



*Arab American University*

*Faculty of Engineering and Information Technology*

*Computer Systems Engineering*

## **SENIOR PROJECT (II)**

**“Plantie” Smart Image Processing for Plant Diseases Detection and  
Farmer Assistant**

Academic Year:

2023/2024

### **Students Statement**

We the undersigned students, certify and confirm that the contents and work submitted in this project report are entirely our own and have not been copied from any other source. Any material that has been used from other sources has been properly cited and acknowledged in the report.

We are fully aware that any copying or improper citation of references/sources used in this report will be considered plagiarism, which is a clear violation of the Code of Ethics of the Arab American University.

Ahmad Ilawa (201911990)

Ahmad Qasem (201911285)

Mohammad Nassar (201911349)

Computer System Engineering Dept.

Submitted in partial fulfillment of the requirements of B.Sc. Degree in Computer  
System Engineering

## Supervisor Certification

This is to certify that the work presented in this senior year project manuscript was carried out under my supervision, which is entitled:

### Plantie App

"Smart Image Processing for Plant Diseases Detection and Farmer Assistant"

Ahmad Ilawa (201911239)

Ahmad Qasem (201911285)

Mohammad Nassar (201911349)

I hereby that the students have successfully finished their senior year project and by submitting this report they have fulfilled in partial the requirements of B.Sc. Degree in Computer Systems Engineering.

I also, hereby that I have read, reviewed, and corrected the technical content of this report and I believe that it is adequate in scope, quality, and content and it is in alignment with the ABET requirements and the department guidelines.

Prof. Mohammed Awad,

## ACKNOWLEDGMENT

Praise be to God, Lord of the worlds, and prayers and peace be upon the most honorable of the prophets and messengers, our master Muhammad, his family, and companions, and those who followed them in goodness until the Day of Judgment.

First and foremost, we thank God for His blessings on us, and on our behalf, we succeeded in this endeavor. We also pray for the knowledge we gain to be useful to the rest of the world, especially to Palestine.

Then we thank those kind people who extended a helping hand to me during this period, headed by Prof. Mohammed Awad, who spared no effort in helping me.

We want to thank our families for the continued support and the encouragement they gave us from start to finish moment.

We thank the dear doctors in the Department of Computer Engineering, who get the credit for reaching this stage with everything they've given us all these years, we will be forever grateful.

To all those and others who have supported us through thick and thin, we offer a big "thank you" straight from our hearts, we bring to you this report and project.

## ABSTRACT

*In the realm of agriculture, where the importance of sustainable practices looms large, the timely detection of diseases in crop production emerges as a critical factor for increasing productivity. However, Due to the scarcity of technologies used in crops, the diagnosis of diseases and pests is largely supported by human inspection, generating errors caused by the subjectivity of individuals. Even if disease is detected, finding the optimal treatment among the thousands of fertilizers and agricultural medicines is a challenging task. Our project successfully developed the Pantie App, a full-of-feature mobile application for caring for plants that use the artificial intelligence subfield of computer vision. Plantie's main feature is a plant disease detection system that analyses photos taken with mobile devices using MobileNetV2 architecture. This system achieves an impressive 98.7% overall accuracy when comparing the photos to a collection of known plant diseases and recommends useful treatments. Plantie provides a range of functions to assist farmers and plant fans in addition to detecting diseases. The app has a community platform where users can interact and share information, real-time weather updates with specific plant care recommendations, a fertilizer calculator based on cultivation area and NPK values, comprehensive plant information, and a feature that lets you find nearby plant stores using Google Maps. Plantie's integration of these features makes crops more productive, lowers losses, and gives farmers access to key tools and information right on their mobile devices. The management of plant diseases and overall agricultural results can be enhanced with this creative method, which is adaptable and scalable.*

Keywords: *Smart Plant Disease Detection, Pests, Farmer, Plant, Smartphone Camera, CNN, Mobile Application.*

# Table of Contents

LIST OF FIGURES .....	x
List of Tables .....	xii
LIST OF SYMBOLS AND ABBREVIATIONS.....	xiii
1 CHAPTER 1: INTRODUCTION .....	1
1.1 Problem Statement and Purpose.....	2
1.2 Project and Design Objectives .....	3
1.3 Intended Outcomes and Deliverables.....	3
1.4 Summary of Report Structure .....	4
2 CHAPTER 2: BACKGROUND.....	5
2.1 Overview.....	5
2.2 Related works.....	6
2.2.1 Plant village Nuru .....	6
2.2.2 Plantix .....	6
2.2.3 Potato Disease Classification.....	7
2.2.4 Intelligent System for Cucumber Leaf Disease Diagnosis.....	8
2.3 Comparison .....	9
3 CHAPTER 3: METHOD AND MATERIALS.....	10
3.1 System Algorithms.....	10
3.1.1 Plant Diagnosis System.....	11
3.1.2 Fertilizer Calculator .....	18
3.1.3 Find the Nearest Plant Shop .....	20
3.1.4 Weather System .....	21
3.1.5 Community System.....	23
3.2 System design.....	24

3.2.1	<b>Architectural Design.....</b>	24
3.2.2	<b>Database Design &amp; UML Diagram .....</b>	25
3.2.3	<b>User Scenarios .....</b>	26
3.2.4	<b>Use Case Diagram.....</b>	29
3.2.5	<b>Sequence Diagram .....</b>	30
3.2.6	<b>ER Diagram.....</b>	34
3.3	<b>Software and Tools.....</b>	34
3.3.1	<b>GitHub .....</b>	34
3.3.2	<b>Google Colab .....</b>	35
3.3.3	<b>Draw.io.....</b>	35
3.3.4	<b>Figma.....</b>	35
3.3.5	<b>Trello .....</b>	36
3.3.6	<b>Dropbox .....</b>	36
3.4	<b>Design Specifications, Standards, and Constraints .....</b>	36
3.4.1	<b>Functional Requirements .....</b>	36
3.4.2	<b>Non-Functional Requirements.....</b>	37
3.5	<b>Programming languages and frameworks .....</b>	40
3.5.1	<b>Flutter.....</b>	40
3.5.2	<b>Python .....</b>	40
3.5.3	<b>Firebase.....</b>	41
3.6	<b>Design Alternatives .....</b>	41
3.6.1	<b>Offline Detection Application .....</b>	41
3.6.2	<b>Online Detection Alternative .....</b>	42
3.6.3	<b>Hybrid Approach Alternative.....</b>	42
3.7	<b>System Analysis and Optimization .....</b>	42

3.7.1	<b>Plant Disease Detection.....</b>	43
3.7.2	<b>Community .....</b>	45
3.7.3	<b>Real-Time Weather.....</b>	45
3.7.4	<b>Nearby Plant Stores .....</b>	45
3.7.5	<b>Fertilizer Calculator .....</b>	46
3.8	<b>Simulation and Experimental Test .....</b>	46
3.8.1	<b>Detection View: .....</b>	47
3.8.2	<b>Fertilizer Calculator: .....</b>	55
3.8.3	<b>Community:.....</b>	55
3.8.4	<b>Nearby Plant Stores:.....</b>	56
4	<b>CHAPTER 4: RESULT AND DISCUSSIONS.....</b>	58
4.1	<b>Result.....</b>	58
4.1.1	<b>Authentication Views.....</b>	58
4.1.2	<b>Plants/Home View.....</b>	59
4.1.3	<b>Disease Detection Views .....</b>	59
4.1.4	<b>Community Views.....</b>	60
4.2	<b>Discussions.....</b>	61
4.2.1	<b>Plant Disease Detection: .....</b>	61
4.2.2	<b>Community:.....</b>	61
4.2.3	<b>Real-Time Weather Updates: .....</b>	62
4.2.4	<b>Nearby Plant Stores:.....</b>	62
4.2.5	<b>Fertilizer Calculator: .....</b>	62
5	<b>CHAPTER 5: PROJECT MANAGEMENT .....</b>	63
5.1	<b>Tasks, Schedule, and Milestones.....</b>	63
5.2	<b>Resources and Cost Management .....</b>	66

5.3	Lessons Learned.....	67
6	CHAPTER 6: IMPACT OF ENGINEERING.....	68
6.1	Economical, Societal and Global .....	68
6.2	Environmental .....	69
6.3	Other Issues .....	69
7	CHAPTER 7: CONCLUSION AND RECOMMENDATION .....	71
7.1	Summary of Achievements of the Project Objectives .....	71
7.2	New Skills and Experiences Learn.....	72
7.3	Recommendations for Future Work.....	72
	References.....	74

## LIST OF FIGURES

Figure 1-1 The number of losses in the agriculture field cause plant diseases.....	2
Figure 2-1 The production increase and the reduction in using fertilizer.....	5
Figure 2-2 Plant Village Nuru.....	6
Figure 2-3 Plantix - Related works.....	7
Figure 2-4 Potato Disease Detection system flow.....	8
Figure 3-1 Algorithm Architecture .....	10
Figure 3-2 Plant Disease Detection Algorithm Architecture .....	11
Figure 3-3 Samples from the pre-collected dataset.....	12
Figure 3-4 3 methods solving the imbalanced data. ....	13
Figure 3-5 Applying image preprocessing.....	14
Figure 3-6 ANN Algorithm layers.....	15
Figure 3-7 CNN Layers .....	15
Figure 3-8 Generating feature map for a single filter. ....	17
Figure 3-9 Max and Average Pooling.....	17
Figure 3-10 LeNet architecture .....	18
Figure 3-11 Weather conditions.....	22
Figure 3-12 Community System.....	23
Figure 3-13 System Architecture .....	24
Figure 3-14 Proposed system database UML Diagram .....	26
Figure 3-15 Use case for the Plantie app .....	29
Figure 3-16 Login/Register Sequence Diagram Post on the community/Comment on the post..	30
Figure 3-17 Post on community/Comment on the post Sequence Diagram .....	31
Figure 3-18 List View Sequence Diagram.....	32
Figure 3-19 Detection of the plant disease using the ML Model Sequence Diagram .....	33
Figure 3-20 ER Diagram.....	34
Figure 3-21 Distribution of images in each class.....	44
Figure 3-22 The Plantie app's main view, showcases weather updates, plant care advice, and detailed plant information. ....	47

Figure 3-23 The Plantie app detecting a disease from an uploaded plant image.....	54
Figure 3-24 The Plantie app's fertilizer calculator in action, recommending the ideal fertilizer dosage based on NPK values and .....	55
Figure 3-25 The Plantie app's community feature, showcasing user interactions and shared plant care tips. ....	56
Figure 3-26 The Plantie app's fertilizer calculator in action, recommending the ideal fertilizer dosage based on NPK values and cultivation area for a selected plant type .....	57
Figure 4-1 Authentication Views.....	58
Figure 4-2 Plants/Home View .....	59
Figure 4-3 Disease Detection View .....	60
Figure 4-4 Community View .....	60
Figure 5-1 Project Schedule with Gantt-Chart 1.....	63
Figure 5-2 Project Schedule with Gantt-Chart 2.....	64
Figure 5-3 Project Schedule with Gantt-Chart 3.....	64
Figure 5-4 Project Schedule with Gantt-Chart 4.....	65
Figure 5-5 Project Schedule with Gantt-Chart 5.....	65
Figure 5-6 Project Schedule with Gantt-Chart 6.....	66

## LIST OF TABLES

Table 1 System Comparison .....	9
Table 2 System Requirement Table.....	36
Table 3 Models Classification Results.....	44
Table 4 Classification Report of MobileNetV2 .....	51
Table 5 Resources and Cost Management .....	66

## LIST OF SYMBOLS AND ABBREVIATIONS

This is where you should put all the symbols used and abbreviations (must be sorted). It is stored in a table format same as the table of contents. Use inserts a row to add more entries.... etc.

LIST OF SYMBOLS	
LIST OF ABBREVIATIONS	
ABET	Accreditation Board for Engineering and Technology
ReLU	rectified linear unit.
CNN	Convolutional Neural Networks
ANN	Artificial Neural Network
NaCRRI	National Crops Resources Research Institute
VCS	Version Control System
AI	Artificial Intelligence
AR	Augmented Reality
MVC	Model-View-Controller
Rest	Representational State Transfer
API	Application Programming Interface
N-P-K	Nitrogen (N), Phosphorus( $P_2O_5$ ), and Potassium ( $K_2O$ )
$Ha^{-1}$	Hectare

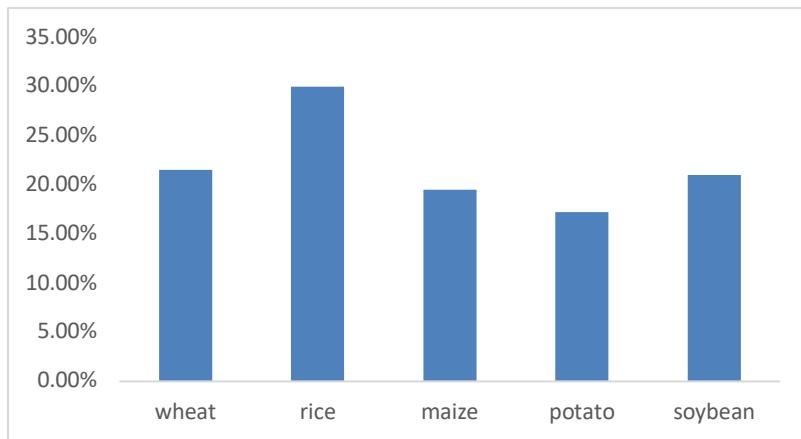
# 1 CHAPTER 1: INTRODUCTION

Throughout history, humans have been developing tools and techniques to enhance agricultural productivity. From simple farm tools and organic fertilizers to greenhouses, tractors, chemical fertilizers, and other inventions made by humans. The most important part of the process of increasing agricultural productivity is maintaining the health of crops and providing farmers with the best treatment. The process of maintaining plant health includes appropriate fertilizers and medications. The detection of plant disease and its treatment is not an easy process, it needs an experienced agronomist who knows all aspects of plants.

With the help of machine learning and deep learning, the agricultural industry has become more advanced. Machine learning and deep learning have become one of the most important subfields of artificial intelligence through which systems are developed that do not need humans to perform their assigned tasks. Not to mention the speed and accuracy of the results possessed by these modern technologies, the results may not be as accurate if the person performed them himself. These technologies came trying to replace human work to reduce effort, budget, and time and increase productivity.

So, why deep learning in plant disease detection? Because the traditional way of predicting plant diseases requires full knowledge of plants. Human inspection can make mistakes in predicting many diseases and providing the best treatment that is needed. On the other hand, the process of detection and prediction of the disease is much faster and more accurate when using deep learning and computer vision. All you need is a camera to predict the disease and provide treatment in less than a minute. In dealing with this issue, the system will use a sub-field of Artificial Neural Network (ANN) called Convolutional Neural Network (CNN) which is particularly well-suited for image recognition and processing tasks. The process includes image acquisition, image preprocessing, image segmentation, feature extraction, feature selection, and subsequent classification of disease. Anyway, plant disease is a huge impact on humans around the world, Our yield loss (range) As shown in figure 1-1 estimates at a global level and per hotspot for wheat (21.5% (10.1–28.1%)), rice (30.0% (24.6–40.9%)), maize (22.5% (19.5–41.1%)), potato (17.2% (8.1–21.0%)) and soybean (21.4% (11.0–32.4%)) suggest that the highest losses are associated

with food-deficit regions with fast-growing populations, and frequently with emerging or reemerging pests and diseases [1].



*Figure 1-1 The number of losses in the agriculture field cause plant diseases.*

### 1.1 Problem Statement and Purpose

The process of plant diagnosis through observation of the symptoms on plant leaves is full of complexity. Experienced agronomists and plant pathologists struggle to diagnose specific diseases, and that leads to making mistakes in predictions of the plant treatment process. Without the use of advanced diagnostic tools and automated computational systems for the detection and diagnosis of plant diseases, errors are more likely to occur. These technologies play an important role for agronomists and pathologists to assist them in the process of detecting plant pests and diseases.

Traditional methods often rely on human experts like agronomists and pathologists to diagnose plant pests and diseases, and that could be a long-time process and can lead to errors due to variations in human judgment. The time an expert spends predicting the disease could lead to an increase in the spread of diseases. Moreover, when dealing with a diverse range of plant species and potential issues, manual inspection may result in inaccuracies. In Addition, the process of manual diagnosis of the plant needs a laboratory analysis and this process may require substantial time, manpower, and equipment.

Building a mobile application that detects plant diseases could be a valuable tool for farmers. It helps the farmers to reduce the expected time of diagnosis and saves them money. The application employs image processing to extract visual features, including color, texture, and patterns, from

plant images that are taken from a phone camera or a gallery. Then a pre-trained CNN model is used to analyze these features and classify plant diseases.

## **1 . 2 P r o j e c t a n d D e s i g n O b j e c t i v e s**

Our project successfully developed Plantie, a comprehensive mobile application aimed to assist farmers in increasing crop yield by deploying a deep-learning model to detect plant diseases. Using convolutional neural networks (CNNs) with MobileNetV2 architecture, the model analyses plant photos taken with mobile devices. The model achieves an amazing 98.7% overall accuracy.

Plantie provides a range of tools and techniques that assist in the farming process in addition to disease detection:

- Build/design a user-friendly mobile application.
- Support the Mobile app with a design scalable and extendable system.
- Support the Mobile app with suggestions based on Weather conditions.
- Support the Mobile app with a community System for mutual information between users.
- Support the Mobile app with the Fertilizer Calculation System.
- Support the Mobile app with information to Find the nearest plant shop to the user.

## **1 . 3 I n t e n d e d O u t c o m e s a n d D e l i v e r a b l e s**

The desired outcome from this project is a mobile application that provides farmers the ability to detect plant disease, which saves time, effort, and money. In addition, the application will provide the farmers the common treatment for specific plant diseases. The application is an assistant for the farmers which reduces the need for experts there's no need to waste much time in trying to predict the desired disease within the possibility of errors occurring with associated documentation that describes the project requirements and project features in advance.

## **1.4 Summary of Report Structure**

The senior project design document is divided into several chapters to ensure that technical experts can read it easily. The following describes each chapter's content:

In Chapter 1, you will find an overview of intended outcomes and deliverables, as well as an introduction to the project, the problem statement, and the purpose of the project.

Chapter 2 provides background information, which includes a project overview, relevant statistics, and previous research and projects that have explored similar ideas.

In Chapter 3, the document outlines the main components of the project and then provides an extensive explanation of the system design, which includes design specifications, related standards, and any constraints. The chapter focuses on design alternatives, system analysis, and optimization, and ends with simulation or experimental tests.

Chapter 4 presents the project's findings and conclusions.

Chapter 5 concentrates on project management, which encompasses tasks, schedules, milestones, resource management, and cost management. The project lessons learned are included in this chapter.

In Chapter 6, the impact of the engineering solution on the environment, economy, society, and other relevant issues is discussed.

The document ends in Chapter 7 with the team's final thoughts on the project, the skills they acquired during the process, and recommendations for future work.

## 2 CHAPTER 2: BACKGROUND

### 2.1 Overview

A plant disease detection system that utilizes deep learning algorithms allows farmers to use a mobile application to handle the recognition process of plant diseases through infected plant leaf images. After the system recognizes the specific disease, it provides the optimal treatment that the plant needs. In Plantie, the system will detect many diseases for different plants. The common plants that the system must handle are the tomato, grape, olive, potato, and cucumber. The flow of this process follows some steps. The first step is image acquisition, image acquisition is the process of capturing the image through the smartphone camera. After we capture the image and convert it to the digital version, the image must go through the preprocessing step. Preprocessing means making some changes in the image to prepare them for analysis. This step may include Noise reduction, Color space conversion, Normalization, and Segmentation. Finally, the image is ready to be inserted in the pre-trained CNN model. Figure 2-1 show that tools such as Plantix and Farm Beats, farmers can reduce herbicide and fertilizer consumption by 25% to 35% as well as increase yields by up to 4% [2].

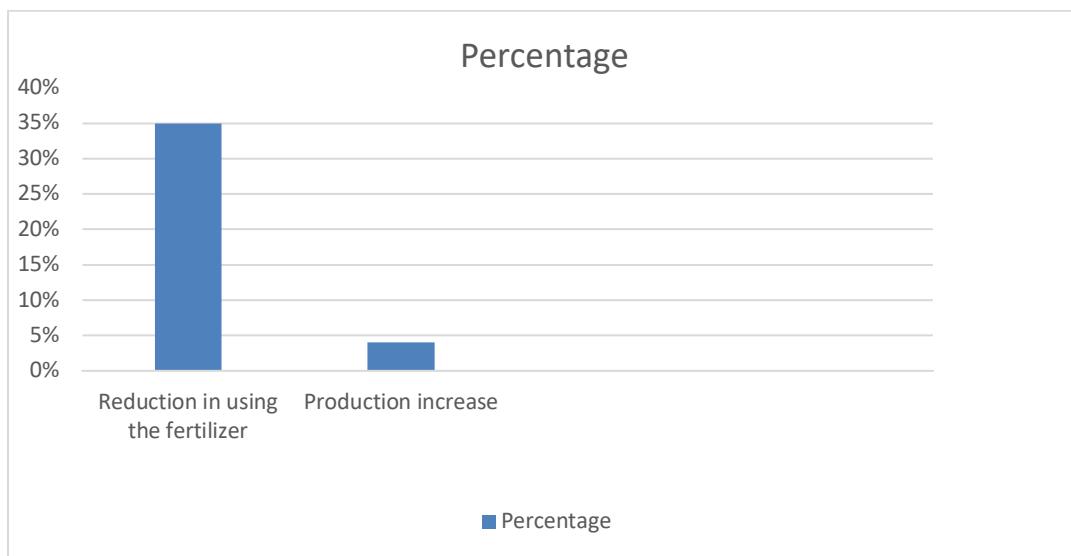


Figure 2-1 The production increase and the reduction in using fertilizer

## 2.2 Related works

### 2.2.1 Plant village Nuru

PlantVillage Nuru is a publicly supported, and publicly developed application that uses a digital assistant to help farmers diagnose crop disease in the field, without an internet connection [3].

Plant Village Nuru is a mobile app that uses CNN to detect and diagnose plant diseases. Plant village Nuru provides many features such as diagnosing plant diseases, a community for farmers, and a chatbot called Cassava AI. The application uses CNN architecture called Inception-v3 which is developed by Google. The Plant Village Nuru uses its dataset called PlantVillage. PlantVillage dataset is the most common dataset for plant disease recognition which provides many diseases for different plants.



Figure 2-2 Plant Village Nuru

### 2.2.2 Plantix

Plantix is a mobile application that helps farmers to take control of their crop health. The application provides features such as plant disease and pest identification, crop advisory, community platform, and offline functionality. Plantix helps farmers diagnose and treat crop problems, improve productivity, and provide farming knowledge. Achieve your farming goals and improve your agricultural experience with Plantix [4]. The application allows you to choose a specific plant to provide you with the information that you need within the possibility of predicting

the disease from the image. Unfortunately, there's not enough information about the architecture of this system. Plantix is one of the most common mobile applications in plant disease recognition. The app shows the weather status and then provides if the current day and the next day are good for applying pesticides or weeding. In addition, the application provides fertilizer calculation and cultivation tips for a specific plant.

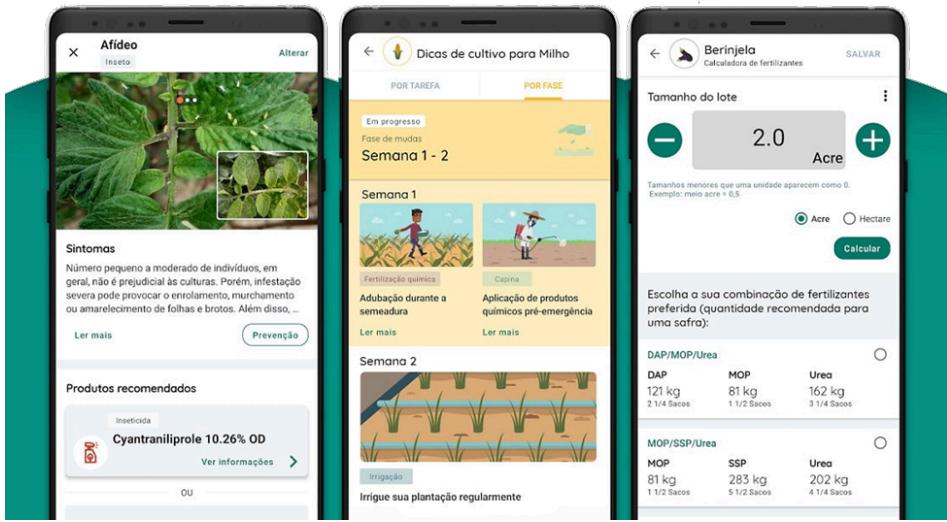
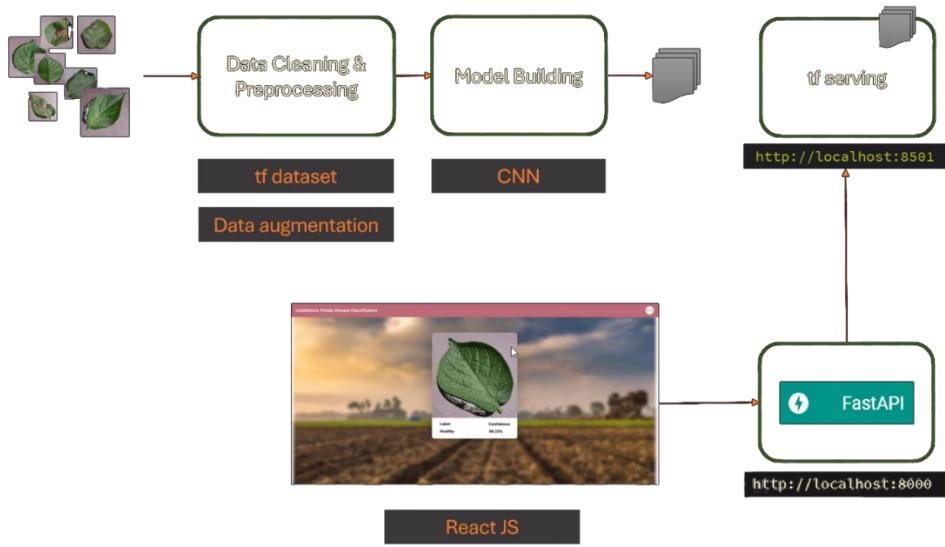


Figure 2-3 Plantix - Related works.

### 2.2.3 Potato Disease Classification

It's a potato disease detection system that utilizes the CNN model inside a backend within a Website and mobile connection. The system detects and recognizes a potato disease in a fast API server to handle the prediction process. The system uses Custom CNN architecture which uses 5 Convolution layers with several filters: 32, 64, 64, 64, 64 in row, Max Pool layers which are applied after each convolutional layer with a pool size of 2x2 [5]. The system CNN model applies the preprocessing inside the model itself as layers that make resizing, rescaling, and augmentation. The result of the architecture is 1 layer for resizing to 220x220 shape, 1 layer for rescaling, 2 layers for augmentation which apply the rotation and flipping, and a series of convolutions layers and Max pool layers. Figure 2-4 shows the system flow of Potato Disease System:



*Figure 2-4 Potato Disease Detection system flow.*

#### 2.2.4 Intelligent System for Cucumber Leaf Disease Diagnosis

In a paper on December 2, 2022, Saman M. Omer, Kayhan Z. Ghafoor, and Shavan K. Askar discuss the plant disease detection for the cucumber plant. The paper discusses an architecture built by the authors to show the comparison between the common architecture and the proposed architecture, and these architectures are Inception-V3 architecture, AlexNet architecture, and ResNet-50 architecture. The paper shows that the proposed model for detecting 5 cucumber diseases for 200 epochs, was better than the rest of the architecture. The recognition accuracy was 98.19%, 97.77%, 97.53, 96.69%, and 96.14%, respectively on 1-CNN (proposed model), Inception-V3, and ResNet-50 [6]. In the context of the paper, it shows that the Data Argumentation to make balance in the dataset is an efficient approach for cucumber disease detection, especially when the data is unbalanced. In this context, we noticed that all related works came together to detect the disease first, and there were differences in how they implemented this process. The Plantix can predict a lot of different plants for different diseases, and as shown in his site it's the top free app for crop diagnosis. The key difference between them is the side features like the community, fertilizer calculator, suggestions based on weather, and the variety of plants that the system can predict. Our project meets all these features plus the finding nearest plant shop.

## 2.3 Comparison

*Table 1 System Comparison*

	<i>Plant Village Nuru</i>	<i>Plantix</i>	<i>Potato Disease Detection</i>	<i>Cucumber Disease Detection</i>	<i>Proposed Project</i>
<i>The number of plants Can Detect</i>	Many	Many	One plant	One plant	5 or more
<i>Community</i>	Not support	Support	Not support	Not support	Support
<i>Fertilizer Calculator</i>	Not support	Support	Not support	Not support	Support
<i>Find the Nearest Plant Shop</i>	Not support	Not support	Not support	Not support	Support
<i>Disease Treatment</i>	Support	Support	Not support	Not support	Support
<i>Suggestions based on weather conditions</i>	Not support	Support	Not support	Not support	Support
<i>Target</i>	Africa	Indian	For education purpose	For education purpose	Palestine

### 3 CHAPTER 3: METHOD AND MATERIALS

In the previous chapters, it has been talking about what the purpose and objectives of the project and how it will help the farmers to keep the crops healthy. In this chapter, we will discover the system architecture design, the system modeling diagram, the specification, and the functional and non-functional requirements of the proposed system within the standards and constraints that will be followed during the process of developing the system.

#### 3.1 System Algorithms

The system design and algorithms are an important process to provide a clear requirement specification for our development process. The maintainability of the system will be easier if we well understand the system design and algorithms. The system design and what algorithms it used, give us a clear knowledge of system flow. Moreover, system design provides the audience with a roadmap for understanding how the system works. This translates to improved maintainability and a robust foundation for future system enhancements. Figure 3-1 shows the architectural design of the system.

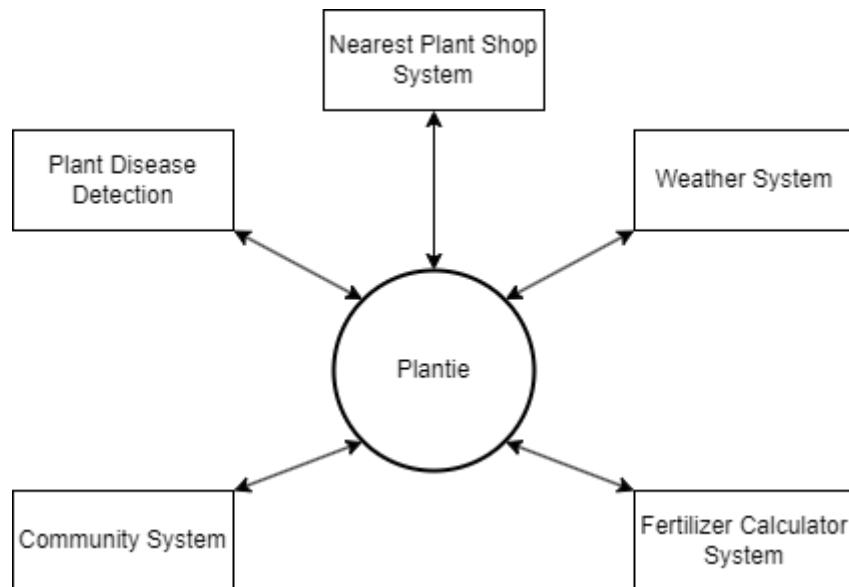


Figure 3-1 Algorithm Architecture

### 3.1.1 Plant Diagnosis System

The process of plant disease detection includes the dataset collection for the training model process and the evaluation of the model accuracy, image acquisition to convert the plant leaf into digital representation, and image preprocessing to apply some changes to enhance the image quality for our purpose. After these steps, now the image is ready to be inserted into the pre-train CNN model which is trained with the previously collected dataset. The model will classify and make predictions of plant disease. Figure 3-2 shows the plant disease recognition flow:

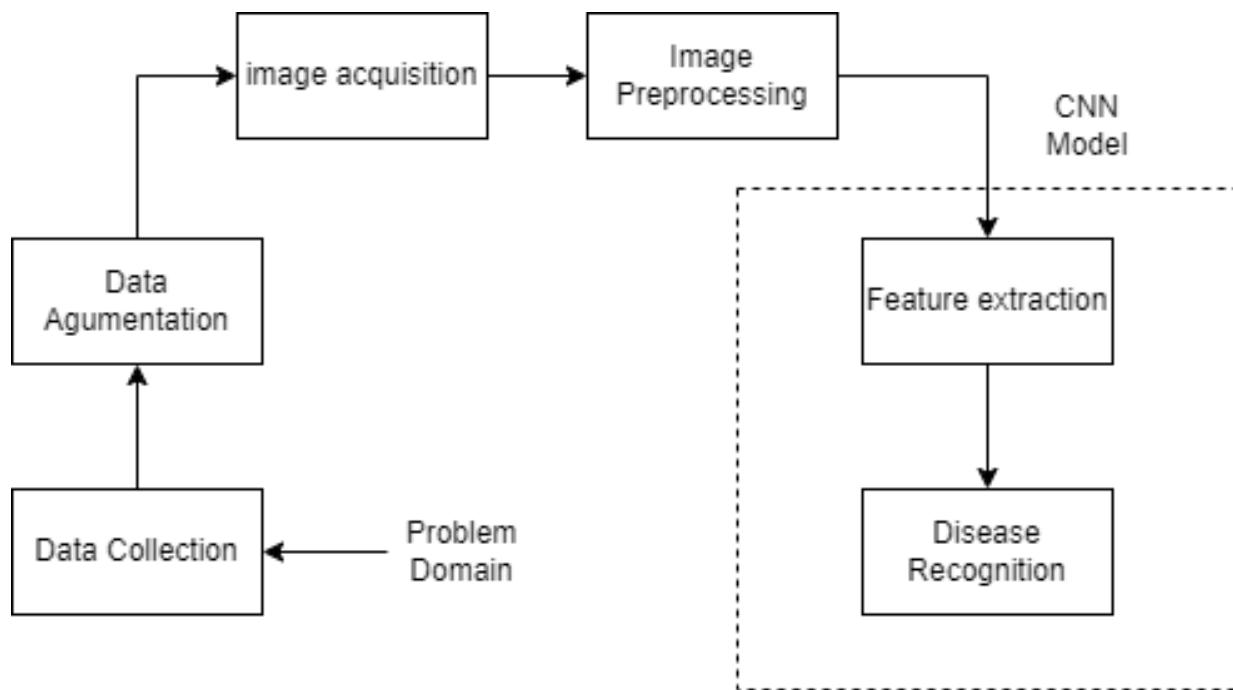


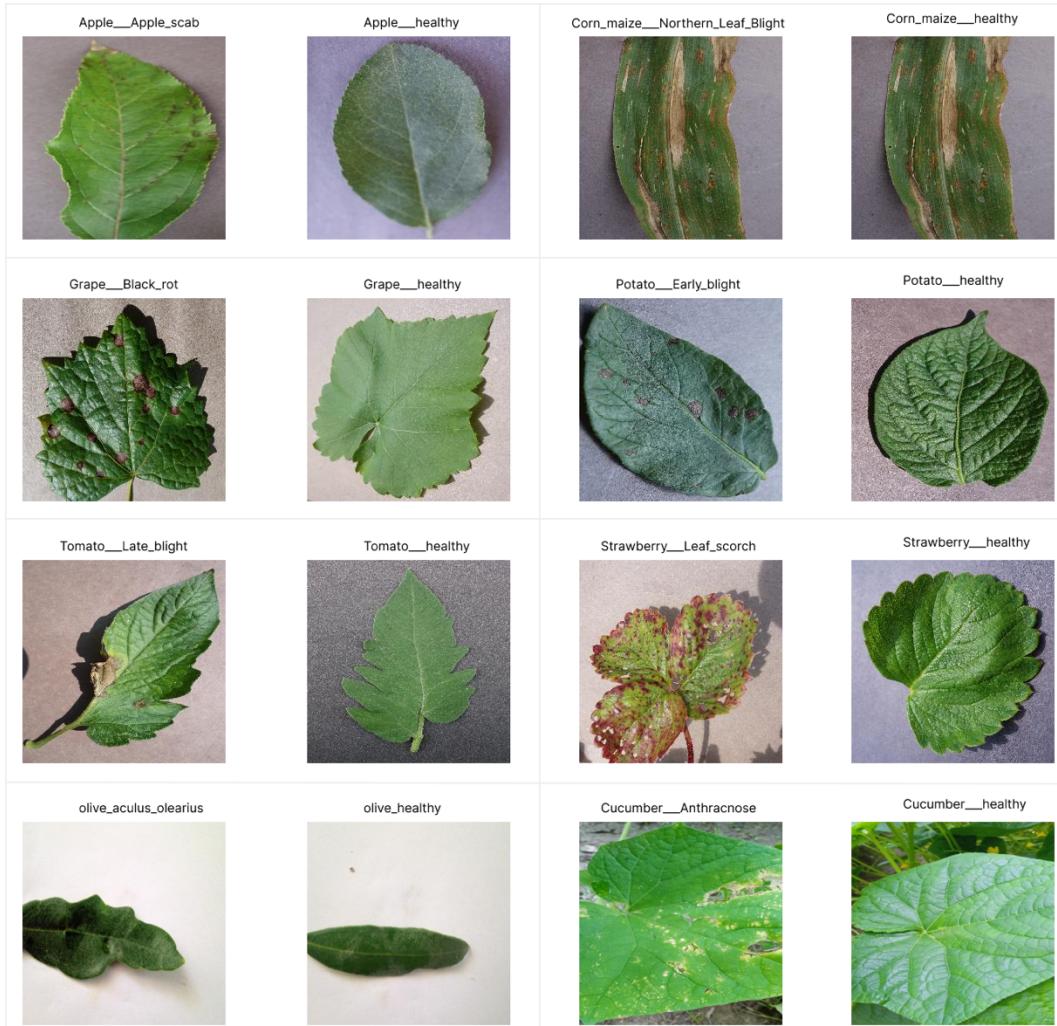
Figure 3-2 Plant Disease Detection Algorithm Architecture

#### 3.1.1.1 Data Collection

As we mentioned before the dataset is an important requirement in the training process. The quality, diversity, and quantity of data affect the accuracy of prediction and the training process time. Also, part of the dataset is used in the testing and evaluation process. We need to test and evaluate the process to determine the effectiveness and of our trained model.

The proposed project uses different datasets and combines them. Dataset-1. PlantVillage: PlantVillage is the most common dataset in the realm of plant disease detection. The PlantVillage

contains many types of plants and diseases. The plants include apples, corn, potato, tomato, pepper, and more plants with different diseases. Dataset-2. olive dataset is a dataset in GitHub collected from Denizli city of Turkey during spring and summer [7]. Dataset-3. ibeans dataset is a dataset in GitHub collected by the Makerere AI lab in collaboration with the National Crops Resources Research Institute (NaCRRI), the national body in charge of research in agriculture in Uganda [8]. Figure 3-3 shows some samples from the pre-collected dataset:

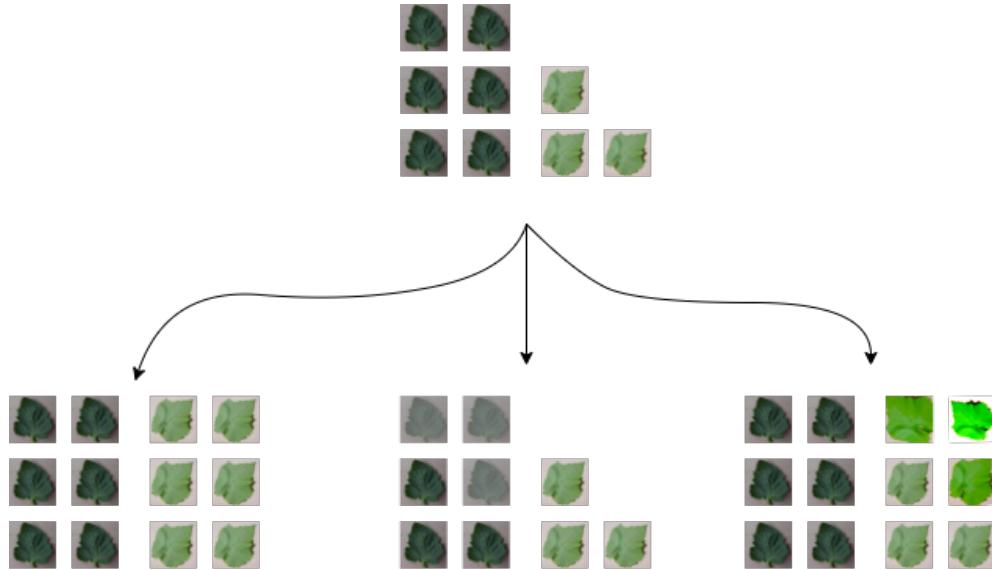


*Figure 3-3 Samples from the pre-collected dataset.*

### 3.1.1.2 Imbalanced Data and Data Augmentation

Another factor in model accuracy is the data balance factor. The data is called, but it is not balanced when the number of images of each class is not close to the number of images for the rest of the classes. For example, when we have a dataset of 1200 images of 3 different classes. Let's say the

data includes 400 images for tomatoes, 200 for potatoes, and 600 for olives, then the dataset is called an unbalanced dataset. So how does the unbalanced data affect prediction accuracy? The model tends to prioritize learning to predict the class that has a higher number of images, and this led to high accuracy in general, but poor performance for the classes that have fewer images. The Figures 3-4 describe the 3 methods to achieve the balance:



*Figure 3-4 3 methods solving the imbalanced data.*

However, there are different ways to solve the unbalanced data problem like oversampling, Undersampling, and Data Augmentation:

1. **Oversampling:** It is a technique in which the data will be repeated many times to achieve balance. In the previous example, both the number of tomatoes and potatoes will be increased to 500-600 which is the maximum number of images in one class.
2. **Undersampling:** It is a technique in which the data will be decreased to achieve the balance. In the previous example, both the number of olives and tomatoes will be decreased to 200-300 which is the minimum number of images in one class.
3. **Data Augmentation:** Data augmentation is a technique in machine learning, that increases the training set by creating modified copies of a dataset using existing data [9]. It includes making minor changes to the dataset or using deep learning to generate new data points. Augmented data is derived from original data with some minor changes. Plantie application

used image augmentation techniques, that make geometric and color space transformations to increase the size and diversity of the dataset. In plant disease detection using deep learning, rotation, flipping, re-sizing, brightness, contrast, and cropping are common image augmentation techniques. image augmentation techniques have several advantages as reduced overfitting, improved model accuracy, and enhanced model generalization.

### 3.1.1.3 Image Preprocessing

Image preprocessing is the steps taken to format images before they are used by model training [10]. As shown in Figure 3-5, image preprocessing includes Resizing, Normalization, Contrast enhancement, and Noise reduction. This process leads to faster training and better performance.

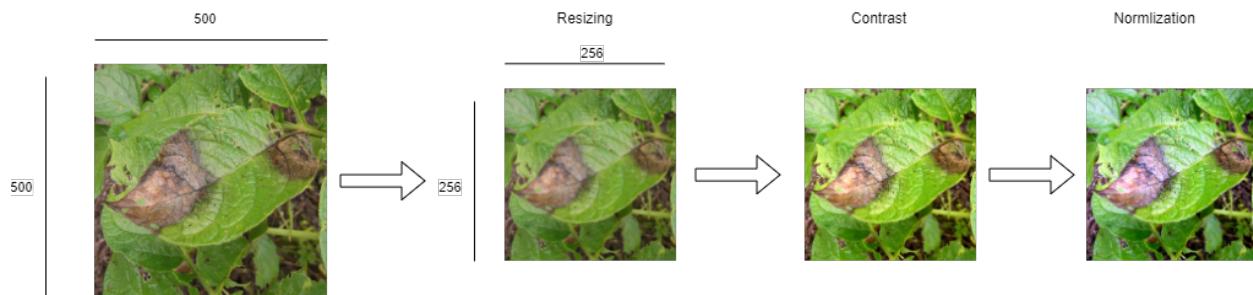


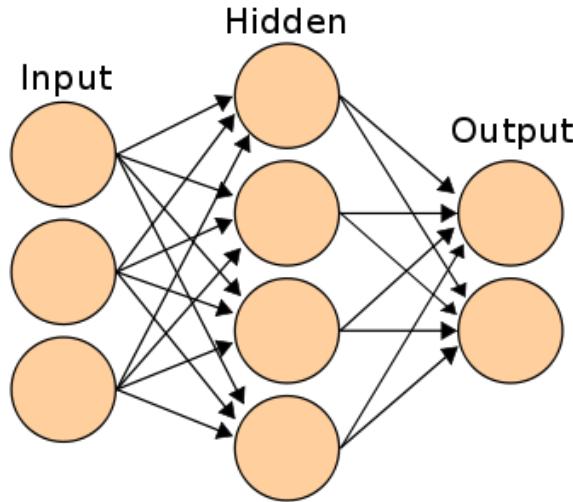
Figure 3-5 Applying image preprocessing.

### 3.1.1.4 ANN Algorithm

An artificial neural network is a powerful tool in the world of machine learning. An artificial neural network is several nodes connected that form a network like the one found in the human brain. Each interconnection has a weight, and this weight is useful in the process of activating the neuron. An activation function is a function that uses weight to decide which interconnection to take to provide a transit path for another interconnection. The core feature of ANNs is that the ANNs use the interconnected nodes to process information and learn from experience, and this process of learning from experience is called training, the ANNs model trains from the dataset and it will make changes to the weight of each interconnection. ANNs employ 3 layers called the fully connected layers, and these layers consist of the input layer, hidden layers, and the output layer. Figure 3-6 shows how the 3 layers are connected.

- Input Layer: This layer receives input data, usually, referring to the dataset features.

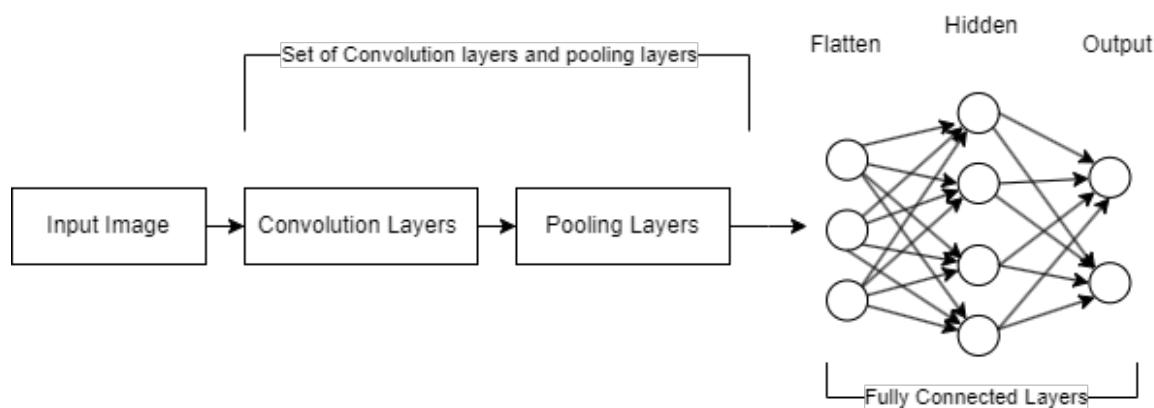
- Hidden Layers: These layers are a set of layers that relate to the input layer and output layer.
- Output Layer: refers to the last neurons that represent the final prediction (label).



*Figure 3-6 ANN Algorithm layers.*

### 3.1.1.5 CNN Algorithm

CNN is a category of machine learning model, namely a type of deep learning algorithm [11]. As mentioned before, CNN is the most suitable algorithm for object detection and recognition. Like ANN, CNN has 3 layers (input layer, hidden layers, output layer) plus some layers for handling images called convolution layers and pooling layers. In turn, the convolution layers work together to extract features from the image. Followed by pooling layers to reduce the dimensionality of the extracted features. This process will result in a set of features to be inserted into the ANN to make predictions. Figure 3-7 shows the general form architecture for the CNN algorithm.



*Figure 3-7 CNN Layers*

---

## Pseudocode CNN model layers

---

```
create_model()
# Convolutional layers with subsampling (pooling):
add_convolutional_layer(number_of_filters,filter_size,stride,activation='tanh')

add_average_pooling_layer(pool_size)
add_convolutional_layer(number_of_filters, filter_size, stride,
activation='tanh')
add_average_pooling_layer(pool_size)

# Fully connected layers:
flatten_layers()
add_fully_connected_layer(number_of_neurons, activation='tanh')
add_fully_connected_layer(number_of_neurons, activation='tanh')

# Output layer (number of output neurons depends on the task):
add_output_layer(num_classes, activation='softmax')
```

---

### ***Convolution Layers***

The convolutional layer is the core building block of a CNN algorithm which employs multiple filters stacked together to extract features from images, each filter has a specific( $n \times m$ )size to extract a specific feature. To understand more about the filters, imagine that there's a human face image, the human face has some features like ears, eyes, mouth, and so on. Each filter tries to extract one of these features. In the end, the output of a convolution layer is a set of feature maps. The feature map is a 2D array that stores the feature result of each filter. Moreover, the filter will slide among the image input and then store the data in the feature map array. Figure 3-8 show how the filters will slide and fill the feature map.

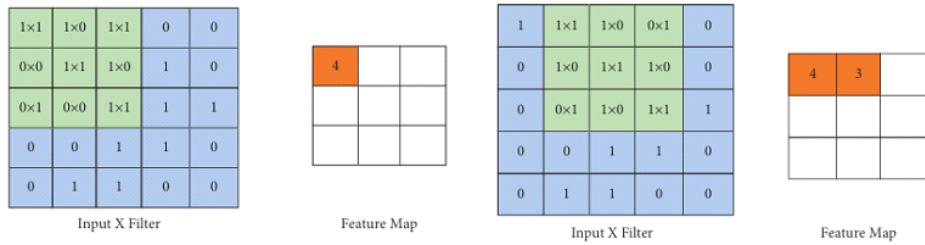


Figure 3-8 Generating feature map for a single filter.

Notice that the numbers within a feature map represent the activations of a particular filter at different spatial locations in the input image.

### **Pooling Layers**

Pooling layers play a crucial role in reducing the dimensionality of feature maps by applying operations like Max pooling and Average pooling. The pooling operation involves sliding a two-dimensional filter over each channel of the feature map and summarizing the features lying within the region covered by the filter [12]. The pooling uses a sliding window with  $(n \times m)$  size to calculate the average or max of the pixels within the rectangular region of the window. The result is a down sampling of the feature map. Figure 3-9 shows how the filters work and how the sliding windows are sliding to fetch and collect the features from the image:

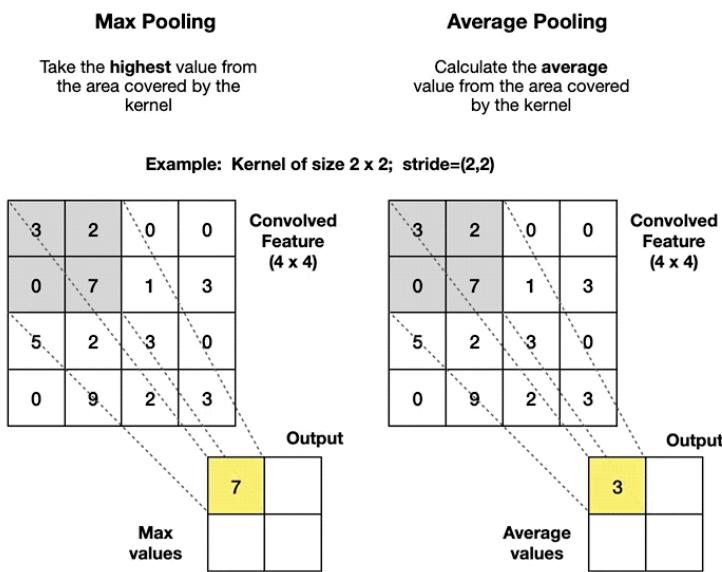


Figure 3-9 Max and Average Pooling

## **Fully Connected Layers**

The fully connected layers refer to the traditional neural network layers which take the input from the previous feature maps, they combine the extracted feature repairing to the final output. The input layer is called flatten; from its name, the image input will be represented into a 1D array to be ready to go as an input in the ANN. As the ANN there are hidden layers and output layers, the operation here is the same as those in the ANN.

## **Proposed Model Architecture**

The Figure below is the basic architecture of the CNN and it's called LeNet. LeNet-5 was one of the earliest convolutional neural networks and promoted the development of deep learning [13]. In the context of CNN, many different architectures exist, and each one has advantages. As we mention in the related works the plant disease detection projects used different architecture, and some of them used their architecture like the “Potato disease detection”, and the “Cucumber diseases detection”. The CNN architecture that we used in this project is the basic MobleNetV2 model, the model as shown in Figure 3-10 shows the layers of the basic CNN architecture:

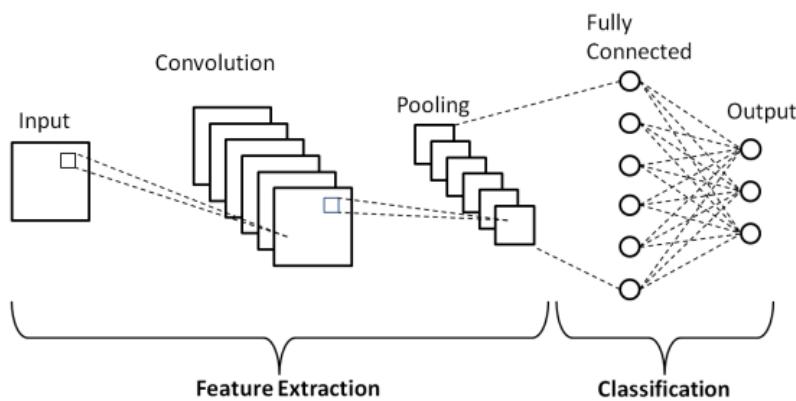


Figure 3-10 LeNet architecture

### **3.1.2 Fertilizer Calculator**

Fertilizer is important in the process of plant growth. The new farmers may struggle with the calculation of several fertilizers. When we want to calculate the number of fertilizers that should be applied to the plant, we must know that 2 things must be known, the N-P-K, and the land size.

The letters "NPK" on a fertilizer label stand for nitrogen, phosphorus, and potassium, the three primary nutrients plants need to grow [14]. The N-P-K rate refers to nitrogen (N), phosphorus( $P_2O_5$ ), and potassium ( $K_2O$ ), and these refer to how much nitrogen, phosphorus, and potassium the plant needs. On the other hand, there some kinds of fertilizers contain a specific amount of these compounds, for instance, the Urea fertilizer contains 46% nitrogen which can be represented as 46-0-0 in the N-P-K form, the MOP contains 60% Potassium, and the SSP contains 16-18% of phosphorus [15]. We can calculate the amount of fertilizer for each of these fertilizers by the equation that could be found in 'study on manures and fertilizers' paper [15]:

$$\text{the required amount of fertilizer} = \frac{100 \times \text{Dose of nutrient}}{\text{Nutrient content in the applied fertilizer (\%)}}$$

For example, if we know that the average N-P-K of the cucumber is (150-75-50), then the required amount of each fertilizer is like the following:

$$\text{The required amount of Urea} = (100*150) / 46 = 320 \text{ kg ha}^{-1}$$

$$\text{The required amount of SSP} = (100*75)/16 = 468.75 \text{ kg ha}^{-1}$$

$$\text{The required amount of MOP} = (100*50)/60 = 83.3 \text{ kg ha}^{-1}$$

All the results here are measured by kilogram per hectare ( $\text{ha}^{-1}$ ), the one hectare equals 10000 square meter ( $\text{m}^2$ ). If the size of the land is more or less than that, then the number of fertilizers will be different. Suppose that land is  $2 \text{ ha}^{-1}$  In the previous example, then the result will multiply by 2. However, the fertilizer calculator will calculate the amount of fertilizer that the plant needs and will show you how much N-P-K the plant needs. In addition, the process of calculating the amount of fertilizer is not sufficient, as the amount of fertilizer must be shown **for each week**. Because all these numbers of fertilizers are not applied immediately when the plant is in the early growth stage. The calculator should show how you divide the amount of fertilizer by weeks.

There is a different combination of fertilizers which is the Urea, DAP and SSP to calculate this we should know that As DAP supply both P and N, we have to calculate the amount of DAP first. At first, we calculate the amount of P as it presents in higher quantity [15]. We know: In DAP, %N = 18 and %  $P_2O_5$  = 46, In urea, %N = 46, In MOP, %  $K_2O$  = 60

The following code is written in Python and shows how the calculation will be done:

---

## Pseudocode 2 algorithms for fertilizer calculations

---

```
function UMS_fertilizer_calculator(NPK):
    ...
    UMS refer to the Urea, MOP and SSP
    take NPK as input
    ...
    amount_of_Urea = (100 * NPK[0])/46
    amount_of_SSP = (100 * NPK[1])/16
    amount_of_MOP = (100 * NPK[2])/60
    return {
        'Urea': amount_of_Urea,
        'MOP': amount_of_MOP,
        'SSP': amount_of_SSP,}
function UDN_fertilizer_calculator(NPK):
    ...
    UDN refers to the Urea, DAP, and MOP
    take NPK as input
    ...
    amount_of_DAP_P = (100 * NPK[1] ) / 46
    amount_of_DAP_N = (18 * amount_of_DAP_P) / NPK[0]

    amount_of_Urea_N = NPK[0] - amount_of_DAP_N
    amount_of_Urea = (100 * amount_of_Urea_N) / 46
    amount_of_MOP = (100 * NPK[2])/60
    return {
        'Urea': amount_of_Urea,
        'MOP': amount_of_MOP,
        'DAP': amount_of_DAP_P,}
```

---

### 3.1.3 Find the Nearest Plant Shop

Plantae's "Find Nearest Plant Shop" feature improves user experience by assisting users in finding the closest plant shop concerning their present location. Map services such as Google Maps or Mapbox are used by the app, which makes use of the device's location capabilities and a variety of plant shops containing names, addresses, and maps. By considering road networks, the system determines the journey times and distances between each plant shop and the user's location while maintaining accuracy. Iteratively measuring the minimum trip distance, the app finds the closest

shop after filtering the easy-to-use interface gives users access to information about the shop, including its name, address, estimated trip time, and any other pertinent characteristics. list of plant shops depending on the user's city or surrounding cities. An

---

### Pseudocode 3

```
function findNearestPlantShop(userLocation):
    # get the array of plant shops with names, addresses, and coordinates
    plantShops = getPlantShops()
    # Filter plant shops based on the user's city or nearby cities
    nearbyShops = filterShopsByCity(plantShops, userCity)

    # Initialize variables for tracking the nearest shop
    nearestShop = nil , minDistance = maxInt (infinity)

    # Loop through each nearby shop
    for shop in nearbyShops:
        # Calculate travel distance and time from user location to the shop
        travelInfo = calculateTravelInfo(userLocation, shop.location)
        # get the nearest shop
        if travelInfo.distance < minDistance:
            minDistance = travelInfo.distance
            nearestShop = shop

    # Display the details of the nearest shop to the user
    displayShopDetails(nearestShop)

# Helper function to calculate travel distance and time using mapping services
function calculateTravelInfo(userLocation, shopLocation):
    # Use mapping services (Google Maps or Mapbox) to get distance and time
    # This involve API calls to the mapping service
```

---

#### 3.1.4 Weather System

Another feature to help farmers is the weather system. Knowing the weather conditions will help farmers in the farming process. The system will provide the farmer with best practices or suggestions based on weather conditions. For instance, if the weather is rainy, it is a good idea for the farmer to supply fertilizer on that day. The system will use an API to provide it the real-time

weather status and will detect what is better to be done based on the status. The following pseudocode explains how the suggestion works:

---

#### Pseudocode 4

---

```
status, temperature, humidity = GoogleWeatherAPI()

if status == 'Sunny' and temperature > 30:
    send_data('The Day is good for framing')

if status == 'Cloudy' and temperature < 25 and humidity < 50%:
    send_data('The Day is good for Applying Irrigation')

if status == 'Rainy' and temperature < 15:
    send_data('The Day is for Giving Pesticide')
```

---

The system connects with an Open Weather API to provide the status, temperature, humidity, and more details based on the user's location. Then the system will search in the knowledge base to retrieve the optimal suggestion based on this information. After the detection what the suggestion the system will show it in the UI. Figure 3-12 shows the weather condition status and the optimal Suggestion:



Figure 3-11 Weather conditions.

### 3.1.5 Community System

Sharing the experience between farmers is an efficient thing, especially for those who are new in the agriculture field. In any application, the community makes it more interactive and keeps farmers up to date with new features in this field. Give the farmers the ability to ask about the plants, new fertilizers, and the best practices. Plantie is not just a tool that allows you to predict the disease, it also provides you the ability to gain more experience from other farmers. The community system allows you to post questions, like, and comment on other posts. Rather than searching or asking in other places, all you need is one hub.

Moreover, the community system helps us to improve the application in the long term by keeping track of the feedback of the users. This user feedback allows us to be knowledgeable about the up-to-date features that the system should provide. Like any generic software application, feedback is the key point of improving the system and any measure of acceptance between the users. Figure 3-13 shows the functionalities of the community system:

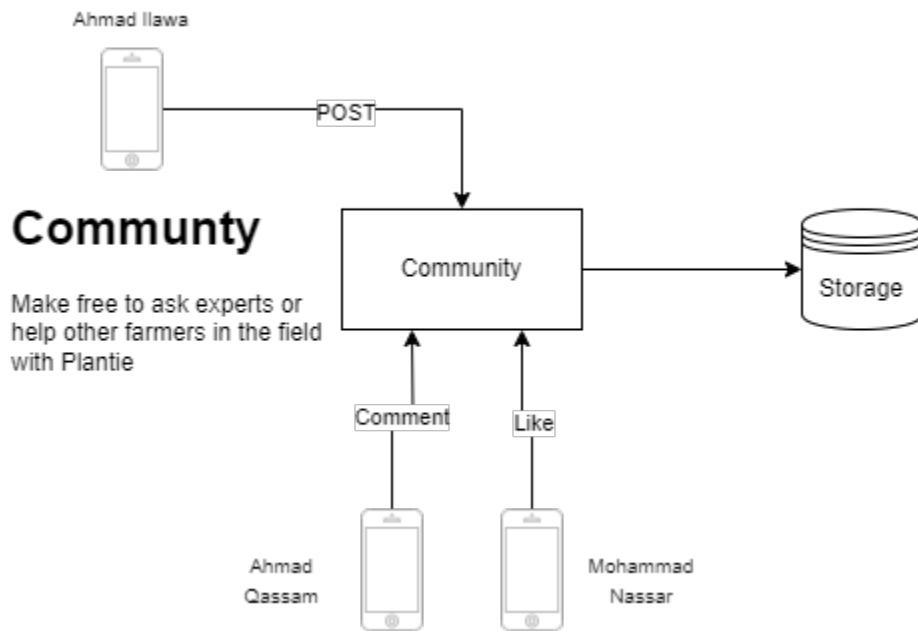


Figure 3-12 Community System

When we talk about a community system, we assume that there are users and there are authorization and authentication systems. Anyway, the community system provides features including:

- Post Questions: the users can ask the community about the field.
- Post Preview: All posts will be shown to all users to communicate with them
- Comment: users can comment/replay to the posts to help them find the solution to their problems
- Like The user can like or dislike posts, which helps to increase the reach of this post.

## 3.2 System design

System design includes important abstraction models like Sequence Diagrams, Database Designs, and Architecture Designs, which together provide a thorough representation of the project's organization. These models offer a blueprint for efficient development and implementation in addition to describing the elements and interactions of the system. The system design process guarantees a well-organized and planned approach to developing a reliable and scalable solution by using these design features.

### 3.2.1 Architectural Design

As shown in Figure 3-14, the application will send an image to the server and the server will predict the pre-trained model and then will retrieve the prediction. Also, the server can make queries in the database.

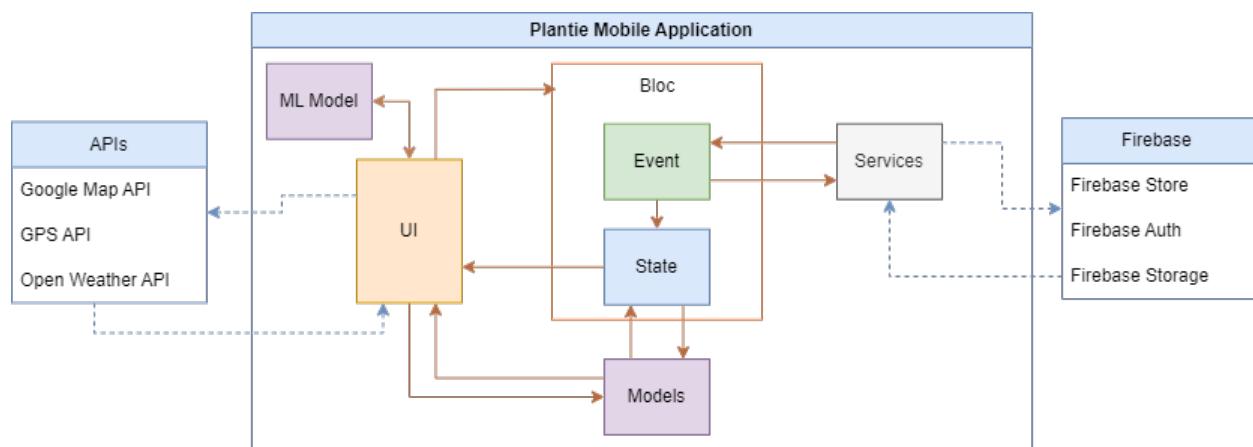


Figure 3-13 System Architecture

Bloc and REST API (Representational State Transfer) are smoothly integrated to create a strong architectural framework in the Plantie App. In this design, the Event serves as the go-between, the UI controls the user interface, and the state contains the UI states. The Model contains the data and

business logic. The use of the REST API makes communication easier. The Plantie App can offer a dynamic and responsive user experience because of its improved scalability, maintainability, and flexibility, which also ensures adaptability for upcoming improvements and external integrations. In addition, the Bloc design encourages the division of responsibilities, facilitating the management of intricate interactions inside the application. Users always have access to the most recent information since REST APIs allow for seamless data synchronization between the app and server. Additionally, this architecture makes data validation and error handling more efficient, which improves the overall stability of the app. The code is more dependable and of higher quality because of the modular design, which makes testing and debugging simpler.

### **3.2.2 Database Design & UML Diagram**

The database architecture of the Plantie App is set up to effectively manage various app features. The Users table contains the user's data, including login credentials and profile information. Information on different plants and related diseases is kept in the Plants and Plant Diseases tables. Users can interact with individual plants they've dealt with by using the User Plants table, which also notes diseases, photos, and detection details. The posts table manages user-generated material and keeps track of favorites and dislikes. By using foreign keys to connect the user and the post, the "Comments" table collects user-generated comments on postings. Environmental data about user locations is captured by the Weather table. Well-defined foreign key constraints and data integrity are the foundation for establishing relationships between tables.

The UML as shown in Figure 3-15 describes the database schema and the main tables and entities of the Plantie application. The tables are the user table for the authentication and authorization process, the User Plants which contains all plants of the user, the Plant Disease table which contains the details about the disease itself, the plant's table which contains information for each plant, and the last 3 tables are for the community which include the post, comment, and likes.

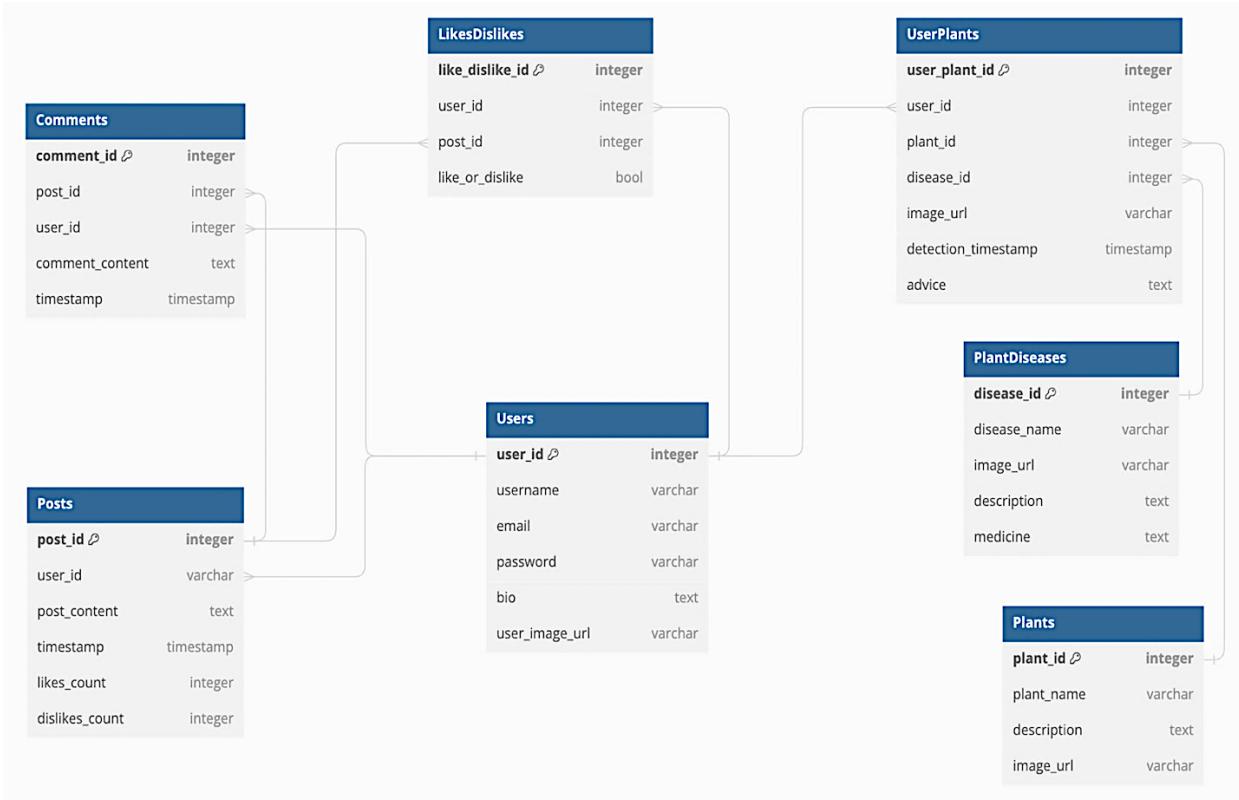


Figure 3-14 Proposed system database UML Diagram

### 3.2.3 User Scenarios

1. Scenario - User Registration via Email:
  - a. The user receives an email prompting them to register.
  - b. The user gives their username.
  - c. The user generates a strong password.
  - d. The database contains the verified registration data.
  - e. The user is taken to the profile page after completing the registration process.
2. Scenario - User Registration via Gmail:
  - a. The user chooses to sign up with their Gmail address.
  - b. The application asks for authorization to use the Gmail account.
  - c. After the user verifies access, pertinent data is retrieved.
  - d. The application verifies and saves the user's data.
  - e. The user is taken to the profile page after completing the registration process.

3. Scenario - User Registration via Facebook:
  - a. The user decides to sign up with their Facebook credentials.
  - b. The application asks permission to access user information on Facebook.
  - c. The user allows access, and the required data is obtained.
  - d. The application verifies and stores the user's information.
  - e. Following a successful registration, the user is taken to the page featuring their profile.
4. Scenario: User logs in:
  - a. The user is asked for their email address.
  - b. The user is asked for their username.
  - c. The user is prompted for a password.
  - d. The profile page opens, and the user is successfully logged in if his credentials match those in the database.
5. Scenario: The user is unable to log in.
  - a. An email request is made to the user.
  - b. A password request is made to the user.
  - c. If the user's credentials do not match those in the database, the user is not logged in.
  - d. There will be a prompt displayed.
  - e. He'll have to submit his information again.
6. Scenario: The user modifies his profile. He can do the following:
  - a. Change his name.
  - b. Modify his bio and user image.
  - c. Modify his password.
7. Scenario - User Posts in Community View:
  - a. The user opens the community view.
  - b. The user writes and/or uploads images to a new post.
  - c. After verifying the post, the program saves it in the database.
  - d. The post is viewable and interactable by other users.
  - e. The user receives notifications when other people engage with the post.
8. Scenario - User Comments in Community View:

- a. The user chooses a post after gaining access to the community view.
  - b. The user comments on the post they have chosen.
  - c. The application verifies and logs the feedback.
  - d. Other users can view the comment by scrolling down below the post.
  - e. The original poster of the post and other participants in the discussion receive notifications.
9. Scenario: User Makes Disease Detection:
- a. The user opens the detection view.
  - b. The user photographs a plant that might be ill.
  - c. The software analyses the image using a machine-learning model.
  - d. The user gets immediate feedback regarding the illness that was identified.
  - e. The app might offer suggestions for managing diseases.
10. Scenario: User Accesses Plants View with Weather and Fertilizer Calculations:
- a. The user views the plants.
  - b. The user chooses a certain plant from the list.
  - c. The application shows the user's location's current weather information.
  - d. Based on the type of plant, the app determines and recommends the amount of fertilizer needed.
  - e. The user gets thorough guidance on how to take care of their plants.
11. Scenario: User Accesses Map:
- a. The user views the disease detection result view.
  - b. The user presses the button to navigate to the Map.
  - c. The user shows the nearest plant shop.

### 3.2.4 Use Case Diagram



Figure 3-15 Use case for the Plantie app

### 3.2.5 Sequence Diagram

#### 3.2.5.1 Login/Register:

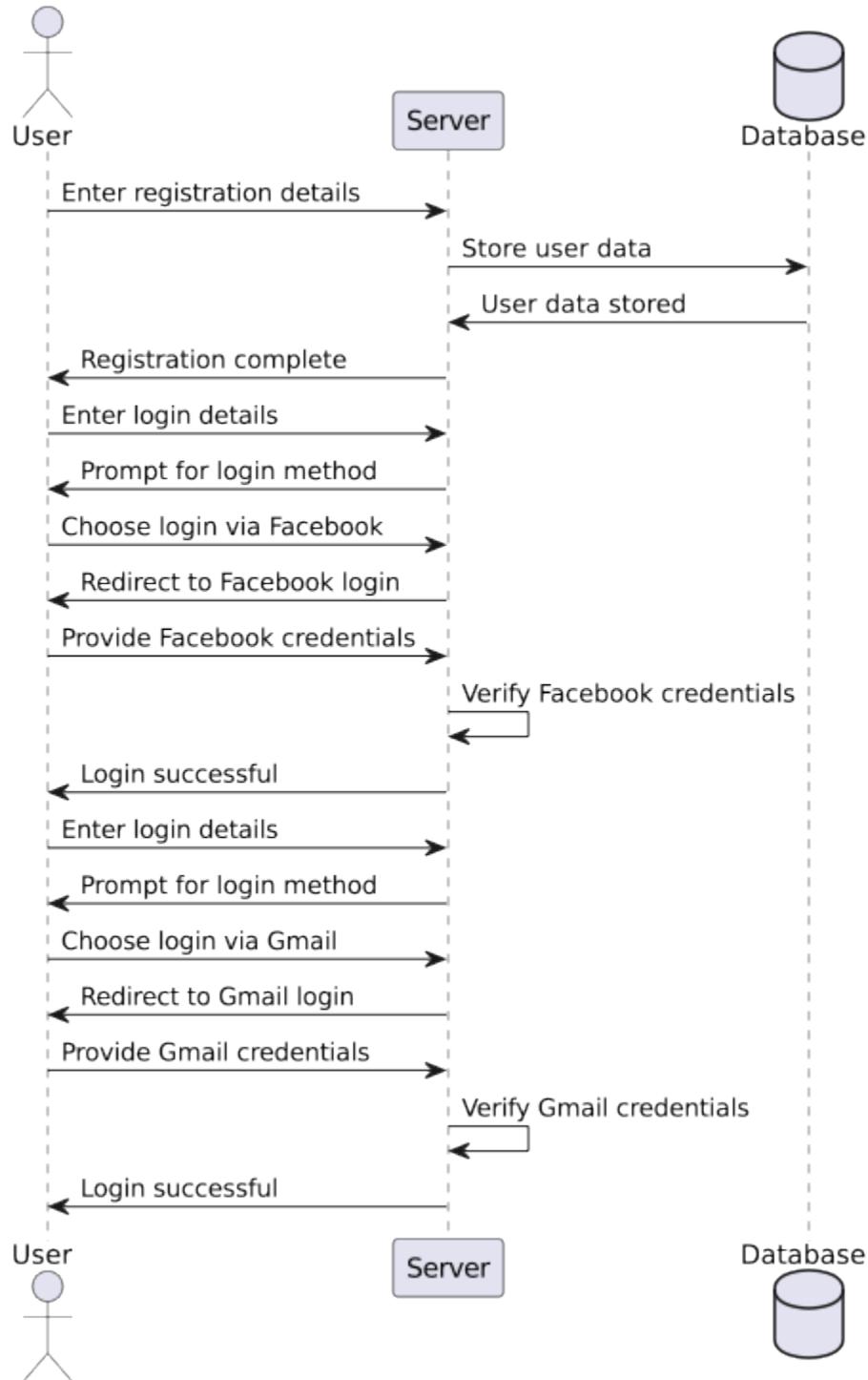


Figure 3-16 Login/Register Sequence Diagram Post on the community/Comment on the post

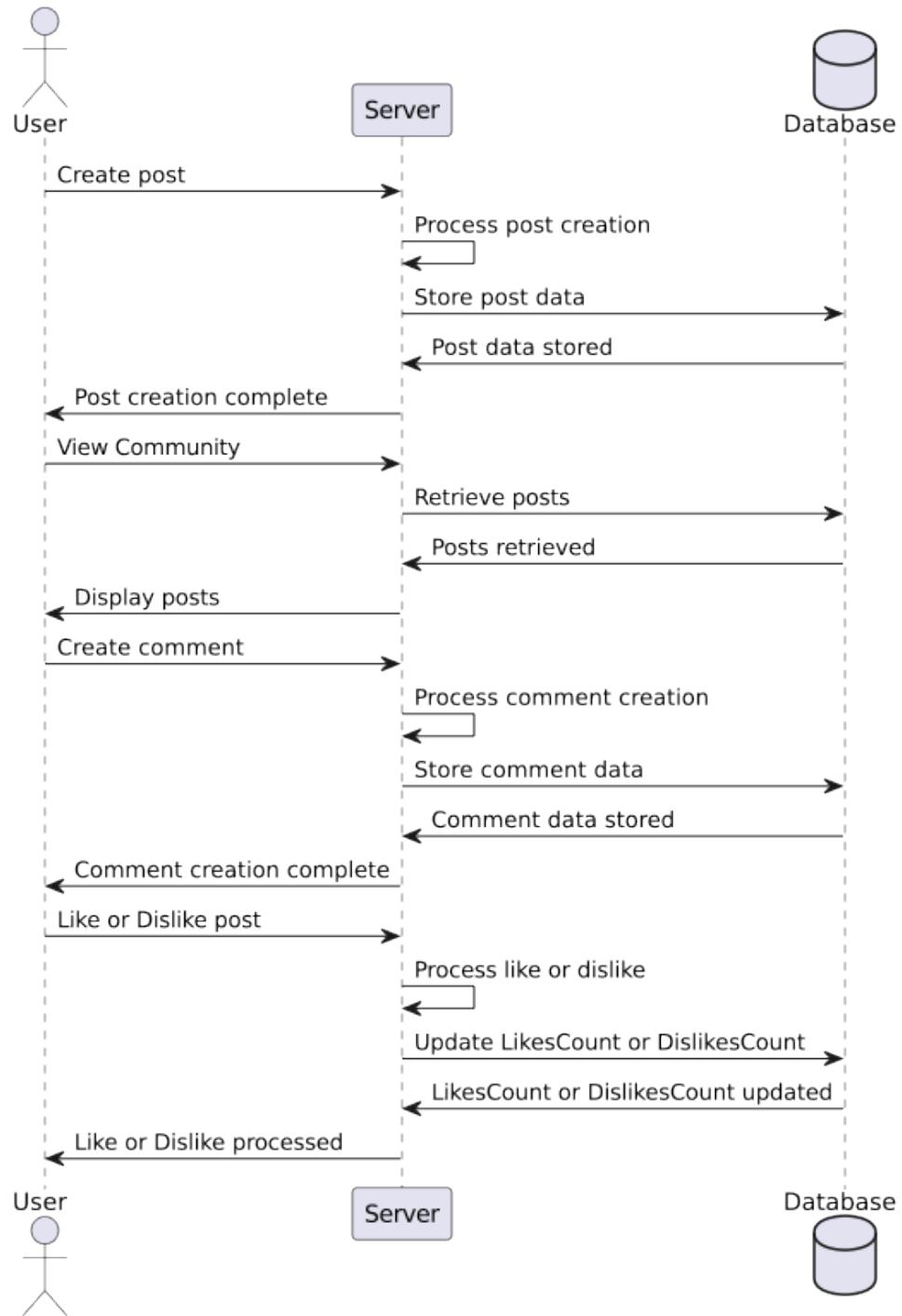


Figure 3-17 Post on community/Comment on the post Sequence Diagram

### 3.2.5.2 List view

- This view contains two sub-views the first for a list of plants, and the second for showing the real-time weather based on user location.
- When clicking on a plant, a sub-view appears behind the information of the plant for calculating fertilizer.

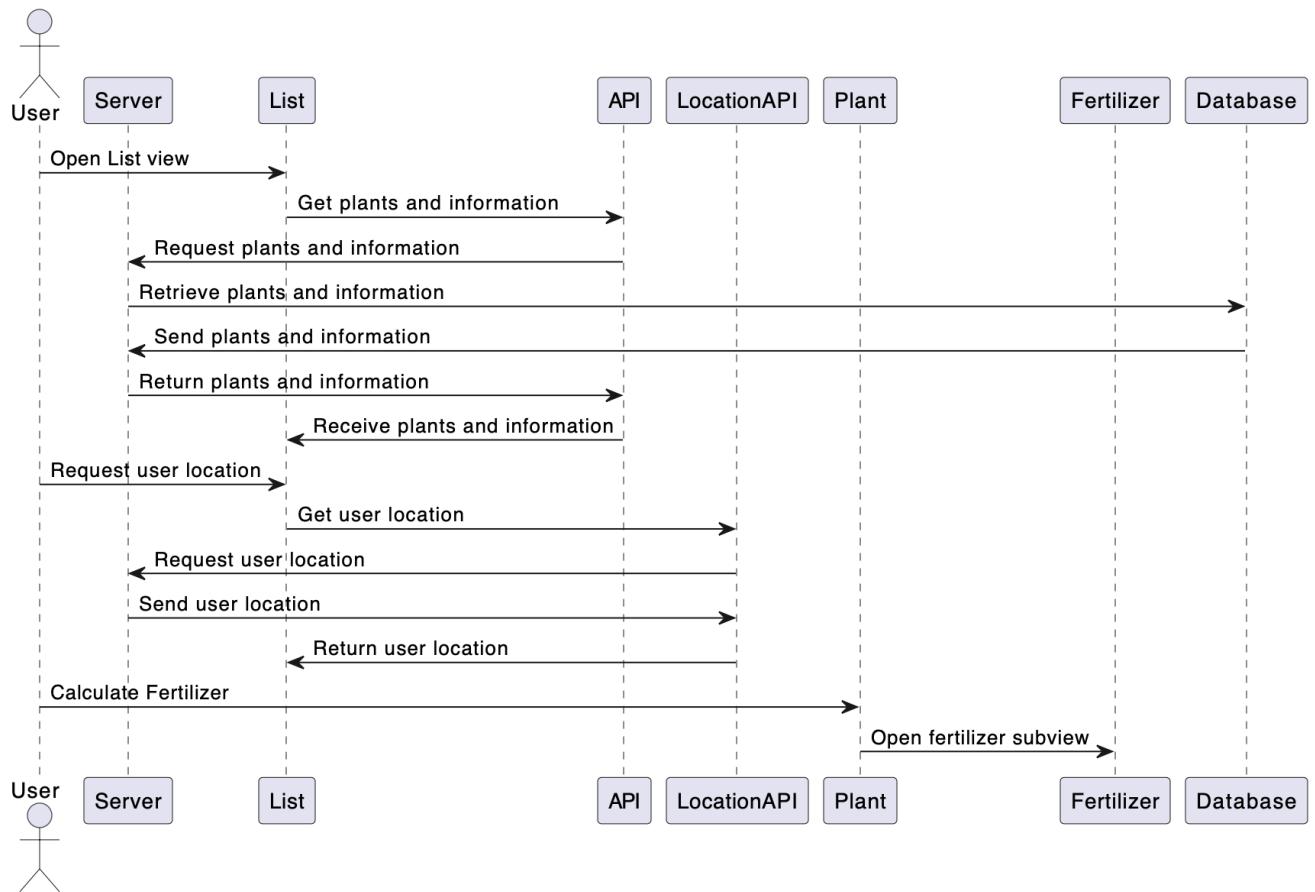


Figure 3-18 List View Sequence Diagram

### 3.2.5.3 Detection of the plant disease using the ML Model:

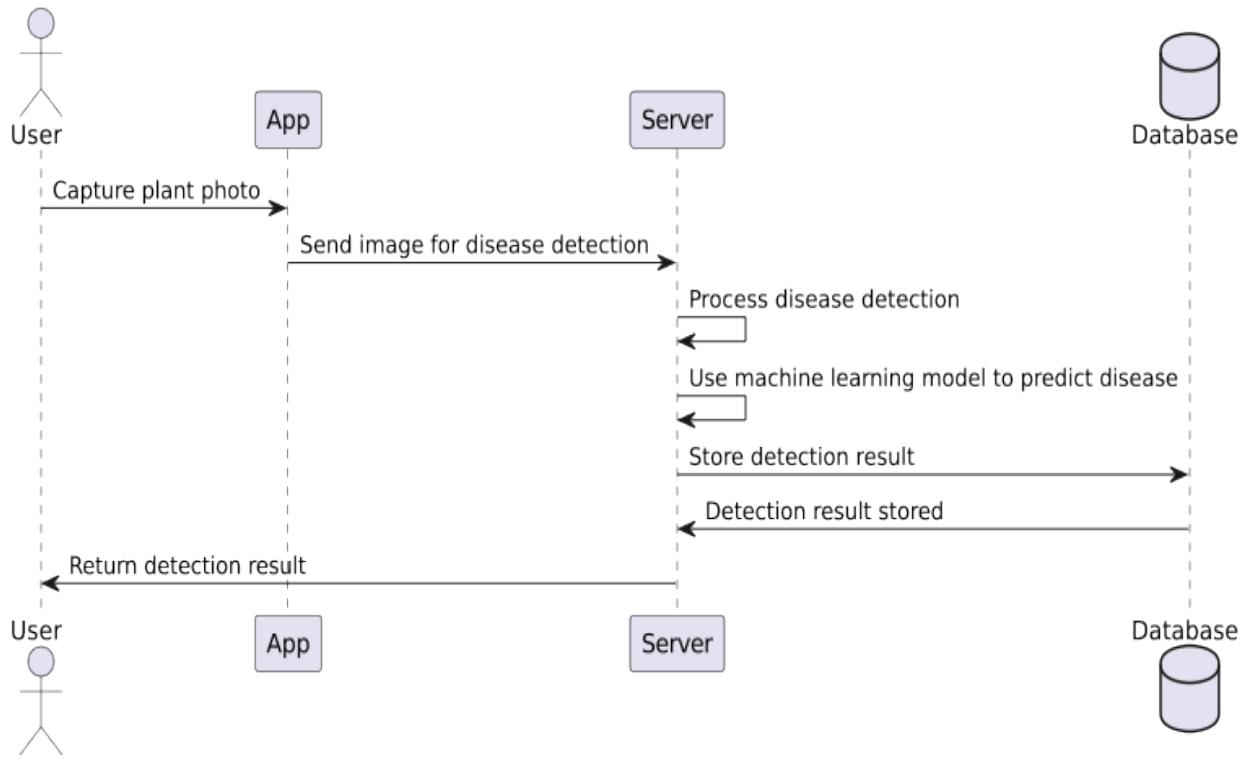


Figure 3-19 Detection of the plant disease using the ML Model Sequence Diagram

### 3.2.6 ER Diagram

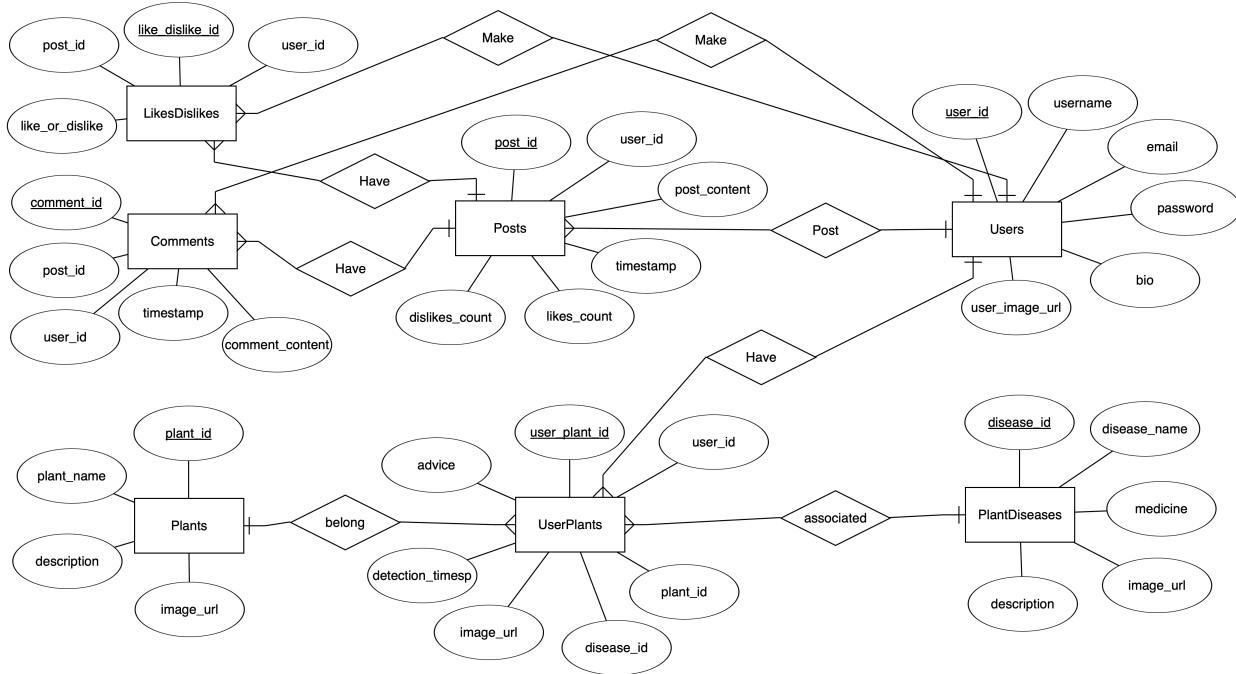


Figure 3-20 ER Diagram

## 3 . 3 S o f t w a r e a n d T o o l s

### 3.3.1 GitHub

GitHub provides hosting and review for the source code which helps us to collaborate in the development process. Also, we used GitHub for uploading agendas and Meeting Reports for each meeting we had. The GitHub and any Version Control System (VCS) is a core tool for developers to work together remotely and reflect the different changes in one hub. GitHub allows creating an organization to work within the group with the ability to customize and control the team members' roles.



### 3.3.2 Google Colab



A free cloud-based platform owned by Google LLC, Google Colab is particularly well-liked by data scientists and machine learning practitioners. It is specifically made for executing Python code. To jointly construct and assess our models, including our "Plantie Model" for plant disease detection, we used Google Colab for shared testing. Colab is a great platform for team projects since it offers the libraries and tools required for creating, refining, and testing machine learning models. Google Colab's collaborative features, which let multiple users work on the same notebook at once, are one of its main advantages. This facilitated seamless teamwork and enhanced productivity.

### 3.3.3 Draw.io



Draw.io is a web platform that allows us to create diagrams such as flowcharts, wireframes, UML diagrams, organizational charts, and network diagrams. With the rich full features that Draw.io you can manage all these much easier. All of that is web-based which gives the ability to work with teams. Tool rich in features and templates that provide the ability to do the diagram comfortably. Don't forget that the offline version of Draw.io has all the features of the online version, except saving to the cloud and sharing online.

### 3.3.4 Figma

Figma is a web application for interface design. We used Figma to design our application UI. Figma provides a community that helps us to find people's works and designs which help us so much in our UI design process, you can copy any design-available in Fig and apply your changes as well. Unlike Adobe XD, Figma is free with the ability to create an infinite number of private designs. Figma allows you to work with the team and reflect on the changes in the design.



### 3.3.5 Trello



Trello is a project management tool that we used to manage the tickets and tasks. Trello is suitable for small teams like ours, with the ability to provide many useful features such as automation and a simple user interface.

### 3.3.6 Dropbox



Dropbox is a file hosting service that allows us to synchronize files between us. Also, we use it as storage to store the papers and some related works that could help us in one place. Dropbox provides file version history which allows us to restore any version when want whatever we want.

## 3.4 Design Specifications, Standards, and Constraints

### 3.4.1 Functional Requirements

*Table 2 System Requirement Table*

Req#	Requirement	Comments	Priority	Date	Reviewed / Approved
R_0_1	User registration and authentication.	Users can register with a unique username and email.	1	23/12/2023	Team
R_0_2	Option to log in via Facebook or Gmail.	-	2	23/12/2023	Team
R_0_3	Users can edit their profile information.	- Validation of user inputs. - let users store and retrieve information securely.	1	23/12/2023	Team
R_0_4	Community Page.	To let users, communicate with each other's and	1	23/12/2023	Team

		exchange information for the plants.			
R_0_5	Users can create posts.	let users create posts on the community and make comments as likes/dislikes.	1	23/12/2023	Team
R_0_6	List of plants with information.	Fetch the information via API to get up-to-date information.	1	23/12/2023	Team
R_0_7	Plant detection with machine learning model.	<ul style="list-style-type: none"> <li>- Let users take photos or choose a photo from a library of plants for disease detection.</li> <li>- Utilize a machine learning model to detect diseases.</li> <li>- Provide a simple-to-use interface for taking or uploading pictures of plants.</li> </ul>	1	23/12/2023	Team
R_0_8	Weather Integration and give advice based on the state of the weather.	<ul style="list-style-type: none"> <li>- Fetch and display real-time weather information based on the location of the user.</li> <li>- Give general advice for plants based on the state of the weather.</li> </ul>	1	23/12/2023	Team
R_0_9	Fertilizer calculator.	<ul style="list-style-type: none"> <li>- Provide a simple-to-use interface for the Fertilizer calculator.</li> </ul>	2	23/12/2023	Team

### 3.4.2 Non-Functional Requirements

#### 3.4.2.1 User Interface Requirements

1. Different screen resolutions and appearances are based on devices.
2. Tap the bar controller for the App for simple moves between views.
3. Responsive simple design.

### **3.4.2.2 Usability**

#### **➤ Accessibility**

- ❖ The app should be easy to always access. This app should be easy to access only with your name and email or via Facebook/Gmail.

#### **➤ Responsiveness**

- ❖ The app shall be responsive both in data transactions, especially because of the reliance on the servers.

#### **➤ Flexibility**

- ❖ The app shall be easy to update to accommodate new requirements.
- ❖ The app shall be designed in such a way that allows us to make changes without starting from the beginning.

#### **➤ Efficiency**

- ❖ The performance of the application will be influenced by how well the server and REST API work.
- ❖ As the software is planned to be fast, there should be an internet connection of at least 20 dedicated Megabytes of internet.

### **3.4.2.3 Performance**

#### **➤ Capacity**

- ❖ Several users at Once: During periods of high demand, the server should be able to manage several users at once with efficiency. Strive for scalability to handle expansion.
- ❖ Concurrent API queries: To ensure responsiveness and minimal latency, design the REST API to support a sizable number of concurrent queries.
- ❖ Database Scalability: To handle the growing amount of user and plant data, the database should scale horizontally. Use partitioning or sharding techniques as necessary.

#### **➤ Availability**

- ❖ The app will be live 24/7.
- ❖ It has a very low probability of downtime.

- ❖ Monitoring and Alerting: Keep tabs on API performance, server health, and other important data by utilizing monitoring tools. Configure automated notifications to receive instant alerts, when possible, problems arise.

#### **3.4.2.4 Maintainability**

- ❖ The application is simpler to update and maintain when its components are separated.
- ❖ Principles of RESTful design encourage a modular and easily maintainable codebase.

#### **3.4.2.5 Security**

- Specify the factors that will protect the system from malicious or accidental access, modification, disclosure, destruction, or misuse. For example:
  - ❖ Encryption.
  - ❖ Activity logging.
  - ❖ The application should also include a list of modules and components that will be required to be accessed from the application, as known as permissions.

#### **3.4.2.6 Data Management**

Careful procedures are needed at every stage of the data lifecycle to ensure proper data handling for the Plantie app. This entails the limited and safe gathering of user data, the use of strong encryption techniques for data in transit and storage, and the implementation of an organized, safe database design with suitable access controls. User privacy and data processing efficiency are improved by asynchronous processing and adherence to API security requirements.

#### **3.4.2.7 Standards Compliance**

By utilizing strong system encryption for increased safety and security, the Plantie app makes sure that legal requirements are met.

#### **3.4.2.8 Portability**

To improve accessibility and user reach, the Plantie app must be portable to adapt and work flawlessly across a range of platforms and devices so, we use Flutter cross-platform.

### **3.4.2.9 Domain Requirements**

Everything about the domain that could be required for the project will be covered in this section. It is possible to consider the domain requirements to be a component of either non-functional or functional requirements at times.

## **3.5 Programming languages and frameworks**

### **3.5.1 Flutter**



Flutter is a cross-platform framework that allows developers to use a single codebase for both Android and iOS app development, saving time and resources. It offers a wide range of customization widgets that can be styled to match the design of the app. Flutter has an active and vibrant community of developers who actively contribute to the framework's development and share their knowledge. It is widely regarded as a suitable option for creating a Minimum Viable Product (MVP) as it enables developers to quickly build and deploy an app without spending excessive amounts of time and resources. Community feedback can also be utilized to improve the system [17].

### **3.5.2 Python**



Python is a popular high-level, interpreted, and general-purpose programming language widely used in the field of machine learning and artificial intelligence. It comes equipped with various libraries and frameworks like TensorFlow, PyTorch, scikit-learn, Keras, etc. These pre-built functions and tools can be used to develop, train, and deploy machine learning models, making it easier for developers to focus on the modeling part without worrying about implementation details.

### **3.5.3 Firebase**



Firebase is a collection of tools and services that developers can use to create mobile and web applications [18]. It has several features such as real-time data synchronization, user authentication, and machine learning services. Firebase is renowned for its user-friendliness and speed, making it the ideal choice for developing real-time collaborative applications. It is a Google-owned product that provides developers with a complete set of tools to create robust and scalable mobile apps. One of its primary features is authentication, which simplifies the process of user registration and login. This significantly speeds up the authentication process for

## **3 . 6 D e s i g n A l t e r n a t i v e s**

### **3.6.1 Offline Detection Application**

In this alternative, the model will be included in the application. The detection process will be within the mobile phone resources, with no need for Wi-Fi to make predictions for the infected plant leaf image. The only thing that needs the use of the Wi-fi in this application is the other features like the map of the closest shop and the access to the community. Other than that, the process of detection and finding the treatment will be within the mobile application (No need to connect with the backend).

#### ***Advantages:***

- No connection to prediction: this process provides a great advantage to users who do not have a connection at any time or who have a bad connection to the internet.
- Portability: when the process of prediction is a standalone application that means there's no limitation on the area of the process.

#### ***Disadvantages:***

- lack of resources for the mobile application

- Extendibility: any development and changes to the model or the treatment tables lead to the need to download the new version of the model from the store

### **3.6.2 Online Detection Alternative**

In this alternative, the whole process is within the need of Wi-Fi. The system will be connected to the backend and the backend will do all entire services which include plant detection, finding the treatment, accessibility to the community, and providing the description of the plant.

Advantages:

- The power of server resources: the prediction within the connection with the server leads to more accuracy and more efficiently
- Scalability: no need to be up to date to use the new versions of the model

Disentangles:

- Wi-Fi limitation: cannot use any feature or service without the need for Wi-Fi.

### **3.6.3 Hybrid Approach Alternative**

In this alternative, we combine the 2 previous alternatives. By adding the ability to download the model and use it offline. This approach provides the capability to let the user choose whatever the prediction of the model is online or offline. So, the application will be online by default with the ability to download the model and make predictions offline.

## **3 . 7 S y s t e m A n a l y s i s a n d O p t i m i z a t i o n**

To ensure effective performance and a smooth user experience, a thorough system analysis and optimization procedure was done while creating the Plantie App. The primary domains of analysis and optimization are listed below:

### **3.7.1 Plant Disease Detection**

**Model Selection and Training:** For plant disease detection, after training using more than a CNN architecture, we chose the MobileNetV2 architecture because it achieves a good compromise between computing efficiency and accuracy, making it a good fit for mobile applications. Using Google Colab's powerful GPUs and pre-installed libraries, the model was trained on a dataset of plant photos and we collected these images from many sources. It contains 37 classes.

**Data Preprocessing:** Extensive image resizing, normalization, augmentation (rotations, flips, and color modifications), and making our classes balanced. This helps to improve model performance. These steps helped in reducing overfitting and improving the model's generalization capability.

**Hyperparameter tuning:** To maximize the number of epochs, batch size, and learning rate, we tuned the hyperparameters. We were able to determine the optimal settings for our dataset by combining grid search and random search approaches in Colab.

**Model Evaluation:** Metrics like accuracy, precision, recall, and F1-score were used to assess the model's performance. Cross-validation was used to make sure the model was reliable. With a 98.7% accuracy rate on the test set, the trained model indicated its effectiveness in detecting plant diseases. You can see our work in Colab: [Notebook Link](#).

#### **3.7.1.1 Plant Disease Datasets and Argumentation**

Figure 3.21 shows the number of images for each class, so that the dataset is not balanced, so employing the augmentation method on the dataset is have must process. After applying the augmentation, the range of images for each class is between (750-800). Also, we noticed that there are some classes is more than 800 images, so we should apply *under-sampling* for them to achieve the balance in the dataset. Note that the unbalanced effect rapidly on the accuracy of results, so this process of making the data balanced is required to achieve the best accuracy.

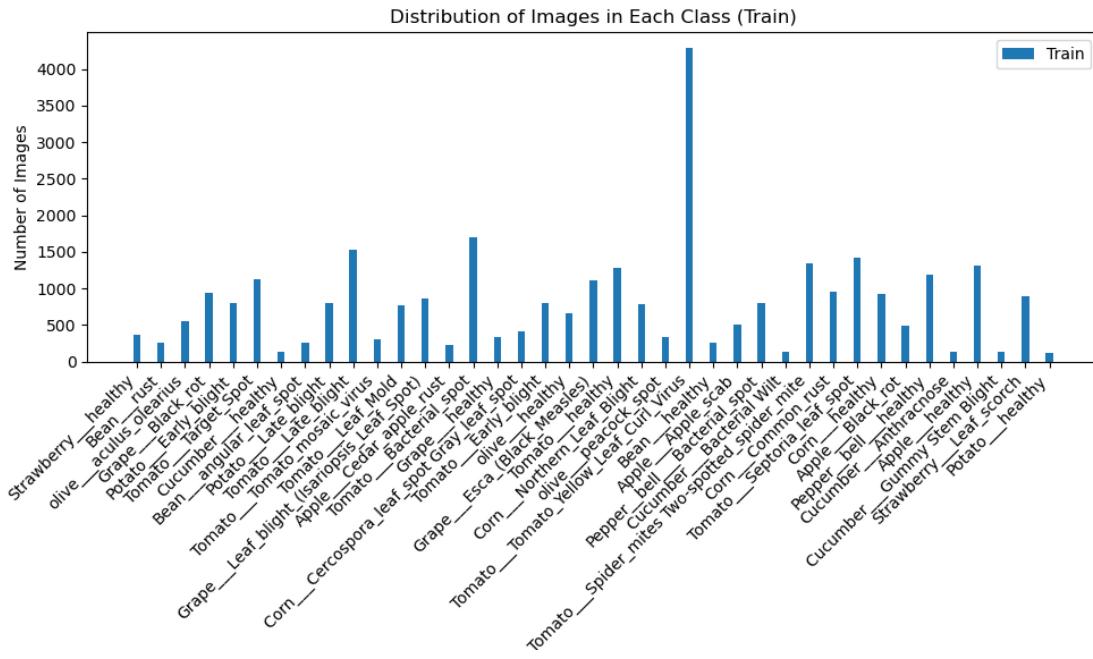


Figure 3-21 Distribution of images in each class

### 3.7.1.2 Models Training

We applied several deep learning models to show which is the best in our application. To assess a model's ability to generalize beyond training data, we calculate its accuracy score using a separate validation set (often 20% of the data). This score, shown for six models in Table 1, helps us compare their effectiveness and choose the best one for the specific task, avoiding models that simply memorize training data and fail on unseen examples. All models in the table below are recorded within 30 epochs.

Table 3 Models Classification Results

Models	CNN	ResNet	GoogleNet	DesnseNet	MobileNet	EfficientNet
Test Accuracy	88%	98.7%	97.9%	98.5%	97.8%	98%
Train Accuracy	90%	99.39%	98.3%	99.8%	97.95%	98.3%
Loss	36%	4.4%	6%	5.6%	8%	5%

As illustrated in Table 1, ResNet stands out by achieving the highest accuracy among the models tested. Specifically, it attains an impressive 98.7% accuracy on the test set and an even higher 99.39% accuracy on the training set. This demonstrates ResNet's superior performance and robustness in both training and evaluation phases. The following figure provides a visual representation of the confusion matrix for the

ResNet model, highlighting its precision in classifying the test data across various categories. But keep in mind that we need an *efficient model* that can be used with the mobile application and the mobile net archive high accuracy with low computational hardware usage which fits our system to use it within the mobile application. The MobileNetV2 achieves 98.7 in the test dataset within 50 epochs with a 4% loss.

### **3.7.2 Community**

Scalability: was taken into consideration when designing the community function, which enables users to post and engage with content. To manage massive amounts of user-generated material, we improved query efficiency and introduced effective database indexing. We use Firebase features like storage and a real-time database.

User involvement: We included elements like liking, disliking, and commenting to increase user involvement. To guarantee low latency and a responsive user interface.

### **3.7.3 Real-Time Weather**

Integration with Weather API: To deliver real-time weather updates depending on the user's current location, we integrated Open Weather API, a reputable weather API. To guarantee frequent updates and pertinent plant care advice, the data is quickly retrieved and processed.

Optimization for Mobile: The weather feature was optimized for mobile devices, ensuring that the data consumption is minimal, and the updates are quick, even on slower network connections.

### **3.7.4 Nearby Plant Stores**

Google Maps Integration: We developed a tool to find the closest plant stores by utilizing the Google Maps API. The integration was fine-tuned to yield precise outcomes while minimizing the frequency of API calls to minimize expenses and enhance efficiency.

User Experience: The easy design of the user interface makes it simple for users to locate and browse to neighboring plant retailers. This required making sure the map interacted smoothly and optimizing loading times.

### **3.7.5 Fertilizer Calculator**

Algorithm Efficiency: Based on user inputs, the fertilizer calculator tool uses simple algorithms to provide quick and accurate recommendations.

User Interface: The fertilizer calculator's user interface was created with ease of use in mind, featuring input fields that are easy to understand and prompt response.

## **3.8 Simulation and Experimental Test**

The main screen of the Plantie app incorporates important functionality into an intuitive UI. It has a weather sub-view with real-time weather updates and customized plant care recommendations based on the state of the weather. It also provides extensive details on a variety of plants, such as when to plant them, how much fertilizer they require, and how to prevent disease. Users are guaranteed simple access to the necessary resources and information for efficient plant care due to this single hub.

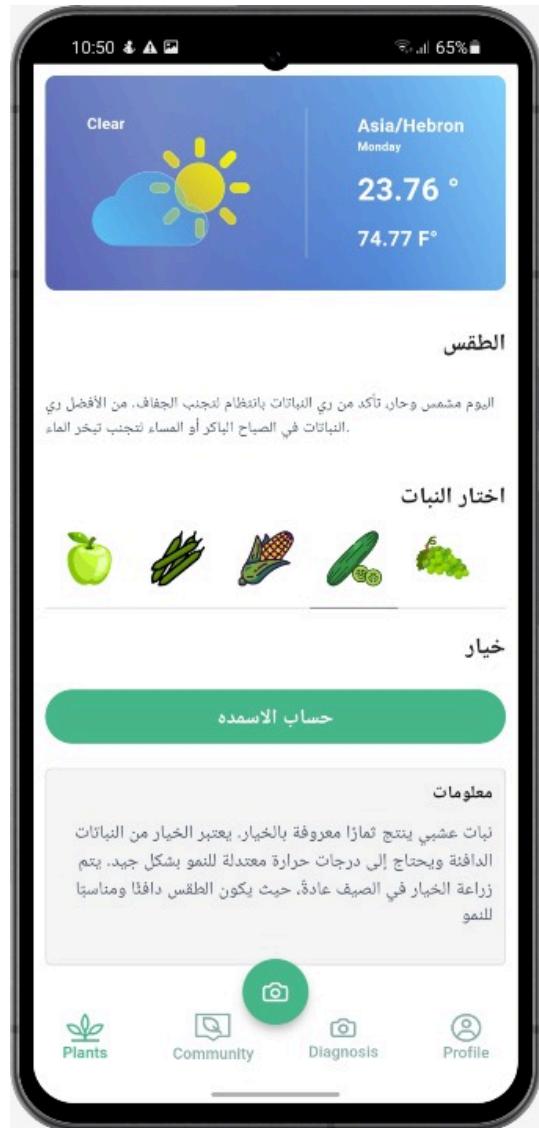


Figure 3-22 The Plantie app's main view, showcases weather updates, plant care advice, and detailed plant information.

### 3.8.1 Detection View:

Dataset Simulation: Using methods like flipping, rotating, and color correction, we enhanced our pre-existing plant disease photos to produce a simulated dataset. This helped evaluate the model's stability and ability to generalize to previously untested data.

The learning rate that has been used in the paper changes during the training process, starting with a 0.0001 learning rate, and will increase or decrease based on the change in the validation accuracy. The following formula is the definition of accuracy:

$$\text{Accuracy} = \frac{TP + TN}{TN + TP + FN + FP}$$

Where TP = true positives, TN = true negatives, FP = false positives, and FN = false negatives. While precision quantifies the number of positive class predictions that belong to the positive class, recall quantifies the number of positive class predictions made out of all positive classes in the dataset, The formula of the F1-score is shown by the following equation:

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1 - Score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

**Model Testing:** Using a different validation set that wasn't used for training, the trained MobileNetV2 model was tested. To simulate real-world situations, this set featured various plant images with different health symbols.

After more than 50 epochs we got the final results that were recorded on the test data:

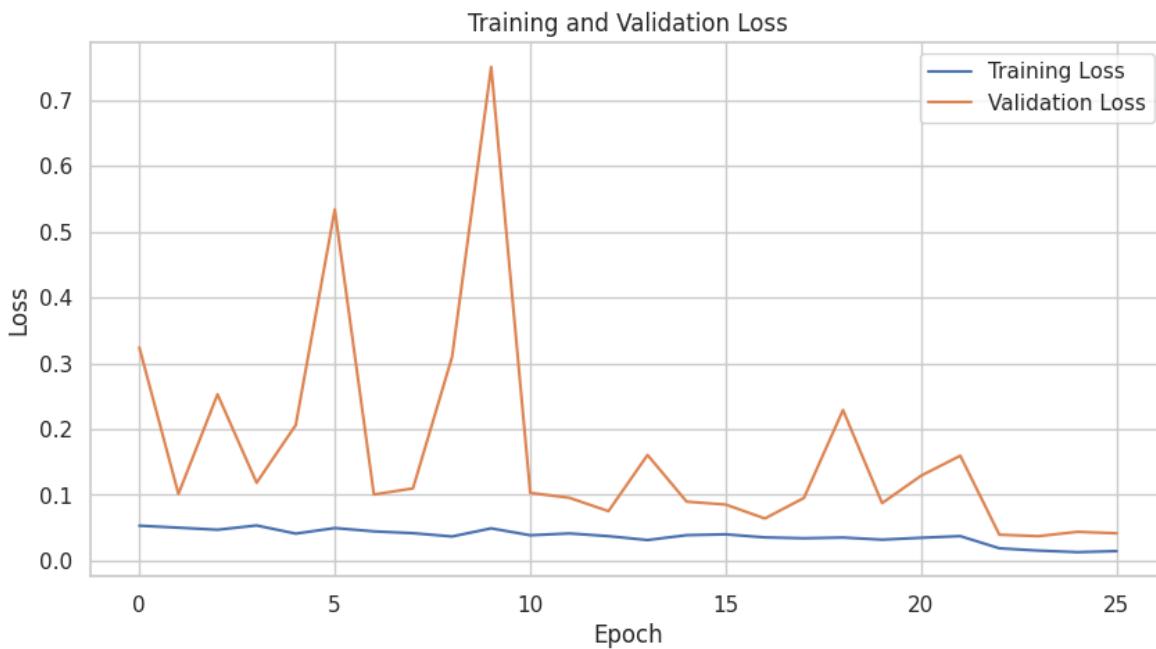
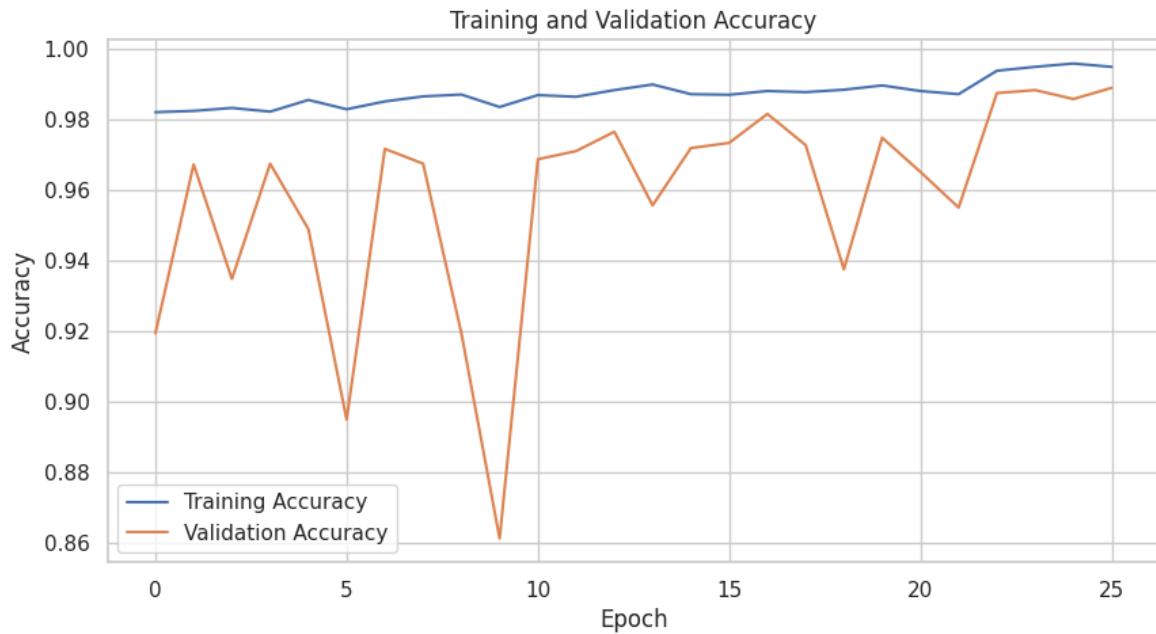
*Table 4 MobileNetV2 accuracy result*

MobileNetv2 accuracy result	
Test Loss	4.3%
Test Accuracy	98.78%

### 3.8.1.1 Visualizations

We use charts and the confusion matrix to visualize our model accuracy and loss, these charts show the relations between the validation result and the training results. However, In the confusion matrix, we will represent the prediction result of the test dataset, and the matrix shows all classes and their predictions as a matrix. The main diagonal refers to the number of correct predictions, otherwise, the prediction is not correct.

After 25 epochs we record the accuracy and loss of the training and validation datasets, as we see in the figures below, the validation and accuracy increase. The validation dropdown was in some cases but in the end, it became close to the training accuracy. Also, we have the loss chart of validation and training which in earlier epochs the difference between the two losses was too high but when reaching to the last epochs we reached to the current accuracy.



## Confusion matrix (contains 37 classes):

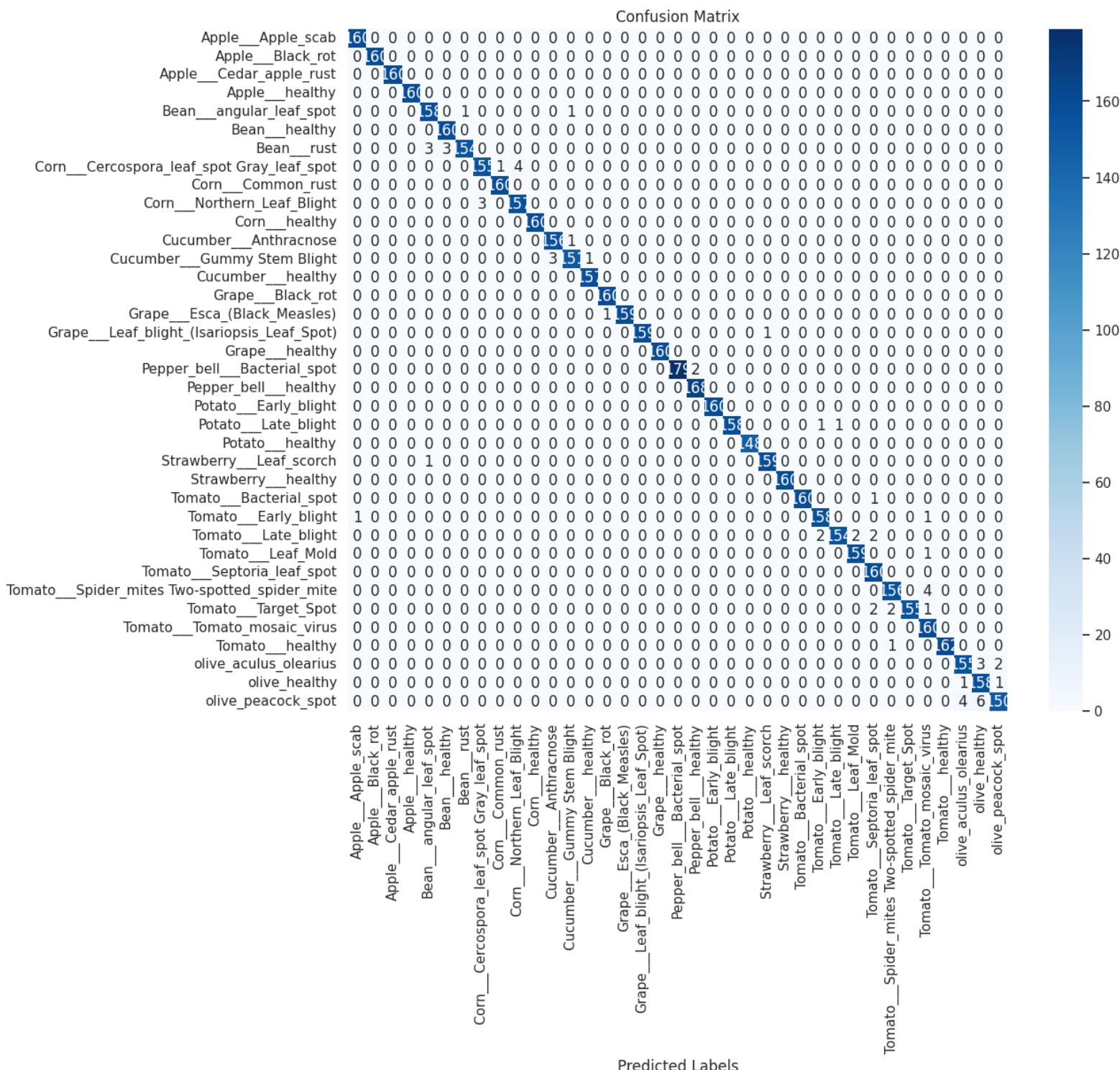


Table 5 Classification Report of MobileNetV2

	PRECISION	RECALL	F1-SCORE	SUPPORT
APPLE__APPLE_SCAB	0.99	0.99	0.99	160
APPLE__BLACK_ROT	0.99	1.00	1.00	160
APPLE__CEDAR_APPLE_RUST	1.00	0.99	1.00	160
APPLE__HEALTHY	0.98	1.00	0.99	160
BEAN__ANGULAR_LEAF_SPOT	0.99	1.00	1.00	160
BEAN__HEALTHY	0.99	0.95	0.97	160
BEAN__RUST	0.96	0.98	0.97	160
CORN__CERCOSPORA_LEAF_SPOT_GRAY_LEAF_SPOT	0.98	0.92	0.95	160
CORN__COMMON_RUST	0.99	1.00	1.00	160
CORN__NORTHERN_LEAF_BLIGHT	0.95	0.98	0.96	160
CORN__HEALTHY	1.00	1.00	1.00	160
CUCUMBER__ANTHRACNOSE	0.95	1.00	0.98	157
CUCUMBER__GUMMY_STEM_BLIGHT	0.99	0.95	0.97	155
CUCUMBER__HEALTHY	0.99	1.00	1.00	157
GRAPE__BLACK_ROT	1.00	1.00	1.00	160
GRAPE__ESCA_(BLACK_MEASLES)	1.00	1.00	1.00	160
GRAPE__LEAF_BLIGHT_(ISARIOPSIS_LEAF_SPOT)	1.00	1.00	1.00	160
GRAPE__HEALTHY	1.00	0.99	1.00	160
PEPPER_BELL__BACTERIAL_SPOT	0.96	1.00	0.98	181
PEPPER_BELL__HEALTHY	1.00	0.96	0.98	168
POTATO__EARLY_BLIGHT	0.99	1.00	1.00	160
POTATO__LATE_BLIGHT	0.99	0.98	0.99	160
POTATO__HEALTHY	1.00	0.99	1.00	148
STRAWBERRY__LEAF_SCORCH	1.00	0.99	0.99	160
STRAWBERRY__HEALTHY	1.00	0.99	1.00	160
TOMATO__BACTERIAL_SPOT	1.00	0.93	0.96	161
TOMATO__EARLY_BLIGHT	0.96	0.97	0.97	160
TOMATO__LATE_BLIGHT	1.00	0.99	0.99	160
TOMATO__LEAF_MOLD	0.99	1.00	0.99	160
TOMATO__SEPTORIA_LEAF_SPOT	0.92	1.00	0.96	160
TOMATO__SPIDER_MITES_TWO-SPOTTED_SPIDER_MITE	0.99	0.97	0.98	160
TOMATO__TARGET_SPOT	0.96	1.00	0.98	160
TOMATO__TOMATO_MOSAIC_VIRUS	0.99	1.00	1.00	160
TOMATO__HEALTHY	1.00	0.99	0.99	163
OLIVE_ACULUS_OLEARIUS	0.99	0.94	0.96	160
OLIVE_HEALTHY	0.98	0.98	0.98	160
OLIVE_PEACOCK_SPOT	0.95	0.98	0.96	160

Test the model in real data and print the top 3 predictions:



**Top 3 Predictions:**

Tomato\_Early\_blight: 0.987829327583313  
Tomato\_Target\_Spot: 0.012168203480541706  
Potato\_Early\_blight: 2.369177309446968e-06

The next step was to transform our MobileNetV2 model into a format appropriate for mobile apps and incorporate it into our Plantie app after it had been successfully trained and validated for plant disease diagnosis. To make sure the model could function well on mobile devices and deliver predictions in real time, this procedure involved multiple processes.

### 3.8.1.2 Conversion to TensorFlow Lite:

Method: To make the trained MobileNetV2 model more suitable for mobile and embedded devices, we transformed it into the TensorFlow Lite (TFLite) format. To decrease the model's size

and increase inference speed without noticeably sacrificing accuracy, this conversion required quantizing the model.

**Benefits:** We could execute the model directly on customers' mobile devices by using TFLite, which ensured prompt and accurate disease diagnosis.

#### **3.8.1.3 Assimilation into Plantie:**

**Implementation:** TensorFlow Lite's Android and iOS libraries included the TFLite model in the Plantie app. To manage picture input, model inference, and result display, we implemented the required APIs.

**Optimization:** To ensure seamless and effective operation on a variety of devices, we reduced the computing load during inference to optimize the app's performance.

#### **3.8.1.4 Making Predictions**

##### **User Interaction:**

Process: Using the Plantie app, users can snap or upload a picture of their plant. After that, the image is preprocessed to conform to the TFLite model's input specifications.

Inference: The model receives the preprocessed image and produces predictions indicating the kind and presence of plant disease.

##### **Displaying Results:**

Disease Detection: The app displays the detected disease along with a confidence score, providing users with a clear understanding of the diagnosis.

Treatment Recommendations: The app offers detailed treatment recommendations based on the identified condition, assisting users in properly managing and caring for their plants.

##### **User Experience and Feedback**

##### **Real-Time Testing:**

Deployment: A group of beta testers received the revised Plantie app with integrated disease detection. These testers offered useful feedback on the functionality of the model as well as the general user experience.

Results: The app's disease detection feature was praised by users for being quick and accurate, making it a useful tool for taking care of plants.

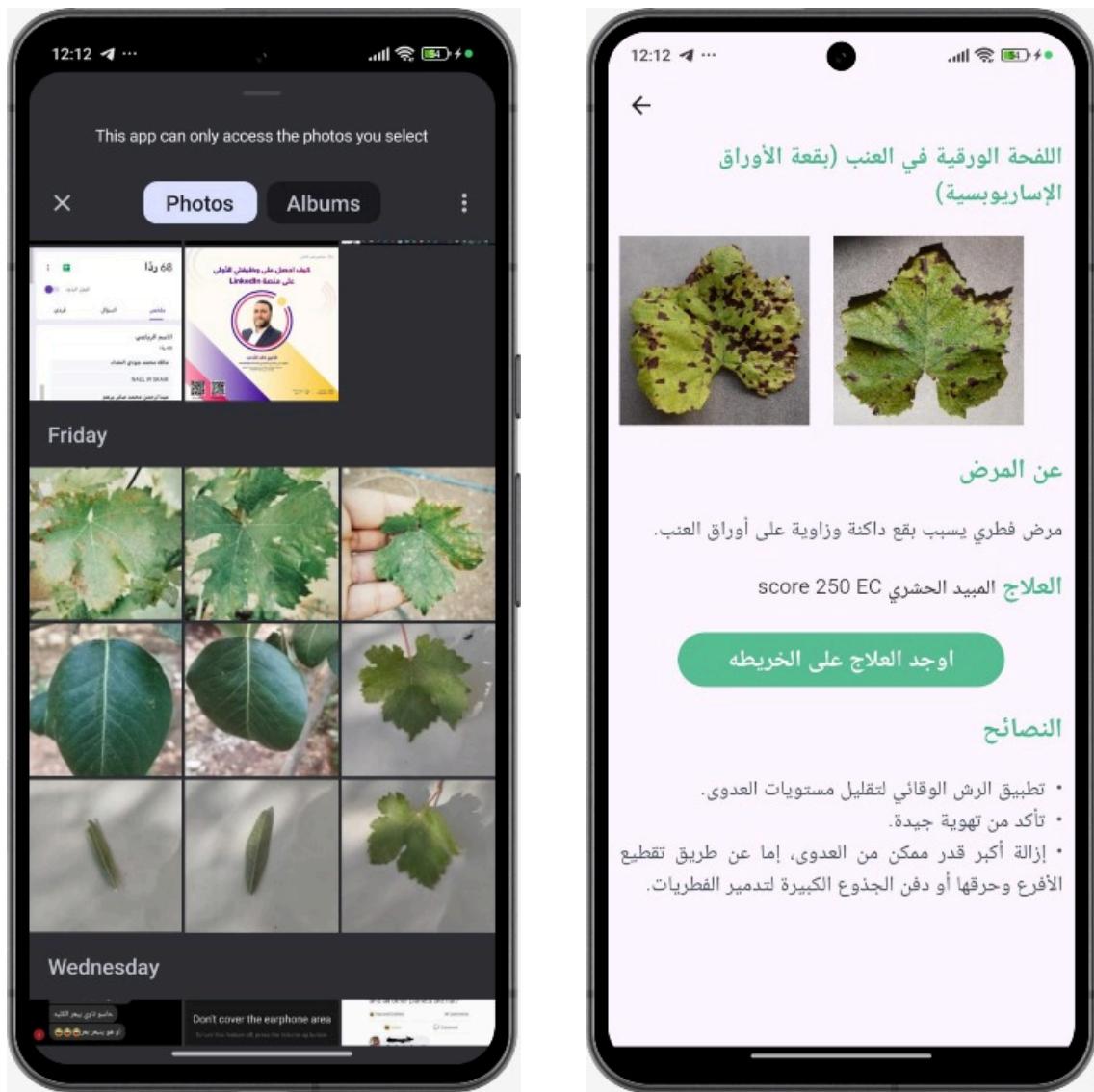


Figure 3-23 The Plantie app detecting a disease from an uploaded plant image

### 3.8.2 Fertilizer Calculator:

The fertilizer calculator feature of the Plantie app simplifies plant care by providing individualized fertilizer recommendations based on the types of plants, and the area of the plant. Users may quickly establish the ideal fertilizer dosage for their plants by combining critical nutrients (NPK) and considering the growing area.

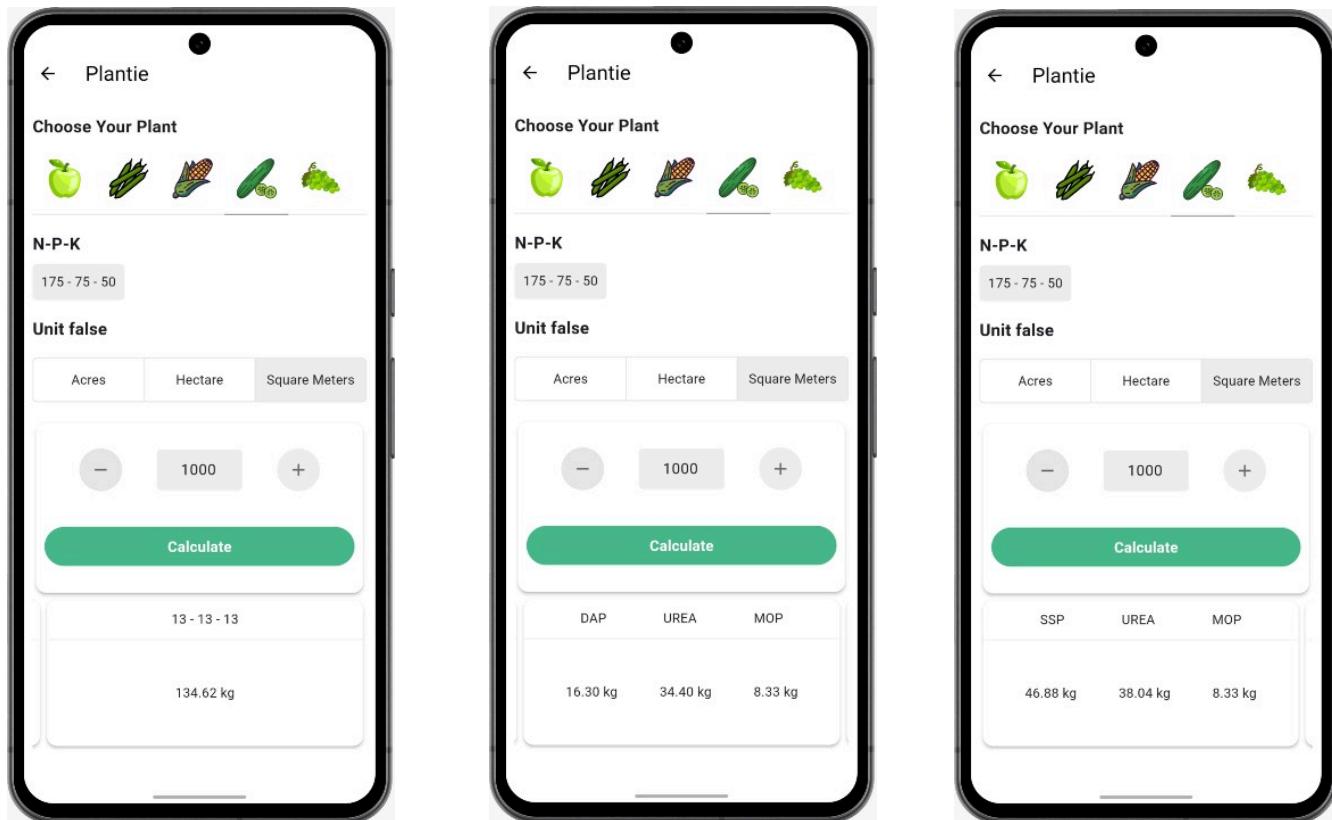


Figure 3-24 The Plantie app's fertilizer calculator in action, recommending the ideal fertilizer dosage based on NPK values and

### 3.8.3 Community:

The community aspect of the Plantie app encourages farmers and plant enthusiasts to work together and share knowledge. By asking questions, offering advice, and responding to each other's

messages, users may build a lively and encouraging community. This feature improves user interaction and offers insightful information about how to take care of plants.

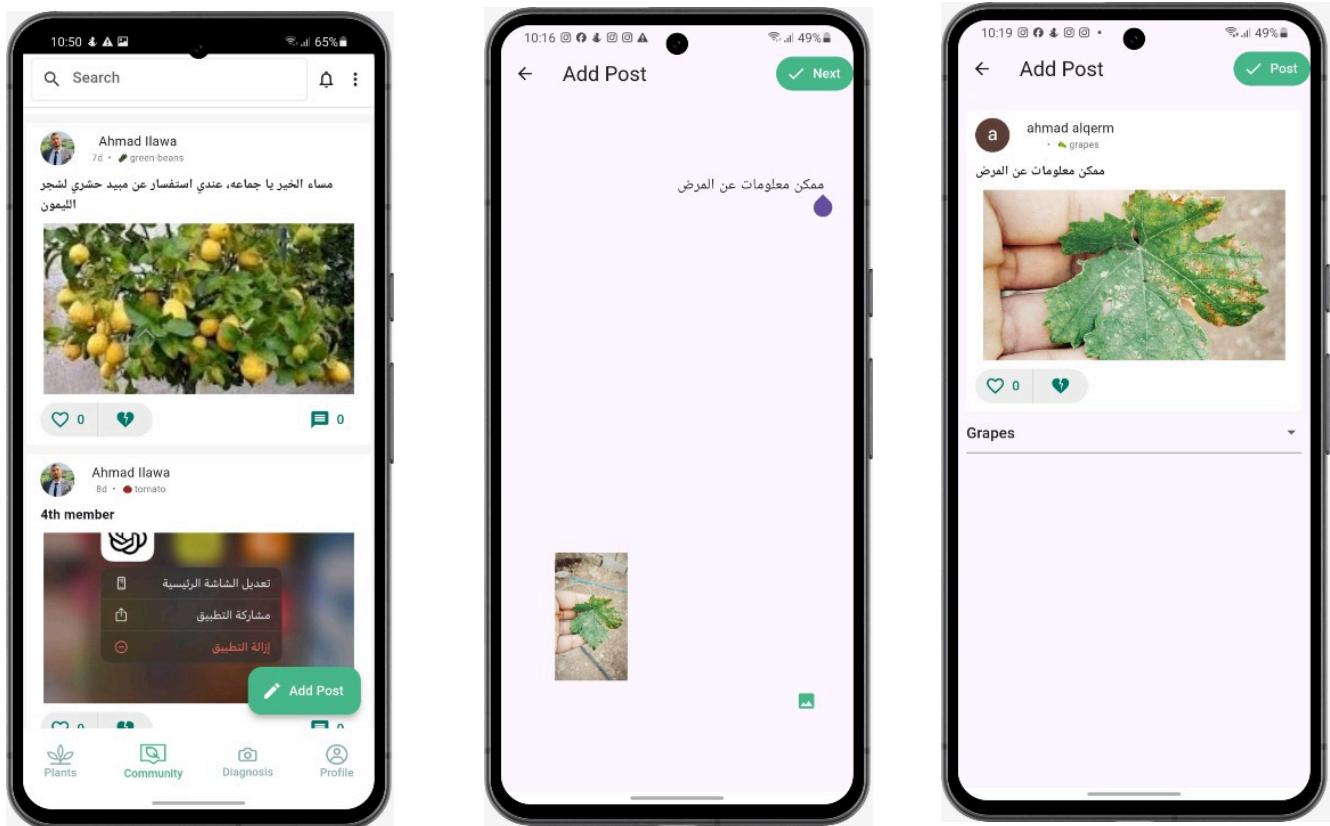


Figure 3-25 The Plantie app's community feature, showcasing user interactions and shared plant care tips.

#### 3.8.4 Nearby Plant Stores:

With Google Maps integration, the Plantie app's "nearby plant stores" function makes it simple for users to find the closest plant retailers. The application retrieves the user's current location and displays a list of stores that are in the vicinity, along with contact details and directions. With the help of this practical function, consumers can locate and visit stores for their gardening needs more quickly, saving time and effort. Accuracy and dependability are guaranteed by the smooth connection with Google Maps.

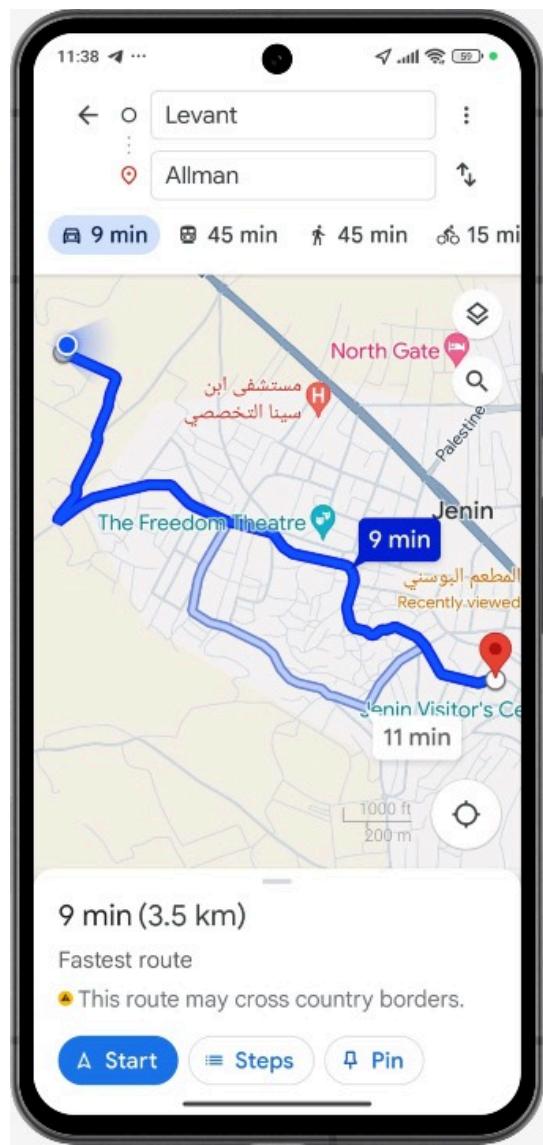


Figure 3-26 The Plantie app's fertilizer calculator in action, recommending the ideal fertilizer dosage based on NPK values and cultivation area for a selected plant type

## 4 CHAPTER 4: RESULT AND DISCUSSIONS

### 4.1 Result

The Visual Result refers to the UI and the prototype of the application, which show the clear initial specification. The UI design is an efficient way to describe what the application looks like and how the application must be done. The following shows the Main Pages/Views of the “Plantie”.

#### 4.1.1 Authentication Views

The authentication system is important to be part of the Plantie community. The authentication system gives you the ability to register either manually by using email, password, and username or through social media accounts like Gmail and Facebook. Figure 4-1 shows the login and register views with the landing view:

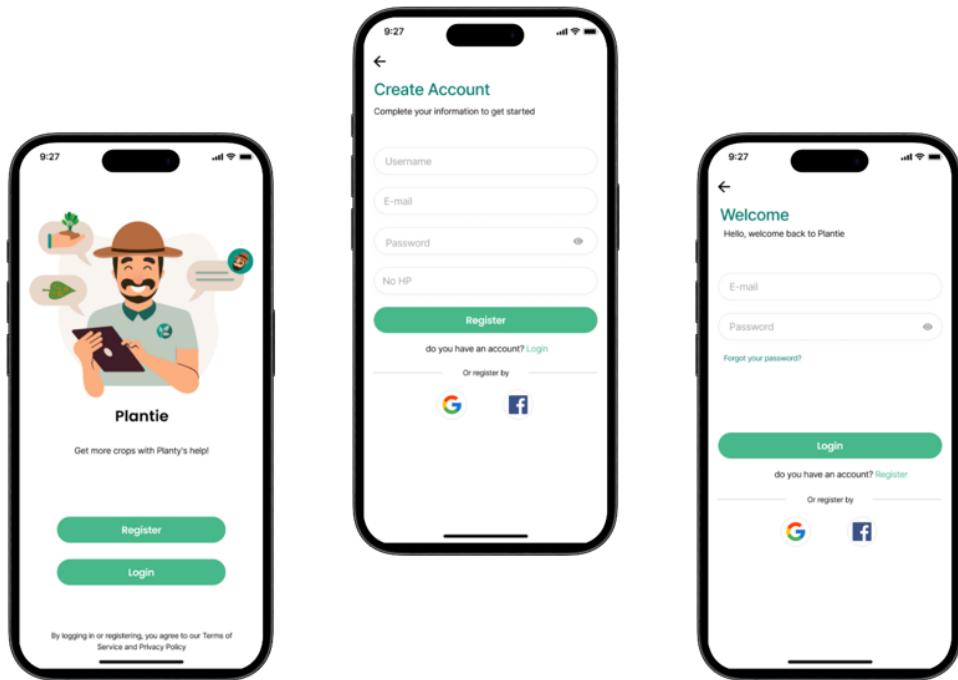


Figure 4-1 Authentication Views

#### 4.1.2 Plants/Home View

The main view of Plantie includes the weather system section which provides the weather conditions, and the temperature based on the user location, and the lower part of the section gives suggestions. The second section of the view includes the horizontally scrollable list which contains the plants with its description and some details like the amount of water that plants need and when the plant grows. Within this section, there's a button that navigates you to the fertilizer calculator as shown at the left of figure 4-2.

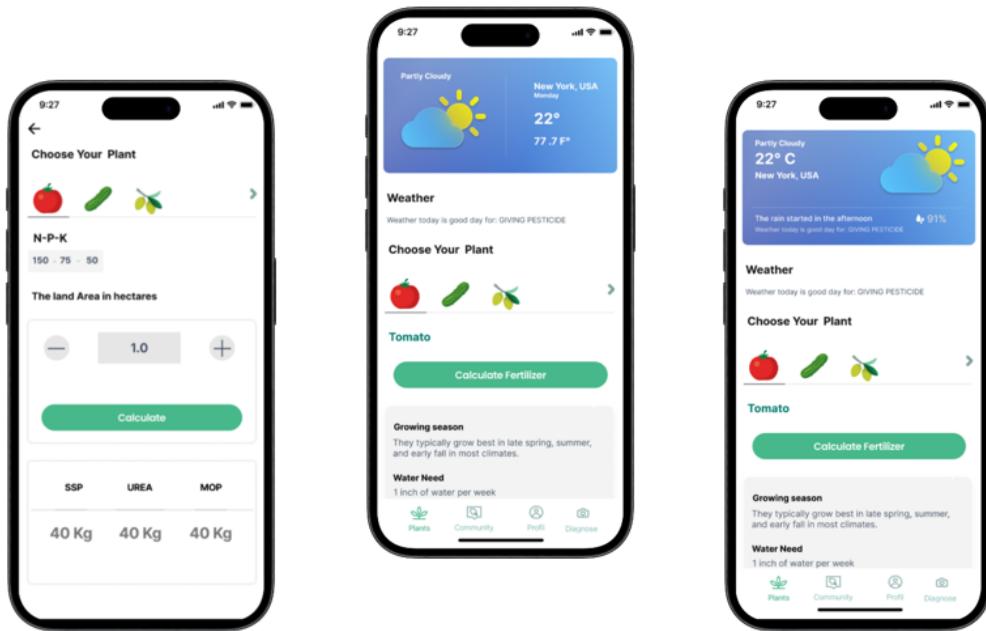
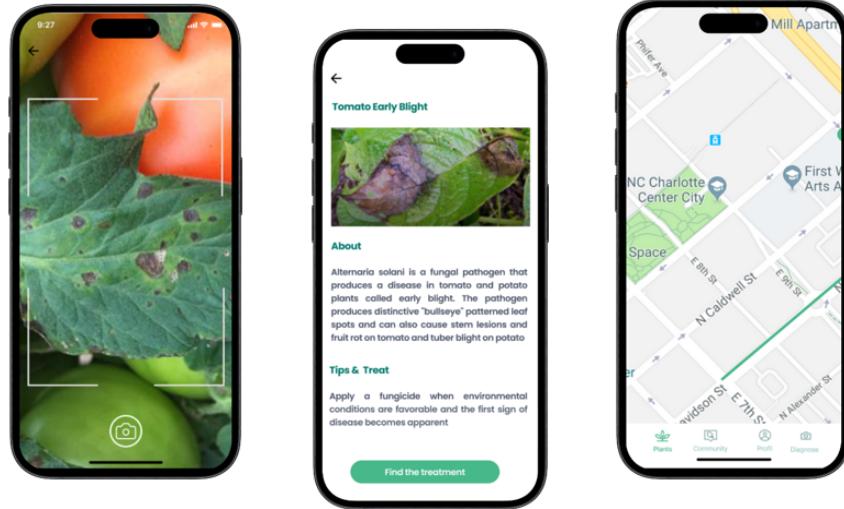


Figure 4-2 Plants/Home View

#### 4.1.3 Disease Detection Views

The disease detection view consists of 3 processes, plant disease prediction, treatment/advice, and the process of finding medicine if necessary. Once when you press on the camera icon in the navigation bottom bar, it will navigate you to the left screen in the figure below to allow you to take a picture or load a picture. After that, the picture will be sent to the server to be processed and the server will retrieve the prediction to show the result as shown at the screen at the middle. This

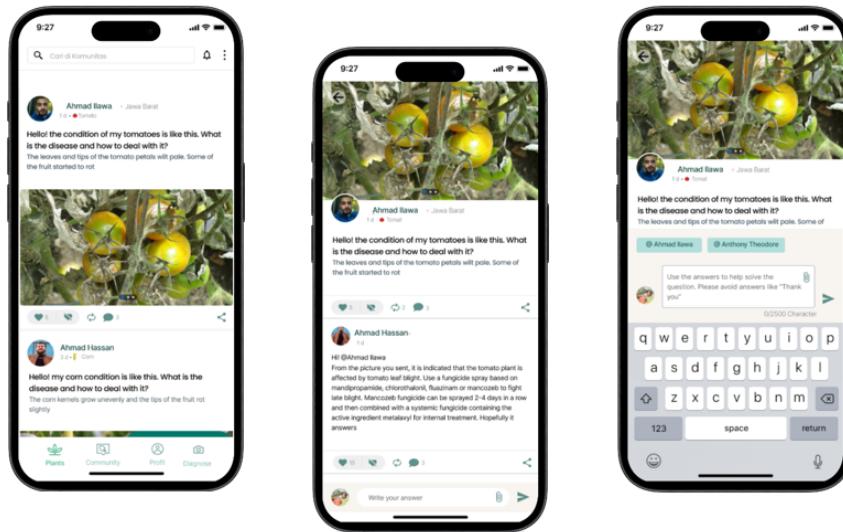
screen includes the disease, its description, the tip or how-to threat, and a button that can navigate you to a map to locate the nearest plant shop.



*Figure 4-3 Disease Detection View*

#### 4.1.4 Community Views

As shown in the figure below, the 3 views show the community view and how the process of posting, and commenting in the post. Figure 4-4 shows the some of Community views:



*Figure 4-4 Community View*

## **4 . 2 D i s c u s s i o n s**

Our Plantie mobile app project was completed successfully, offering a wide range of features that make it an all-inclusive tool for agricultural support and identifying plant diseases. The main features of our project are listed here, together with the conclusions and learnings from the phases of creation and testing.

### **4.2.1 Plant Disease Detection:**

**Functionality:** Plant disease diagnosis using a MobileNetV2 model is the main function of Plantie. Users may snap pictures of their plants with the app, and the model will identify any problems and suggest treatments.

**Testing:** Field testing and a validation dataset were used to test the model thoroughly. The model proved to be highly accurate in identifying a wide range of plant diseases during testing and offered reliable treatment recommendations.

**Result:** Users could more successfully maintain plant health because they found the plant disease detection feature to be accurate and useful.

### **4.2.2 Community:**

**Functionality:** Users can post and exchange information, engage in conversation, and offer advice and experiences using the community feature.

**Testing:** To make sure the community feature could handle large amounts of user activity without experiencing performance concerns, load testing, and beta testing were carried out.

**Result:** The product was well accepted, and users actively participated in conversations and contributed insightful comments, creating a community that was encouraging.

#### **4.2.3 Real-Time Weather Updates:**

**Functionality:** The app delivers plant care advice specific to current weather conditions and provides real-time weather updates based on the user's location.

**Testing:** To verify the precision and promptness of weather updates, simulations were run. The applicability of the plant care advice was confirmed by field validation.

**Result:** Users thought the plant care advice was useful and helpful for preserving the health of their plants, and they valued the prompt weather updates.

#### **4.2.4 Nearby Plant Stores:**

**Functionality:** The software assists users in finding the closest plant stores by integrating with Google Maps.

**Testing:** To guarantee precise location services and seamless user interface interactions, simulations, and user testing were carried out.

**Result:** By making it easier for consumers to locate local plant stores, the innovation improved the convenience of buying items linked to plants.

#### **4.2.5 Fertilizer Calculator:**

**Functionality:** Based on user inputs, the fertilizer calculator makes recommendations to help users figure out how much fertilizer is best for their plants.

**Testing:** Detailed simulations verified the suggestions were accurate, and user input helped to improve the feature.

**Result:** Users reported that the fertilizer calculator was accurate and simple to use, which improved plant maintenance and maximized fertilizer usage.

## 5 CHAPTER 5: PROJECT MANAGEMENT

### 5.1 Tasks, Schedule, and Milestones

The figures below show the project schedule, how the tasks are being done, and how we divided our time and made full use of our time, on another hand, the right side includes a Gantt chart that describes the process we have done.

ID	Task Mode	Task Name	Duration	Start	Finish	Predecessors	Resource Names	33 9   12
1	Plantie		94 days	Mon 10/16/23	Thu 2/22/24			
2		1. initiation	17 days	Mon 10/16/23	Tue 11/7/23			
3		1.1 Define team members and roles	3 days	Mon 10/16/23	Wed 10/18/23		ahmad llawa	
4		1.2 Kick-off Meeting	2 days	Thu 10/19/23	Fri 10/20/23	3	Ahmad Qassam	
5		1.3 Define Project	9 days	Mon 10/23/23	Thu 11/2/23	4	Ahmad Qassam	
6		1.3.1 Information Gathering	8 days	Mon 10/23/23	Wed 11/1/23		Ahmad Qassam	
7		1.3.2 Get Consultation	1 day	Thu 11/2/23	Thu 11/2/23	6	Ahmad Qassam	
8		1.4 Develop Project Charter	1 day	Mon 11/6/23	Mon 11/6/23	5	ahmad llawa	
9		1.5 Project Approve	1 day	Tue 11/7/23	Tue 11/7/23	8	ahmad llawa	
10		2. Planning	16 days	Sun 11/5/23	Mon 11/27/23	2		
11		2.1 Develop Project Plan	4 days	Sun 11/5/23	Wed 11/8/23	2		
12		2.1.1 Define Project Objecti	1 day	Tue 11/7/23	Tue 11/7/23		Mohammad Nassar	
13		2.1.2 Define Project Specification	1 day	Wed 11/8/23	Wed 11/8/23	12	Mohammad Nassar	
14		2.1.3 Define Project Deliverables	1 day	Wed 11/8/23	Wed 11/8/23	12	Ahmad Qassam	
15		2.2 Budget Plan	4 days	Wed 11/8/23	Mon 11/13/23	14		
16		2.2.1 Specify the Possible Resources	3 days	Wed 11/8/23	Fri 11/10/23		ahmad llawa	
17		2.2.2 Software Budget & Costs	1 day	Mon 11/13/23	Mon 11/13/23	16	Ahmad Qassam	
18		2.3 Develop Risk Plan	1 day	Tue 11/14/23	Tue 11/14/23	15	Mohammad Nassar	
Project: Project1.mpp Date: Thu 2/8/24		Task		Inactive Summary		External Tasks		
		Split		Manual Task		External Milestone		
		Milestone	◆	Duration-only		Deadline	▼	
		Summary		Manual Summary Rollup		Progress		
		Project Summary		Manual Summary		Manual Progress		
		Inactive Task		Start-only				
		Inactive Milestone	◇	Finish-only	□			

Page 1

Figure 5-1 Project Schedule with Gantt-Chart 1

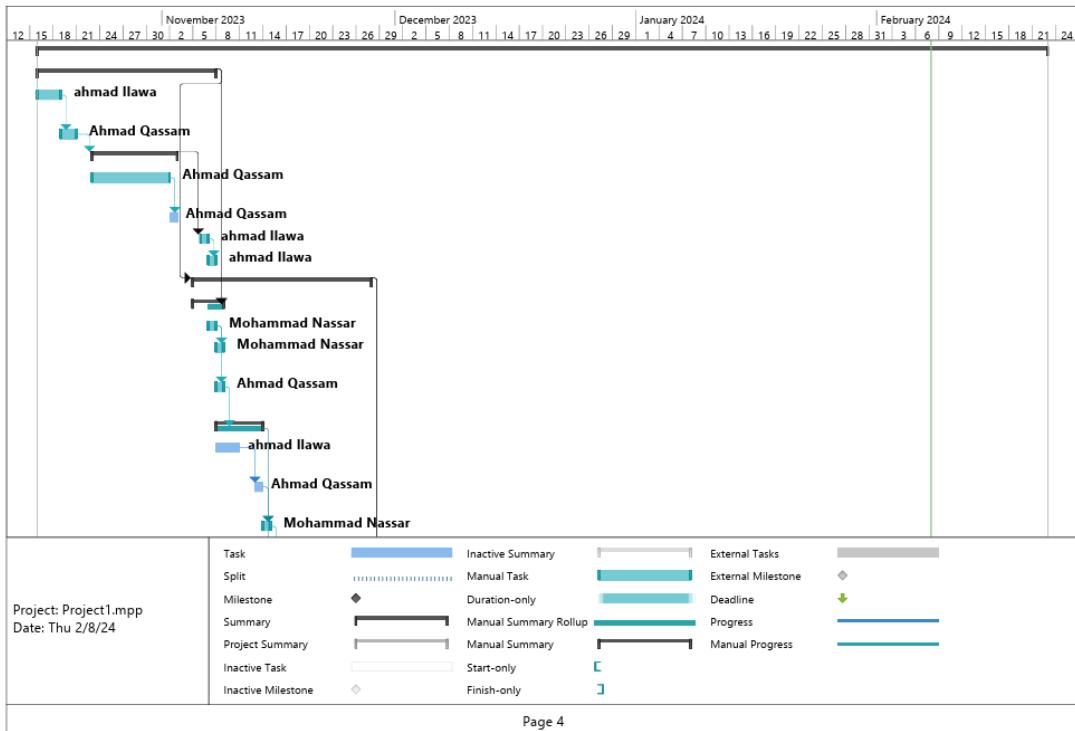


Figure 5-2 Project Schedule with Gantt-Chart 2

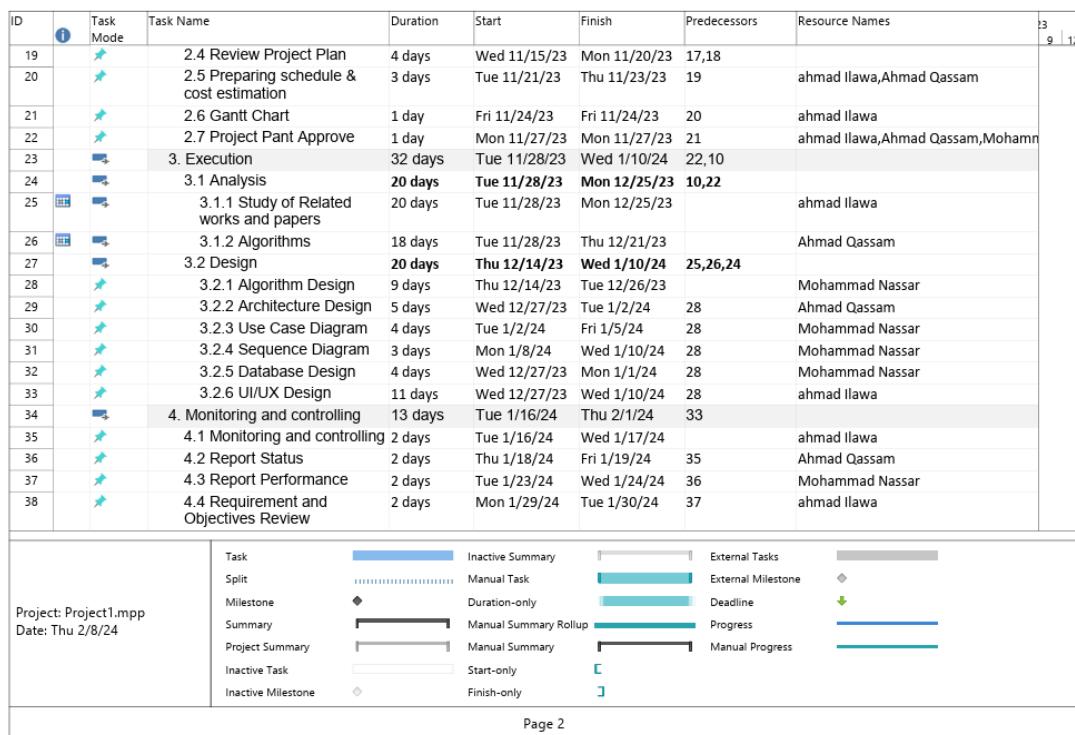


Figure 5-3 Project Schedule with Gantt-Chart 3

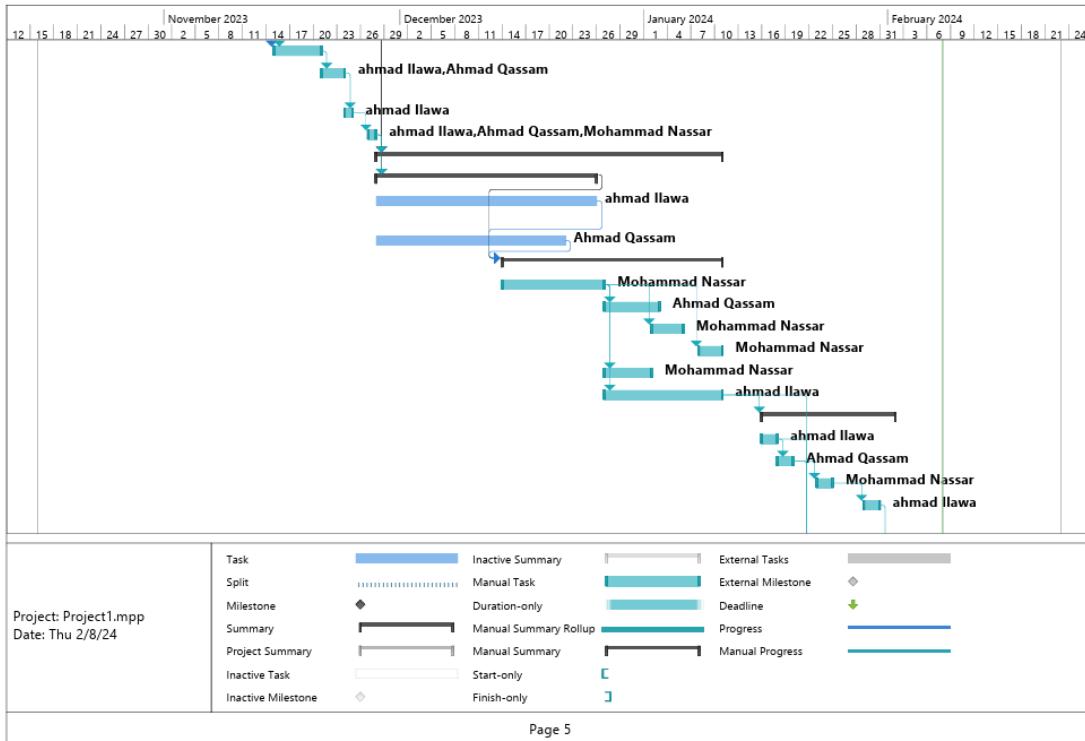


Figure 5-4 Project Schedule with Gantt-Chart 4

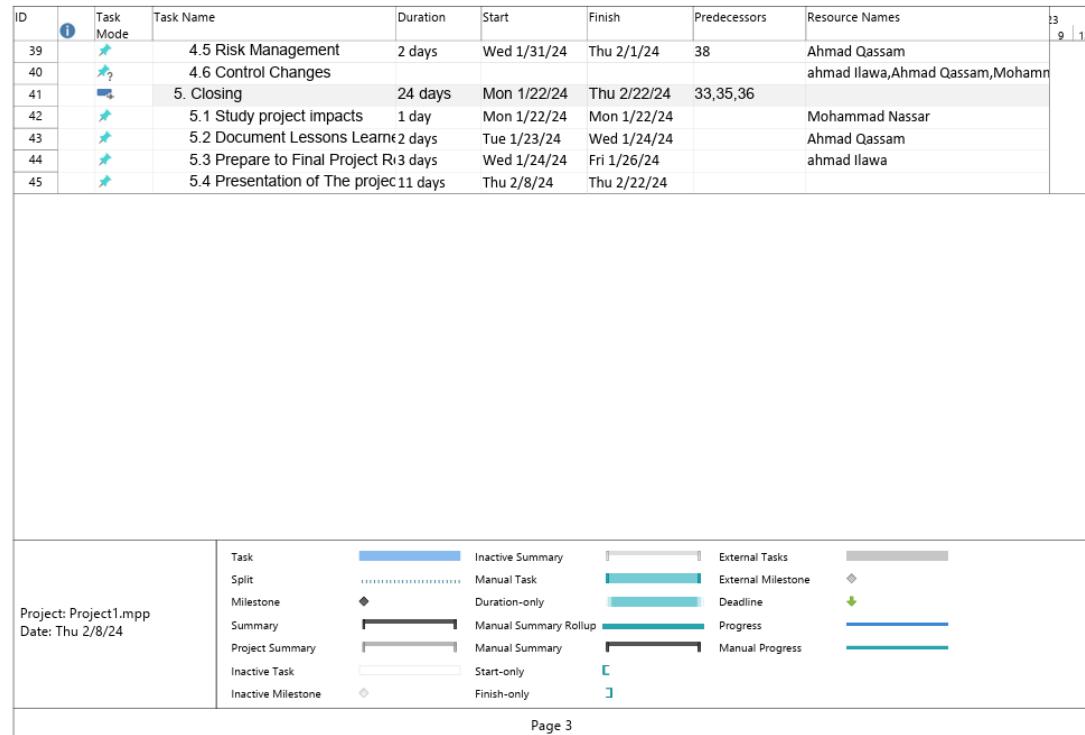
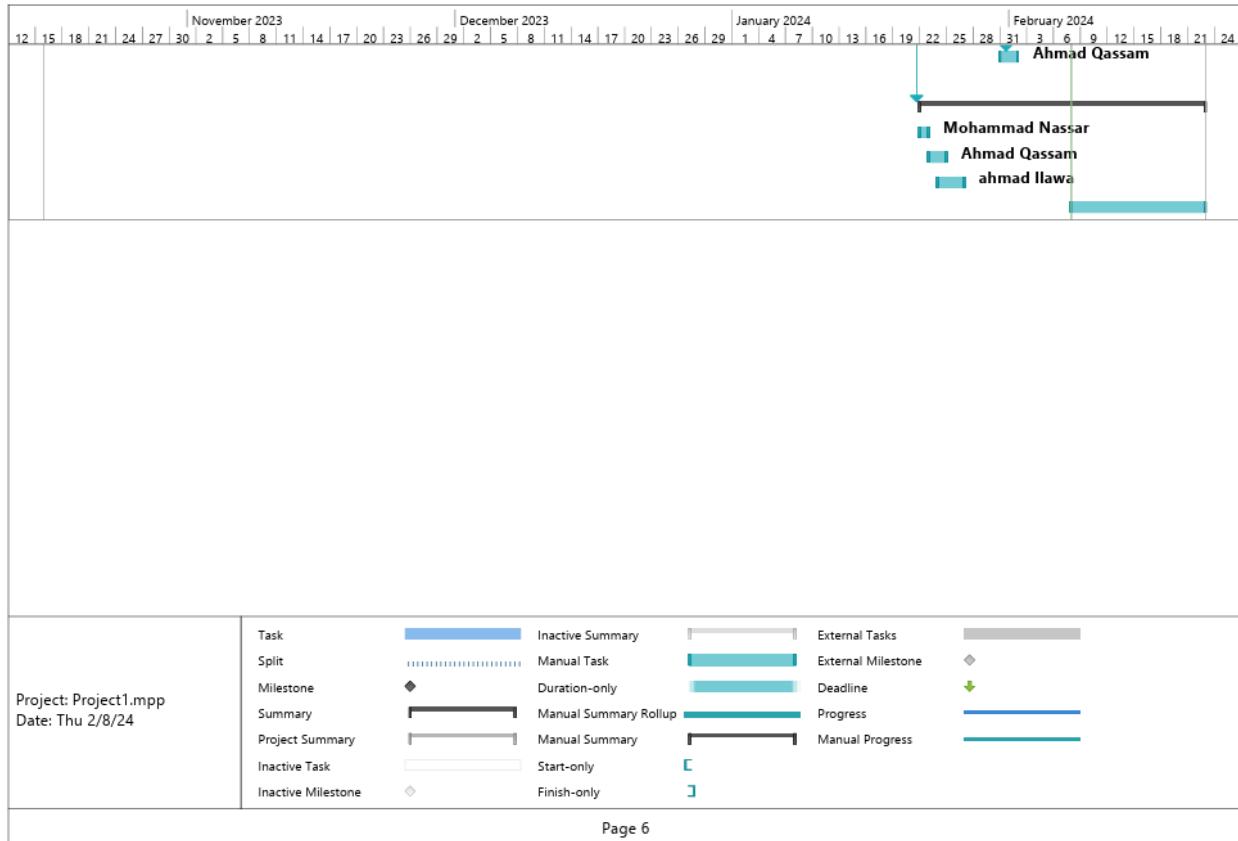


Figure 5-5 Project Schedule with Gantt-Chart 5



*Figure 5-6 Project Schedule with Gantt-Chart 6*

## 5.2 Resources and Cost Management

The table below shows the estimated cost of the project as a whole. All of these numbers are estimates and personal judgments and are not 100% accurate.

*Table 6 Resources and Cost Management*

	# Units/Hrs.	Cost/Unit/Hr.	Subtotals	WBS Level 1 Total	% of Total
WBS Items					
1. Project Management				\$28,700	32.3%
1.1 Project manager	100 hours	\$150	\$15,000		
1.2 Project team members	235 hours	\$40	\$9,400		

Contractors (10% of software development and testing)			\$4,300		
2. Hardware				\$6,900	0.7%
2.1 Handheld devices	30 devices	\$130	\$3,900		
2.2 Servers	2 servers	\$1500	\$3,000		
3. Software				\$40,000	45%
3.1 Licensed Software		\$1300	\$1300		
3.2 Software development*			\$38,000		
4. Testing (10% of total hardware and software costs)			\$4,690	\$4,690	0.5%
5. Training and Support				\$15,400	17.3%
5.1 Trainee cost	100 Hours	\$60	\$6000		
5.2 Travel cost	2	\$70	\$140		
5.3 Project team members	463	\$20	\$9,260		
Total project cost estimate				\$88,790	

### 5 . 3 L e s s o n s   L e a r n e d

During the project, we acquired valuable skills such as time management, task delegation among team members, and monitoring the project plan. We also came to understand the importance of clear requirements to prevent unnecessary changes and delays in project completion. Furthermore, we gained knowledge on new technologies in the field of Deep Learning, specifically CNN, and their workings.

## **6 CHAPTER 6: IMPACT OF ENGINEERING**

The "Plantie" smartphone app, which is intended to help farmers detect plant diseases, has the potential to have a big impact on a lot of different areas of agriculture and society. Using cloud-based model storage and a deep learning algorithm based on CNN, the app seeks to transform plant disease management and farmer-to-farmer communication.

### **6.1 Economical, Societal and Global**

**Economic Impact:** Farmers can benefit financially from the app in a big way by increasing agricultural productivity, reducing crop loss, and saving money by using targeted disease management. The program provides farmers with trustworthy information on disease diagnosis and management, enabling them to make well-informed decisions that can lead to increased productivity and profitability. The app's affordability and accessibility also play a part in its economic impact. It removes financial obstacles to entry by being a free and easy-to-use tool, guaranteeing that farmers can receive vital disease management knowledge and improve their farming methods regardless of their financial situation.

**Social Impact:** By giving farmers the tools to make knowledgeable decisions about their crops, the "Plantie" app can empower them and possibly enhance their standard of living and food security. Farmers may communicate, exchange expertise, and aid one another through the app's community feature, which fosters a sense of belonging and teamwork.

**Global Impact:** The "Plantie" app demonstrates significant global impact by fostering collaboration and knowledge exchange among farmers worldwide. Through the app's platform for exchanging knowledge and creative approaches, it adds to an international network of agricultural resilience. Its focus on environmentally friendly agricultural methods also helps to conserve the environment and reduces the effects of issues such as plant diseases and climate change. The app becomes an engine for positive change on a worldwide scale, supporting food security and resilience in the face of changing agricultural landscapes, through technology transfer and farmer empowerment.

## **6 . 2 E n v i r o n m e n t a l**

Environmental Impact: By decreasing the needless use of pesticides through early illness diagnosis, the "Plantie" app can help maintain a sustainable ecosystem. Early disease detection allows farmers to control diseases with customized treatments, minimizing the need for environmentally hazardous broad-spectrum pesticides.

## **6 . 3 O t h e r I s s u e s**

Assurance of Health and Food Safety: The "Plantie" app is essential for guaranteeing the safety and health of food that is produced. The software helps produce healthier crops by enabling farmers to diagnose and control diseases better, which lowers the possibility of hazardous toxins entering the food supply chain. This focus on food safety is online with customer expectations and international norms, which promotes trust in the agricultural goods that are grown with the app's help.

Empowering Smallholder Farmers: Smallholder farmers, who frequently struggle to obtain access to contemporary agricultural technologies, are greatly assisted by the app. The "Plantie" app supports small-scale farmers by overcoming the technology gap. This allows them to compete in the larger agricultural scene and may even improve their socioeconomic circumstances.

User Engagement and Community Building: The community aspect of the "Plantie" app allows farmers to actively participate in debates, knowledge exchanges, and mutual support, hence fostering user engagement. The software can improve social interaction among farmers, encourage best practices, and empower individuals through shared expertise by fostering a collaborative farming community.

Empowerment through Education: The app serves as a central location for education, offering farmers chances for ongoing education beyond the diagnosis of illnesses. Using educational materials, guides, and updates on contemporary farming techniques, the application transforms into a platform for exchanging knowledge, encouraging a mindset of ongoing enhancement and creativity among farmers.

Accessibility and Inclusivity: The "Plantie" app was created with inclusivity in mind, considering the varied demands of farmers, particularly those who live in isolated locations with poor internet

or smartphone access. The goal of the app is to target neglected farming areas and develop offline functionality strategies so that all users, regardless of technological limitations, may continue to use the vital features.

**Long-Term Effect and Sustainable Agriculture:** The "Plantie" app hopes to have a long-term influence on farming methods and the overall agricultural environment, going beyond quick disease identification. Through the promotion of environmentally conscious farming practices, accurate disease control information, and community building, the app hopes to further agricultural technological advancements and the uptake of sustainable farming approaches.

In conclusion, by empowering farmers, encouraging knowledge exchange, and improving the sustainability of farming operations, the "Plantie" app has the potential to have a significant impact on agriculture and society. The app aims to transform farmer-to-farmer interactions, education, and crop cultivation through its novel approach to disease detection, community development, and inclusivity. This will ultimately contribute to a more robust and interconnected agricultural ecosystem.

## **7 CHAPTER 7: CONCLUSION AND RECOMMENDATION**

### **7.1 Summary of Achievements of the Project Objectives**

Plant diseases are still mostly detected using traditional techniques, given the availability of modern, advanced technologies. However, these techniques frequently lack precision and can be time-consuming. To solve this, our project is creating Plantie, a mobile app that uses deep learning algorithms to improve and simplify the diagnosis process.

Key Features and Innovations:

- Deep Learning Techniques: Our system analyses plant photos taken using a smartphone camera with accuracy by leveraging the use of deep learning techniques CNN with MobileNetV2 architecture. The application achieves a high overall accuracy of 98.7%.
- Disease Detection from Images: To identify diseases in particular plant species, the system is trained on a dataset of plants. The device can detect and identify diseases in apples, beans, corn, cucumbers, grapes, olives, peppers, potatoes, strawberries, and tomatoes by taking a picture of the plant.
- Advice for Correct Treatment: After a plant disease is identified, the application guides farmers by providing comprehensive information on efficient treatments.
- Nearest Plant Shop: Plantie, which integrates with Google Maps, assists users in finding the closest plant shops. This ensures that users can quickly obtain the resources they need and saves important time.
- Additional Features:
  - User-Friendly Design: Plantie's user-friendly interface makes it available to all farmers.
  - Scalable and extensible system: The app's architecture is designed to accommodate upcoming feature additions and expansions, guaranteeing long-term usage.
  - Weather-Based Suggestions: Farmers may make well-informed decisions based on the present conditions by using real-time weather information and customized plant care advice.

- Community System: For users to share information, tips, and exchange experiences, fostering a collaborative farming community.
- Fertilizer Calculation System: a tool that optimizes plant nutrition by offering customized fertilizer recommendations based on NPK values and cultivated area.

## **7.2 New Skills and Experiences Learn**

The ability to split the work between us and efficiently manage our time was the most important skill we learned. We acquired new skills in designing algorithms using the Draw.io application to understand how system components communicate with each other and identify any conflicts or delays. Additionally, learned to design application master pages using Figma, which helped us understand how main pages work. Also acquired knowledge about CNN algorithms and image processing, as well as principles of coordination, report writing, and finding similar systems to our own.

## **7.3 Recommendations for Future Work**

The desired outcome from this project is a mobile application that provides farmers the ability to detect plant disease, which saves time, effort, and money. In addition, the application will provide the farmers the common treatment for specific plant diseases. The application is an assistant for the farmers which reduces the need for experts there's no need to waste much time in trying to predict the desired disease within the possibility of errors occurring with associated documentation that describes the project requirements and project features in advance.

### **Future Work Recommendations:**

- Expanded Classification of Diseases:

Expand the model's capacity for classification to encompass a wider variety of plants and diseases. This will increase the application's adaptability and benefit a larger group of farmers.

- Training on Local Data:

To increase the relevance and accuracy of disease diagnosis for particular areas, use local datasets. This can assist in detecting diseases unique to a given area and provide recommendations for more potent treatments.

- Community Engagement Features:

Improve the community system by adding user levels or ranks that are determined by contributions and participation. This may promote increased user involvement and information exchange, building a more robust and encouraging community.

- Predictive Analytics:

Provide predictive analytics tools that, using past data and present environmental circumstances, can predict possible disease outbreaks. This can assist farmers in proactively taking preventive action.

- Combining IoT Device Integration:

Examine how to include Internet of Things (IoT) sensors to monitor environmental variables and plant health in real time. IoT sensors can supply constant data streams that boost the disease detection system's precision and responsiveness.

- User Feedback method:

To get opinions and recommendations from farmers using the app, put in place a user feedback method. The input provided by users can be used to address any problems and make ongoing improvements.

By concentrating on these areas, future updates can greatly improve the Plantie app's functionality and efficacy, making it an even more useful tool for farmers all around the world. These enhancements will increase the impact of the application on agricultural sustainability and productivity while also expanding the range of services available.

## REFERENCES

- [1] S. W. L. P. S. e. a. Savary, "The global burden of pathogens and pests on major food crops," *Nature Ecology & Evolution*, vol. 3, no. 3, pp. 430 - 439, 2019.
- [2] معاذ, "الذكاء الاصطناعي في الزراعة," arsco, 30 January 2023. [Online]. Available: <https://arsco.org/article-detail-32292-4-0?fbclid=IwAR1Ux6kRuOmMgcGXGkErZ5Scdi-VQwWAdmSkdQuNFDUee3OM3gsj26BdgLE>.
- [3] PlantVillage, "PlantVillage Nuru," google\_logo Play, 15 Aug 2023. [Online]. Available: <https://play.google.com/store/apps/details?id=plantvillage.nuru&hl=en&gl=US>.
- [4] plantix, plantix, [Online]. Available: <https://plantix.net/en/>. [Accessed 5 Jan 2024].
- [5] codebasics, "Deep learning project end to end | Potato Disease Classification Using CNN," YouTube, 22 Aug 2021. [Online]. Available: <https://www.youtube.com/watch?v=dGtDTjYs3xc>.
- [6] K. Z. G. S. K. A. Saman M. Omer, "An Intelligent System for Cucumber Leaf Disease Diagnosis," *Mobile Information Systems*, vol. 2022, p. 16, 2022.
- [7] sinanuguz, "CNN olive dataset," github, 2021. [Online]. Available: [https://github.com/sinanuguz/CNN\\_olive\\_dataset](https://github.com/sinanuguz/CNN_olive_dataset).
- [8] AI-Lab-Makerere, "ibean," Github, 2020 . [Online]. Available: <https://github.com/AI-Lab-Makerere/ibean/>.
- [9] "A Complete Guide to Data Augmentation," datacamp, Nov 2022. [Online]. Available: <https://www.datacamp.com/tutorial/complete-guide-data-augmentation>.
- [10] J. Nelson, "What is Image Preprocessing and Augmentation?," Roboflow, 26 JAN 2020. [Online]. Available: <https://blog.roboflow.com/why-preprocess-augment/>.
- [11] L. Craig, "convolutional neural network (CNN)," techtarget, Jan 2024. [Online]. Available: <https://www.techtarget.com/searchenterpriseai/definition/convolutional-neural-network>.

- [12] savyakhosla, "CNN | Introduction to Pooling Layer," geeksforgeeks, 21 Apr 2023. [Online]. Available: <https://www.geeksforgeeks.org/cnn-introduction-to-pooling-layer/>.
- [13] "LeNet," Wikipedia, 10 Dec 2023. [Online]. Available: <https://en.wikipedia.org/wiki/LeNet#:~:text=In%20general%2C%20LeNet%20refers%20to,in%20large%2Dscale%20image%20processing..>
- [14] D. BEAULIEU, "What Is NPK Fertilizer?," thespruce, 02 03 2023. [Online]. Available: <https://www.thespruce.com/what-does-npk-mean-for-a-fertilizer-2131094>.
- [15] M. hasanuzzaman, "Study on manures and fertilizers".
- [16] wikipedia, "Kaggle," wikipedia, 11 December 2023. [Online]. Available: <https://en.wikipedia.org/wiki/Kaggle>.
- [17] "Why is Flutter the best choice for building a mobile app Minimum Viable Product?," appunite, 5 Sep 2022. [Online]. Available: <https://appunite.com/blog/why-is-flutter-the-best-choice-for-building-a-mobile-app-minimum-viable-product>.
- [18] D. Stevenson, "What is Firebase? The complete story, abridged.," medium, 25 Sep 2018. [Online]. Available: <https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0>.
- [19] V. V. S. I. J. S. Christian Szegedy, "Rethinking the Inception Architecture for Computer Vision," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818-2826, 2016.
- [20] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," *Computers and Electronics in Agriculture*, vol. 145, pp. 311-318, 2018.