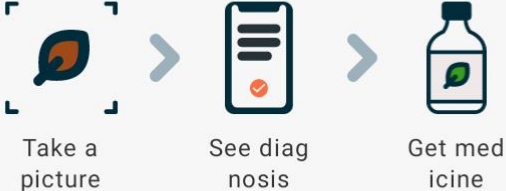


Figure 26 verify sign up

Heal your crop



Take a picture See diagnosis Get medicine

Take a picture

Previous pictures

[View All](#)



Weather

Ongata Rongai, 29 May

15.7°C

Sunset 6:33 PM

Possible light rain in the evening. 🌧️ 65%



Figure 27 Home screen

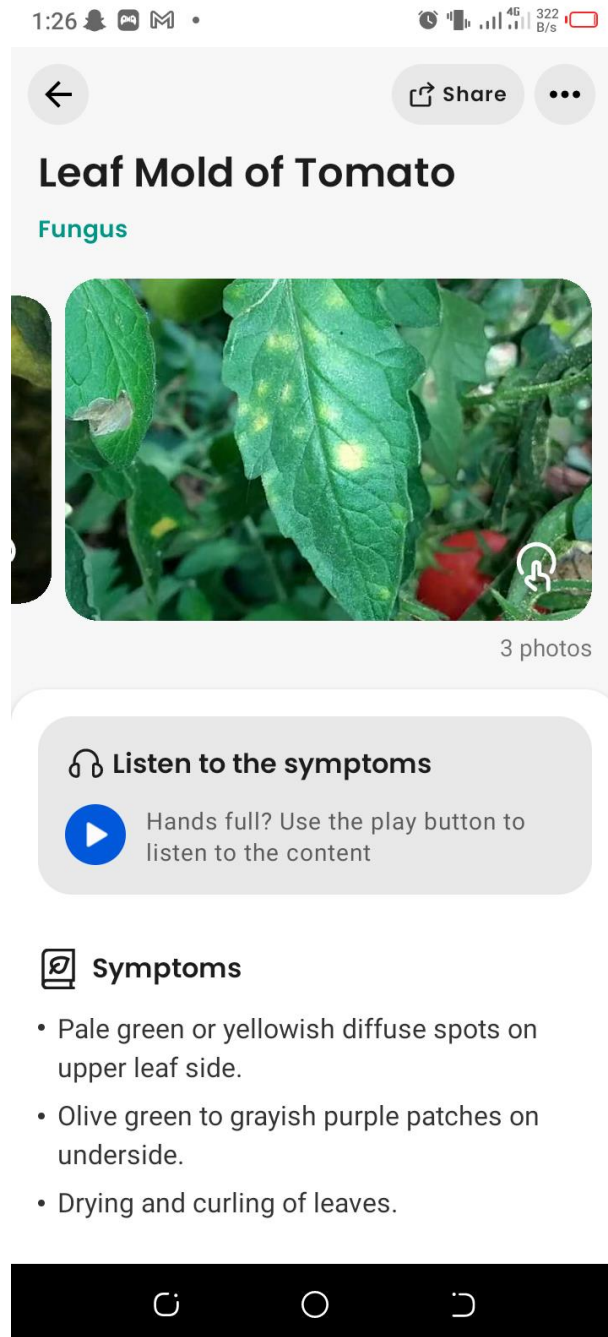


Figure 28 disease diagnosis results

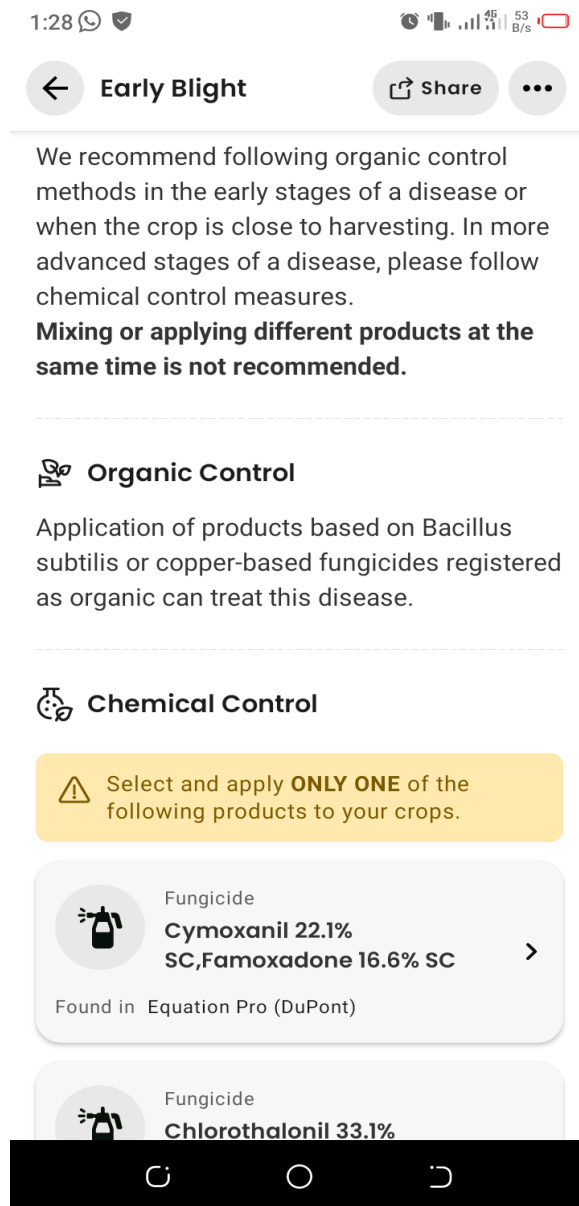


Figure 29 disease treatment

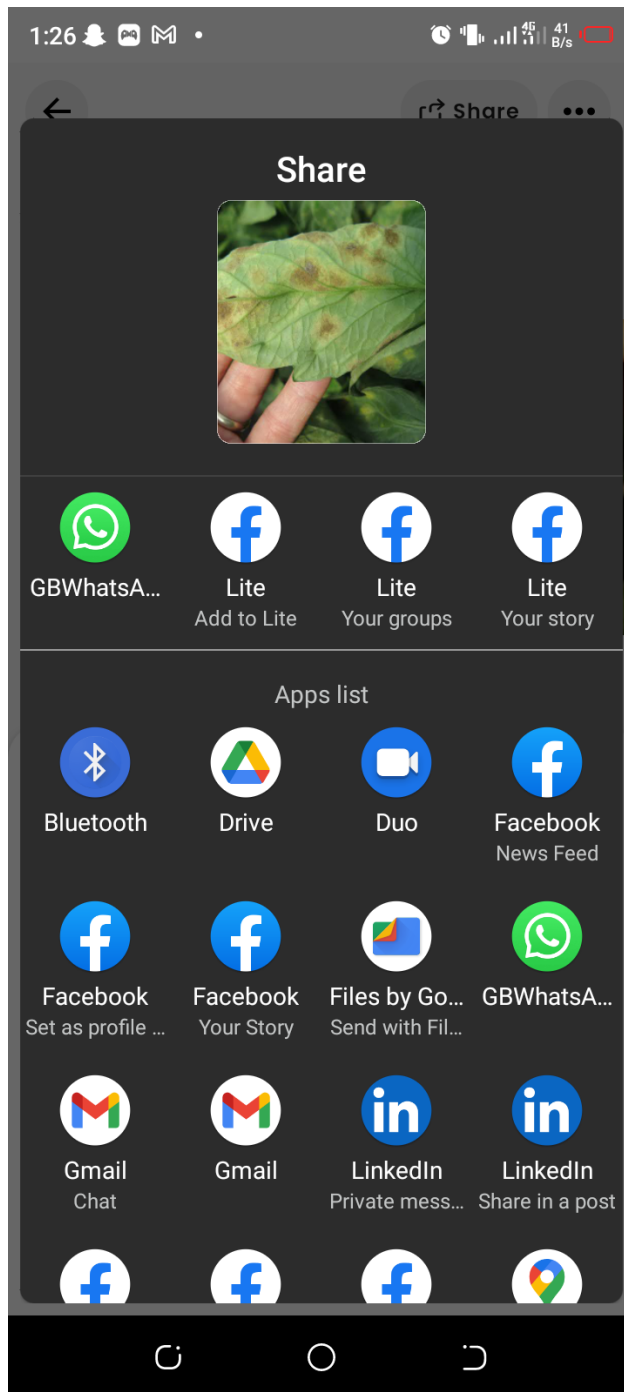


Figure 30 share results to social media

Performance Evaluation

TC Number	TC description	Pre-conditions	Test procedure	Test inputs	Expected outcome	Actual outcome	Pass/Fail	Comments
TC-001	Convert the inputted image in to array format.	The image of a diseased plant leaf should be uploaded . The image should be resized.	1. Upload the image. 2. Convert the image in to an array.	Shape of the image Image	The inputted plant leaf image converted to an array.	Image is converted to an array.	Pass	
TC-002	Choose the most accurate possibility value from each classes for the prediction .	Image should be converted and uploaded .	1. Choose the highest possibility value for all the classes.	Possibility values for every classes.	View highest possibility value.	Highest possibility value viewed .	Pass	
TC-003	Checking the final prediction result for a given image.	Get the most possible value for a given image from the possibilities for	1. Choose the highest value for the array index. 2. Map the label	Possibility values for every classes.	View final prediction for a given plant leaf	The predicted class is viewed along with the confide	Pass	

		each class array.	from the json object array. 3. Return the final result as the response.		image .	nce value.		
--	--	-------------------	--	--	---------	------------	--	--

Functional Test Evaluation

TC Number	TC description	Pre-condition s	Test procedure	Test input s	Expected outcome	Actual outcome	Pass/Fail	Comments
TC-001	Testing capturing image functionality.	Android/I OS mobile that contains a working built-in camera. User should be logged in to the system.	1. Select the 'Take picture from camera' option. 2. Take a picture from the camera. 3. Click 'confirm' button	click 'Take a picture' button Picture	Capture d photo should be appeared in the screen and ready to edited or submit.	Capture d image viewed in the mobile screen.	Pass	
TC-002	Testing selecting image from the gallery	Android/I OS mobile phone which has	1. Click 'Select from gallery' option	Click select from gallery	Selected photo should be appeared	Selected image viewed in the screen.	Pass	

	functional ity.	installed the applicatio n. User should be logged in to the system.		butto n Image	d in the screen and ready to edited or submit.			
TC-003	Testing image upload functional ity	Android/I OS mobile phone which has installed the applicatio n User should <input type="checkbox"/> Be logged in to the system.	1. Select an image from the gallery or take an image from the camera. 2. Select confirm button.	Click confir m butto n Image	A spinner should be visible until the upload is complet ed. The response should be visible.	A spinner was visible until the upload is complet ed. The response displaye d.	Pass	
TC-004	Testing result view functional ity.	Android/I OS mobile phone which has installed the applicatio n. There should be a selected	1. Click 'Confir m' button after selectin g an image.	Click confir m butto n Image	An image of the predicte d plant disease should be appeare d and disuses name, level of confiden ce and a short descripti on about	An image of the predicte d plant disease should be appeare d and disuses name, level of confiden ce and a short descripti on about	Pass	

					the disease should be visible.	the disease		
--	--	--	--	--	--	----------------	--	--

CHAPTER 4: DISCUSSION AND CONCLUSION

5.1 Overview

In this chapter, a summary of the result is given and their relationship, achievements, constraints and recommendations for further work.

A cross platform mobile application for tomato disease diagnosis using deep learning was developed.

5.2 Achievements, Limitations and recommendation

5.2.1 Achievements

The following achievements achieved after the development of the system:

1. Object detection to recognize that an image is a plant leaf
2. Object classification in order to diagnose the tomato plant disease
3. Analysis of the diagnosis made
4. User functionality achieved.

5.2.2 Limitations of the study

The completion of the system was coupled with several constraints despite its completion.

1. Implementation of the proposed solution was hard since it required lots of time for research and large datasets which were not readily available.
2. Hand-labeling data was time consuming since I took more than 4 hours to just prepare a dataset for image recognition
3. I faced challenges during training of the datasets since Google colab does not run more than 12 hours.
4. Optimization of the model through quantization techniques to reduce load times.

5.2.3 Recommendation for further work

There are recommendations on how the system can be improved in the later versions. Some of them are:

1. Using a large dataset for training and testing the algorithm
2. Scaling the app to include more diseases eg maize, potatoes etc
3. Offline disease diagnosis and analysis
4. Developing an interface for farmers, agrovets and experts to be able to communicate
5. Addition of chatbot to aid in use of the app

5.3 Conclusion

The report emphasized about the design and implementation of a deep learning based cross platform application that can identify crop diseases by analyzing plant leaves. In order to deliver a quality product the research is carried out on image processing techniques, artificial intelligence, traditional machine learning techniques and deep learning based approaches along

with the similar system case study. In order to handle the feature extraction of diseased plant leaves it is used deep learning which can handle automatic feature extraction process. Each and every selected technology for each and every task in this project has justified along with the supported evidences.

Furthermore, the developer achieved 94.99% accuracy for the deep learning model which performs well on both validation and real time data. The local dataset for the project had to be created manually by visiting local agricultural areas and capturing images of various kinds of diseased leaves. The strategy of this CNN architecture and the result of the model are based on both validation and test dataset. Throughout the representation of the confusion matrix which illustrate the accuracy of the test data the performance of the model has been decided.

Despite the challenges faced during development, it is evident that the system objectives and user requirements were met. In conclusion, this application is very helpful in helping farmers diagnose diseases before it brings a huge loss to their crops

References

<https://www.fao.org/3/i2050e/i2050e.pdf>

[\(http://www.fao.org/](http://www.fao.org/)

<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5519495/>

Kenya Agricultural Research Institute (2015), Plant Clinics In Kenya: An Innovative Way of Controlling plant Diseases. Available on the link:

<https://www.plantwise.org/Uploads/Plantwise/Kenya%20Poster%20Web.pdf>

<https://towardsdatascience.com/transfer-learning-in-tensorflow-9e4f7eae3bb4>

<http://plantix.net/press-reviews/>

<https://machinelearningmastery.com/how-to-use-transfer-learning-when-developing-convolutional-neural-network-models/>

<https://www.plantvillage.org/>

<https://thedatafrog.com/en/articles/image-recognition-transfer-learning/>

<https://machinelearningmastery.com/transfer-learning-for-deep-learning/>

APPENDIX 1

Project Structure

Task	Start Date	Duration	End date
Research, literature review and proposal documentation	4 th October 2021	5 weeks	12 th November 2021
Requirement Analysis	13 th November 2021	2 weeks	26 th November 2021
System Design	27 th November 2021	11 weeks	17 th December 2021
System implementation	3 rd January 2022	9 Weeks	10 th March 2022
Application Testing	3 rd February 2022	11 weeks	April 2022
Deployment	10 th March 2022	1 Week	20 th March 2022
Documentation	4 th October 2021	20 weeks	April 2022

Figure 31 project structure

Gantt Chart

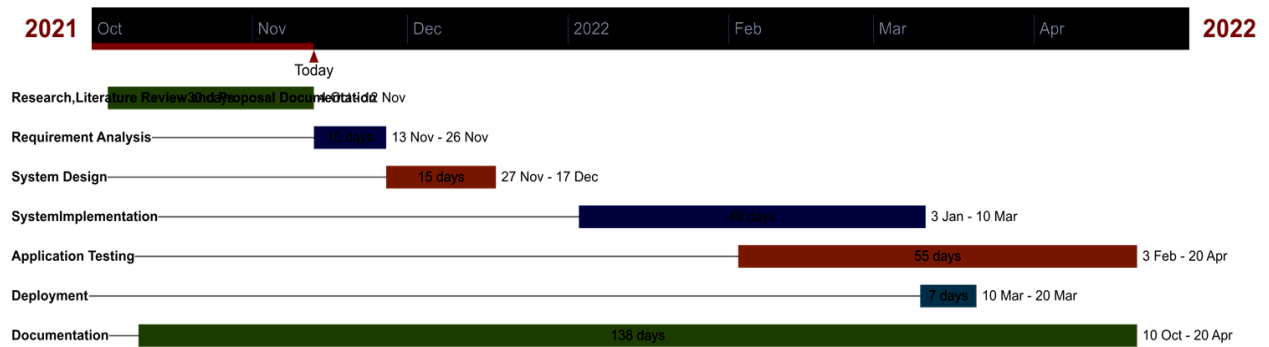


Figure 32 Gantt chart

APPENDIX 2

Sample Code

```
model.compile(optimizer='adam', loss=keras.losses.categorical_crossentropy, metrics=['accuracy'])

from keras.callbacks import ModelCheckpoint, EarlyStopping

#early stopping to monitor validation
es= EarlyStopping(monitor='val_accuracy', min_delta=0.01, patience=3, verbose=1)
#model checkpoint
mc= ModelCheckpoint(filepath="bestmodel.h5",
                    monitor='val_accuracy', min_delta=0.01,
                    patience=3, verbose=1,
                    save_best_only= True)

cb=[es,mc]

his = model.fit_generator(train,
                        steps_per_epoch=16,
                        epochs=50,
                        verbose=1,
                        callbacks=cb,
                        validation_data=val,
                        validation_steps=16)

h=his.history
h.keys()

plt.plot(h['accuracy'])
plt.plot(h['val_accuracy'], c='red')
plt.title('acc vs val-acc')
plt.show()

plt.plot(h['loss'])
plt.plot(h['val_loss'], c='red')
plt.title('acc vs val-acc')
plt.show()
```

Figure 33 sample code 1

```

#load best model
from keras.models import load_model

model = load_model("/content/bestmodel.h5")

acc = model.evaluate_generator(val)[1]

print(f"The accuracy of the model is {acc*100} %")

ref=dict(zip(list(train.class_indices.values()),list(train.class_indices.keys()))))

def prediction(path):
    img = load_img(path, target_size=(256,256))
    i= img_to_array(img)
    im = preprocess_input(i)
    img = np.expand.dimesnion(im,axis=0)
    pred= np.argmax(model.predict(img)) # take the largest value
    print(f"the image belongs to {ref[pred]}")
    print(img.shape)

path=""
prediction(path)

```

Figure 34sample code 2

Dart script for running the disease detection

```
main.dart X
1 import 'package:flutter/material.dart';
2 import 'package:hive_flutter/hive_flutter.dart';
3 import 'package:plant_disease_detector/services/disease_provider.dart';
4 import 'package:plant_disease_detector/src/home_page/home.dart';
5 import 'package:plant_disease_detector/src/home_page/models/disease_model.dart';
6 import 'package:plant_disease_detector/src/suggestions_page/suggestions.dart';
7 import 'package:provider/provider.dart';
8
9 Future main() async {
10   WidgetsFlutterBinding.ensureInitialized();
11   await Hive.initFlutter();
12   Hive.registerAdapter(DiseaseAdapter());
13
14   await Hive.openBox<Disease>('plant_diseases');
15
16   runApp(const MyApp());
17 }
18
19 class MyApp extends StatelessWidget {
20   const MyApp({Key? key}) : super(key: key);
21
22   @override
23   Widget build(BuildContext context) {
24     return ChangeNotifierProvider<DiseaseService>(
25       create: (context) => DiseaseService(),
26       child: MaterialApp(
27         debugShowCheckedModeBanner: false,
28         title: 'Detect diseases',
```

Figure 35main.dart

Dart script for homepage, allow image selection from camera and gallery

```
35 Widget build(BuildContext context) {
36   // Get disease from provider
37   final _diseaseService = Provider.of<DiseaseService>(context);
38
39   // Hive service
40   HiveService _hiveService = HiveService();
41
42   // Data
43   Size size = MediaQuery.of(context).size;
44   final Classifier classifier = Classifier();
45   late Disease _disease;
46
47   return Scaffold(
48     floatingActionButtonLocation: FloatingActionButtonLocation.centerFloat,
49     floatingActionButton: SpeedDial(
50       icon: Icons.camera_alt,
51       spacing: 10,
52       children: [
53         SpeedDialChild(
54           child: const FaIcon(
55             FontAwesomeIcons.file,
56             color: kWhite,
57           ), // FaIcon
58           label: "Choose image",
59           backgroundColor: kMain,
60           onTap: () async {
61             late double _confidence;
62             await classifier.getDisease(ImageSource.gallery).then((value) {
63               _disease = Disease(
```

Figure 36 homepage

Script to display results, image data, confidence and description

```

main.dart X home.dart X
92     label: "Take photo",
93     backgroundColor: kMain,
94     onTap: () async {
95         late double _confidence;
96
97         await classifier.getDisease(ImageSource.camera).then((value) {
98             _disease = Disease(
99                 name: value![0]["label"],
100                 imagePath: classifier.imageFile.path);
101
102             _confidence = value[0]['confidence'];
103         });
104
105         // Check confidence
106         if (_confidence > 0.8) {
107             // Set disease for Disease Service
108             _diseaseService.setDiseaseValue(_disease);
109
110             // Save disease
111             _hiveService.addDisease(_disease);
112
113             Navigator.restorablePushNamed(
114                 context,
115                 Suggestions.routeName,
116             );
117         } else {
118             // Display unsure message
119
120         }

```

Figure 37home

```

    };

    getDiseasePredection = async (image) => {
      const data = new FormData();
      data.append('file', image);
      try {
        this.setState({
          isLoading: true
        });

        const response = await axios.post(
          `http://192.168.1.4:5000/api/predict`,
          data
        );

        if (response.data) {
          this.setState({
            isLoading: false
          });

          const { disease, confidence, id } = response.data;

          let _disease = disease.split('__').join(' ');

          this.props.navigation.navigate('PredictionScreen', {
            _disease,
            confidence,
            id
          });
        }
      } catch (err) {
        console.log(err.message);
      }
    };

    onShowModal = () => {
      this.setState({
        isVisible: true
      });
    };

    render() {

```

Figure 38 call flask api

```

    image: require('../assets/data/Potato__Late_blight.png'),
  },
  {
    id: 6,
    name: 'Tomato Target Spot',
    image: require('../assets/data/Tomato__Target_Spot.png'),
  },
  {
    id: 7,
    name: 'Tomato Mosaic Virus',
    image: require('../assets/data/Tomato__Tomato_mosaic_virus.png'),
  },
  {
    id: 8,
    name: 'Tomato YellowLeaf Curl Virus',
    image: require('../assets/data/Tomato__Tomato_YellowLeaf__Curl_Virus.png'),
  },
  {
    id: 9,
    name: 'Tomato Bacterial Spot',
    image: require('../assets/data/Tomato_Bacterial_spot.png'),
  },
  {
    id: 10,
    name: 'Tomato Early Blight',
    image: require('../assets/data/Tomato_Early_blight.png'),
  },
  {
    id: 11,
    name: 'Tomato Healthy',
    image: require('../assets/data/Tomato_healthy.png'),
  },
  {
    id: 12,
    name: 'Tomato Late Blight',
    image: require('../assets/data/Tomato_Late_blight.png'),
  },
  {
    id: 13,
    name: 'Tomato Leaf Mold',
    image: require('../assets/data/Tomato_Leaf_Mold.png'),
  },
},

```

Figure 39 labels.txt